

Gazebo
3.0.2

Generated by Doxygen 1.8.1.2

Fri Apr 11 2014 18:07:34

Contents

1	Gazebo API Reference	1
2	Todo List	3
3	Module Index	5
3.1	Modules	5
4	Namespace Index	7
4.1	Namespace List	7
5	Class Index	9
5.1	Class Hierarchy	9
6	Class Index	17
6.1	Class List	17
7	File Index	29
7.1	File List	29
8	Module Documentation	35
8.1	Common	35
8.1.1	Detailed Description	39
8.1.2	Macro Definition Documentation	39
8.1.2.1	gzdbg	39
8.1.2.2	gzerr	39
8.1.2.3	gzlog	40
8.1.2.4	gzLogInit	40
8.1.2.5	gzmsg	40
8.1.2.6	gzthrow	40
8.1.2.7	gzwarn	40
8.1.3	Enumeration Type Documentation	40

8.1.3.1	PluginType	40
8.1.4	Function Documentation	41
8.1.4.1	MovingWindowFilter	41
8.1.4.2	MovingWindowFilterPrivate	41
8.1.4.3	~MovingWindowFilter	41
8.1.4.4	add_search_path_suffix	41
8.1.4.5	DownloadDependencies	41
8.1.4.6	find_file	41
8.1.4.7	find_file	42
8.1.4.8	find_file_path	42
8.1.4.9	Fini	42
8.1.4.10	Get	42
8.1.4.11	get_sha1	42
8.1.4.12	GetDBConfig	43
8.1.4.13	GetModelConfig	43
8.1.4.14	GetModelFile	43
8.1.4.15	GetModelName	43
8.1.4.16	GetModelPath	44
8.1.4.17	GetModels	44
8.1.4.18	GetModels	44
8.1.4.19	GetURI	44
8.1.4.20	GetWindowFilled	44
8.1.4.21	GetWindowSize	45
8.1.4.22	HasModel	45
8.1.4.23	load	45
8.1.4.24	SetWindowSize	45
8.1.4.25	Start	45
8.1.4.26	Update	45
8.1.5	Variable Documentation	46
8.1.5.1	PixelFormatNames	46
8.2	Events	47
8.2.1	Detailed Description	47
8.2.2	Function Documentation	47
8.2.2.1	EventT	47
8.2.2.2	~EventT	48
8.2.2.3	Connect	48
8.2.2.4	ConnectionCount	48

8.2.2.5	Disconnect	48
8.2.2.6	Disconnect	48
8.3	Math	50
8.3.1	Detailed Description	52
8.3.2	Function Documentation	52
8.3.2.1	clamp	52
8.3.2.2	equal	52
8.3.2.3	fixnan	52
8.3.2.4	fixnan	52
8.3.2.5	isnan	53
8.3.2.6	isnan	53
8.3.2.7	isPowerOfTwo	53
8.3.2.8	max	54
8.3.2.9	mean	54
8.3.2.10	min	54
8.3.2.11	parseFloat	54
8.3.2.12	parseInt	55
8.3.2.13	precision	55
8.3.2.14	roundUpPowerOfTwo	55
8.3.2.15	variance	55
8.3.3	Variable Documentation	56
8.3.3.1	NAN_D	56
8.3.3.2	NAN_I	56
8.4	Messages	57
8.4.1	Detailed Description	59
8.4.2	Macro Definition Documentation	59
8.4.2.1	GZ_REGISTER_STATIC_MSG	59
8.4.3	Function Documentation	59
8.4.3.1	Convert	59
8.4.3.2	Convert	59
8.4.3.3	Convert	60
8.4.3.4	Convert	60
8.4.3.5	Convert	60
8.4.3.6	Convert	60
8.4.3.7	Convert	61
8.4.3.8	Convert	61
8.4.3.9	Convert	61

8.4.3.10	Convert	61
8.4.3.11	Convert	62
8.4.3.12	Convert	62
8.4.3.13	CreateRequest	62
8.4.3.14	FogFromSDF	62
8.4.3.15	GeometryFromSDF	63
8.4.3.16	GetHeader	63
8.4.3.17	GUIFromSDF	63
8.4.3.18	Init	63
8.4.3.19	LightFromSDF	64
8.4.3.20	MeshFromSDF	64
8.4.3.21	SceneFromSDF	64
8.4.3.22	Set	64
8.4.3.23	Set	65
8.4.3.24	Set	65
8.4.3.25	Set	65
8.4.3.26	Set	65
8.4.3.27	Set	65
8.4.3.28	Set	65
8.4.3.29	Set	66
8.4.3.30	Set	66
8.4.3.31	Set	66
8.4.3.32	Stamp	66
8.4.3.33	Stamp	66
8.4.3.34	TrackVisualFromSDF	67
8.4.3.35	VisualFromSDF	67
8.5	Classes for physics and dynamics	68
8.5.1	Detailed Description	72
8.5.2	Macro Definition Documentation	72
8.5.2.1	GZ_REGISTER_PHYSICS_ENGINE	72
8.5.3	Typedef Documentation	72
8.5.3.1	PhysicsFactoryFn	72
8.5.4	Function Documentation	72
8.5.4.1	create_world	72
8.5.4.2	fini	72
8.5.4.3	get_world	72
8.5.4.4	getUniqueld	73

8.5.4.5	init_world	73
8.5.4.6	init_worlds	73
8.5.4.7	load	73
8.5.4.8	load_world	73
8.5.4.9	load_worlds	73
8.5.4.10	pause_world	74
8.5.4.11	pause_worlds	74
8.5.4.12	remove_worlds	74
8.5.4.13	run_world	74
8.5.4.14	run_worlds	74
8.5.4.15	stop_world	74
8.5.4.16	stop_worlds	75
8.5.4.17	worlds_running	75
8.5.5	Variable Documentation	75
8.5.5.1	EntityTypename	75
8.6	DART Physics	76
8.6.1	Detailed Description	77
8.6.2	Function Documentation	77
8.6.2.1	DARTRayShape	77
8.6.2.2	DARTRayShape	77
8.6.2.3	~DARTRayShape	77
8.6.2.4	ConvPose	77
8.6.2.5	ConvPose	77
8.6.2.6	ConvQuat	78
8.6.2.7	ConvQuat	78
8.6.2.8	ConvVec3	78
8.6.2.9	ConvVec3	78
8.6.2.10	GetIntersection	78
8.6.2.11	SetPoints	78
8.6.2.12	Update	78
8.7	Bullet Physics	79
8.7.1	Detailed Description	79
8.7.2	Function Documentation	79
8.7.2.1	DARTMultiRayShape	79
8.7.2.2	~DARTMultiRayShape	79
8.7.2.3	AddRay	80
8.7.2.4	UpdateRays	80

8.8	Simbody Physics	81
8.8.1	Detailed Description	82
8.9	Rendering	83
8.9.1	Detailed Description	85
8.9.2	Function Documentation	85
8.9.2.1	create_scene	85
8.9.2.2	fini	85
8.9.2.3	get_scene	85
8.9.2.4	init	86
8.9.2.5	load	86
8.9.2.6	remove_scene	86
8.10	Sensors	87
8.10.1	Detailed Description	89
8.10.2	Macro Definition Documentation	89
8.10.2.1	GZ_REGISTER_STATIC_SENSOR	89
8.10.3	Function Documentation	89
8.10.3.1	create_sensor	89
8.10.3.2	create_sensor	89
8.10.3.3	disable	90
8.10.3.4	enable	90
8.10.3.5	fini	90
8.10.3.6	get_sensor	90
8.10.3.7	init	90
8.10.3.8	load	90
8.10.3.9	remove_sensor	91
8.10.3.10	remove_sensors	91
8.10.3.11	run_once	91
8.10.3.12	run_threads	91
8.10.3.13	stop	91
8.11	Transport	92
8.11.1	Detailed Description	94
8.11.2	Typedef Documentation	94
8.11.2.1	CallbackHelperPtr	94
8.11.3	Function Documentation	94
8.11.3.1	clear_buffers	94
8.11.3.2	connectToMaster	94
8.11.3.3	fini	94

8.11.3.4	get_master_uri	95
8.11.3.5	get_topic_namespaces	95
8.11.3.6	getAdvertisedTopics	95
8.11.3.7	getAdvertisedTopics	95
8.11.3.8	getMinimalComms	95
8.11.3.9	getTopicMsgType	96
8.11.3.10	init	96
8.11.3.11	is_stopped	96
8.11.3.12	pause_incoming	96
8.11.3.13	publish	97
8.11.3.14	request	97
8.11.3.15	requestNoReply	97
8.11.3.16	requestNoReply	97
8.11.3.17	run	98
8.11.3.18	setMinimalComms	98
8.11.3.19	stop	98
8.11.3.20	waitForNamespaces	98
8.12	Utility	99
8.12.1	Detailed Description	99
8.12.2	Macro Definition Documentation	99
8.12.2.1	DIAG_TIMER_LAP	99
8.12.2.2	DIAG_TIMER_START	99
8.12.2.3	DIAG_TIMER_STOP	99
9	Namespace Documentation	101
9.1	boost Namespace Reference	101
9.2	gazebo Namespace Reference	101
9.2.1	Detailed Description	103
9.2.2	Typedef Documentation	103
9.2.2.1	GUIPluginPtr	103
9.2.2.2	ModelPluginPtr	103
9.2.2.3	SensorPluginPtr	103
9.2.2.4	SystemPluginPtr	103
9.2.2.5	VisualPluginPtr	103
9.2.2.6	WorldPluginPtr	103
9.2.3	Function Documentation	103
9.2.3.1	add_plugin	103

9.2.3.2	addPlugin	104
9.2.3.3	find_file	104
9.2.3.4	fini	104
9.2.3.5	init	104
9.2.3.6	load	104
9.2.3.7	loadWorld	104
9.2.3.8	print_version	105
9.2.3.9	printVersion	105
9.2.3.10	run	105
9.2.3.11	runWorld	105
9.2.3.12	setupClient	105
9.2.3.13	setupServer	106
9.2.3.14	shutdown	106
9.2.3.15	stop	106
9.3	gazebo::common Namespace Reference	106
9.3.1	Detailed Description	109
9.3.2	Typedef Documentation	110
9.3.2.1	AnimationPtr	110
9.3.2.2	DiagnosticTimerPtr	110
9.3.2.3	NodeMap	110
9.3.2.4	NodeMapIter	110
9.3.2.5	NumericAnimationPtr	110
9.3.2.6	Param_V	110
9.3.2.7	PoseAnimationPtr	110
9.3.2.8	RawNodeAnim	110
9.3.2.9	RawNodeWeights	110
9.3.2.10	RawSkeletonAnim	110
9.3.2.11	SphericalCoordinatesPtr	110
9.3.2.12	StrStr_M	110
9.3.3	Variable Documentation	110
9.3.3.1	SpeedOfLight	110
9.4	gazebo::event Namespace Reference	110
9.4.1	Detailed Description	111
9.4.2	Typedef Documentation	111
9.4.2.1	Connection_V	111
9.4.2.2	ConnectionPtr	111
9.5	gazebo::math Namespace Reference	111

9.5.1	Detailed Description	113
9.5.2	Typedef Documentation	113
9.5.2.1	GeneratorType	113
9.5.2.2	NormalRealDist	113
9.5.2.3	NRealGen	113
9.5.2.4	UIntGen	113
9.5.2.5	UniformIntDist	113
9.5.2.6	UniformRealDist	113
9.5.2.7	URealGen	113
9.6	gazebo::msgs Namespace Reference	113
9.6.1	Detailed Description	115
9.6.2	Typedef Documentation	115
9.6.2.1	MsgFactoryFn	115
9.7	gazebo::physics Namespace Reference	115
9.7.1	Detailed Description	122
9.7.2	Typedef Documentation	122
9.7.2.1	Actor_V	122
9.7.2.2	ActorPtr	122
9.7.2.3	Base_V	122
9.7.2.4	BasePtr	122
9.7.2.5	BoxShapePtr	122
9.7.2.6	Collision_V	122
9.7.2.7	CollisionPtr	122
9.7.2.8	ContactPtr	122
9.7.2.9	CylinderShapePtr	122
9.7.2.10	DARTCollisionPtr	122
9.7.2.11	DARTJointPtr	122
9.7.2.12	DARTLinkPtr	122
9.7.2.13	DARTModelPtr	122
9.7.2.14	DARTPhysicsPtr	122
9.7.2.15	DARTRayShapePtr	122
9.7.2.16	EntityPtr	122
9.7.2.17	GripperPtr	122
9.7.2.18	HeightmapShapePtr	122
9.7.2.19	InertialPtr	122
9.7.2.20	Joint_V	122
9.7.2.21	JointController_V	122

9.7.2.22	JointControllerPtr	122
9.7.2.23	JointPtr	123
9.7.2.24	JointState_M	123
9.7.2.25	Link_V	123
9.7.2.26	LinkPtr	123
9.7.2.27	LinkState_M	123
9.7.2.28	MeshShapePtr	123
9.7.2.29	Model_V	123
9.7.2.30	ModelPtr	123
9.7.2.31	ModelState_M	123
9.7.2.32	MultiRayShapePtr	123
9.7.2.33	PhysicsEnginePtr	123
9.7.2.34	RayShapePtr	123
9.7.2.35	RoadPtr	123
9.7.2.36	ShapePtr	123
9.7.2.37	SimbodyCollisionPtr	123
9.7.2.38	SimbodyLinkPtr	123
9.7.2.39	SimbodyModelPtr	123
9.7.2.40	SimbodyPhysicsPtr	123
9.7.2.41	SimbodyRayShapePtr	123
9.7.2.42	SphereShapePtr	123
9.7.2.43	SurfaceParamsPtr	123
9.7.2.44	WorldPtr	123
9.8	gazebo::rendering Namespace Reference	123
9.8.1	Detailed Description	127
9.8.2	Typedef Documentation	127
9.8.2.1	ArrowVisualPtr	127
9.8.2.2	AxisVisualPtr	127
9.8.2.3	CameraPtr	128
9.8.2.4	CameraVisualPtr	128
9.8.2.5	COMVisualPtr	128
9.8.2.6	ContactVisualPtr	128
9.8.2.7	DepthCameraPtr	128
9.8.2.8	DynamicLinesPtr	128
9.8.2.9	GpuLaserPtr	128
9.8.2.10	JointVisualPtr	128
9.8.2.11	LaserVisualPtr	128

9.8.2.12	LightPtr	128
9.8.2.13	RFIDTagVisualPtr	128
9.8.2.14	RFIDVisualPtr	128
9.8.2.15	ScenePtr	128
9.8.2.16	SelectionObjPtr	128
9.8.2.17	SonarVisualPtr	128
9.8.2.18	UserCameraPtr	128
9.8.2.19	VisualPtr	128
9.8.2.20	WindowManagerPtr	128
9.8.2.21	WrenchVisualPtr	128
9.8.3	Enumeration Type Documentation	128
9.8.3.1	RenderOpType	128
9.9	gazebo::sensors Namespace Reference	129
9.9.1	Detailed Description	132
9.9.2	Typedef Documentation	132
9.9.2.1	CameraSensor_V	132
9.9.2.2	CameraSensorPtr	132
9.9.2.3	ContactSensor_V	132
9.9.2.4	ContactSensorPtr	132
9.9.2.5	DepthCameraSensor_V	132
9.9.2.6	DepthCameraSensorPtr	132
9.9.2.7	ForceTorqueSensorPtr	132
9.9.2.8	GaussianNoiseModelPtr	132
9.9.2.9	GpsSensorPtr	132
9.9.2.10	GpuRaySensor_V	132
9.9.2.11	GpuRaySensorPtr	132
9.9.2.12	ImageGaussianNoiseModelPtr	132
9.9.2.13	ImuSensor_V	132
9.9.2.14	ImuSensorPtr	132
9.9.2.15	MultiCameraSensor_V	132
9.9.2.16	MultiCameraSensorPtr	132
9.9.2.17	NoisePtr	132
9.9.2.18	RaySensor_V	132
9.9.2.19	RaySensorPtr	133
9.9.2.20	RFIDSensor_V	133
9.9.2.21	RFIDSensorPtr	133
9.9.2.22	RFIDTag_V	133

9.9.2.23	RFIDTagPtr	133
9.9.2.24	Sensor_V	133
9.9.2.25	SensorFactoryFn	133
9.9.2.26	SensorPtr	133
9.9.2.27	SonarSensorPtr	133
9.9.2.28	WirelessReceiver_V	133
9.9.2.29	WirelessReceiverPtr	133
9.9.2.30	WirelessTransceiver_V	133
9.9.2.31	WirelessTransceiverPtr	133
9.9.2.32	WirelessTransmitter_V	133
9.9.2.33	WirelessTransmitterPtr	133
9.9.3	Enumeration Type Documentation	133
9.9.3.1	SensorCategory	133
9.10	gazebo::transport Namespace Reference	133
9.10.1	Typedef Documentation	136
9.10.1.1	ConnectionPtr	136
9.10.1.2	MessagePtr	136
9.10.1.3	NodePtr	136
9.10.1.4	PublicationPtr	136
9.10.1.5	PublicationTransportPtr	136
9.10.1.6	PublisherPtr	136
9.10.1.7	SubscriberPtr	136
9.10.1.8	SubscriptionTransportPtr	136
9.11	gazebo::util Namespace Reference	136
9.11.1	Typedef Documentation	137
9.11.1.1	DiagnosticTimerPtr	137
9.11.1.2	OpenALSinkPtr	137
9.11.1.3	OpenALSourcePtr	137
9.12	google Namespace Reference	137
9.13	google::protobuf Namespace Reference	137
9.14	google::protobuf::compiler Namespace Reference	137
9.15	google::protobuf::compiler::cpp Namespace Reference	137
9.16	Ogre Namespace Reference	137
9.17	ogre Namespace Reference	137
9.18	SimTK Namespace Reference	137
9.19	SkyX Namespace Reference	137

10 Class Documentation	139
10.1 gazebo::physics::Actor Class Reference	139
10.1.1 Detailed Description	141
10.1.2 Constructor & Destructor Documentation	141
10.1.2.1 Actor	141
10.1.2.2 ~Actor	142
10.1.3 Member Function Documentation	142
10.1.3.1 Fini	142
10.1.3.2 GetSDF	142
10.1.3.3 Init	142
10.1.3.4 IsActive	142
10.1.3.5 Load	142
10.1.3.6 Play	142
10.1.3.7 Stop	142
10.1.3.8 Update	143
10.1.3.9 UpdateParameters	143
10.1.4 Member Data Documentation	143
10.1.4.1 active	143
10.1.4.2 autoStart	143
10.1.4.3 bonePosePub	143
10.1.4.4 interpolateX	143
10.1.4.5 lastPos	143
10.1.4.6 lastScriptTime	143
10.1.4.7 lastTraj	143
10.1.4.8 loop	144
10.1.4.9 mainLink	144
10.1.4.10 mesh	144
10.1.4.11 oldAction	144
10.1.4.12 pathLength	144
10.1.4.13 playStartTime	144
10.1.4.14 prevFrameTime	144
10.1.4.15 scriptLength	144
10.1.4.16 skelAnimation	144
10.1.4.17 skeleton	144
10.1.4.18 skelNodesMap	144
10.1.4.19 skinFile	145
10.1.4.20 skinScale	145

10.1.4.21	startDelay	145
10.1.4.22	trajectories	145
10.1.4.23	trajInfo	145
10.1.4.24	visualId	145
10.1.4.25	visualName	145
10.2	gazebo::math::Angle Class Reference	145
10.2.1	Detailed Description	147
10.2.2	Constructor & Destructor Documentation	147
10.2.2.1	Angle	147
10.2.2.2	Angle	147
10.2.2.3	Angle	147
10.2.2.4	~Angle	147
10.2.3	Member Function Documentation	148
10.2.3.1	Degree	148
10.2.3.2	Normalize	148
10.2.3.3	operator!=	148
10.2.3.4	operator*	148
10.2.3.5	operator*	148
10.2.3.6	operator*==	148
10.2.3.7	operator+	149
10.2.3.8	operator+=	149
10.2.3.9	operator-	149
10.2.3.10	operator-=	149
10.2.3.11	operator/	150
10.2.3.12	operator/=	150
10.2.3.13	operator<	150
10.2.3.14	operator<=	150
10.2.3.15	operator==	151
10.2.3.16	operator>	151
10.2.3.17	operator>=	151
10.2.3.18	Radian	151
10.2.3.19	SetFromDegree	152
10.2.3.20	SetFromRadian	152
10.2.4	Friends And Related Function Documentation	152
10.2.4.1	operator<<	152
10.2.4.2	operator>>	152
10.2.5	Member Data Documentation	153

10.2.5.1	HalfPi	153
10.2.5.2	Pi	153
10.2.5.3	TwoPi	153
10.2.5.4	Zero	153
10.3	gazebo::common::Animation Class Reference	153
10.3.1	Detailed Description	154
10.3.2	Member Typedef Documentation	155
10.3.2.1	KeyFrame_V	155
10.3.3	Constructor & Destructor Documentation	155
10.3.3.1	Animation	155
10.3.3.2	~Animation	155
10.3.4	Member Function Documentation	155
10.3.4.1	AddTime	155
10.3.4.2	GetKeyFrame	155
10.3.4.3	GetKeyFrameCount	155
10.3.4.4	GetKeyFramesAtTime	156
10.3.4.5	GetLength	156
10.3.4.6	GetTime	156
10.3.4.7	SetLength	156
10.3.4.8	SetTime	156
10.3.5	Member Data Documentation	157
10.3.5.1	build	157
10.3.5.2	keyFrames	157
10.3.5.3	length	157
10.3.5.4	loop	157
10.3.5.5	name	157
10.3.5.6	timePos	157
10.4	gazebo::rendering::ArrowVisual Class Reference	157
10.4.1	Detailed Description	158
10.4.2	Constructor & Destructor Documentation	158
10.4.2.1	ArrowVisual	158
10.4.2.2	~ArrowVisual	159
10.4.3	Member Function Documentation	159
10.4.3.1	Load	159
10.4.3.2	ShowRotation	159
10.5	gazebo::rendering::ArrowVisualPrivate Class Reference	159
10.5.1	Detailed Description	160

10.5.2	Member Data Documentation	160
10.5.2.1	headNode	160
10.5.2.2	rotationNode	160
10.5.2.3	shaftNode	160
10.6	gazebo::common::AssertionInternalError Class Reference	160
10.6.1	Detailed Description	161
10.6.2	Constructor & Destructor Documentation	161
10.6.2.1	AssertionInternalError	161
10.6.2.2	~AssertionInternalError	161
10.7	gazebo::common::AudioDecoder Class Reference	161
10.7.1	Detailed Description	162
10.7.2	Constructor & Destructor Documentation	162
10.7.2.1	AudioDecoder	162
10.7.2.2	~AudioDecoder	162
10.7.3	Member Function Documentation	162
10.7.3.1	Decode	162
10.7.3.2	GetFile	162
10.7.3.3	GetSampleRate	163
10.7.3.4	SetFile	163
10.8	gazebo::rendering::AxisVisual Class Reference	163
10.8.1	Detailed Description	164
10.8.2	Constructor & Destructor Documentation	164
10.8.2.1	AxisVisual	164
10.8.2.2	~AxisVisual	164
10.8.3	Member Function Documentation	164
10.8.3.1	Load	164
10.8.3.2	ScaleXAxis	165
10.8.3.3	ScaleYAxis	165
10.8.3.4	ScaleZAxis	165
10.8.3.5	SetAxisMaterial	165
10.8.3.6	ShowRotation	165
10.9	gazebo::rendering::AxisVisualPrivate Class Reference	165
10.9.1	Detailed Description	166
10.9.2	Member Data Documentation	166
10.9.2.1	xAxis	166
10.9.2.2	yAxis	166
10.9.2.3	zAxis	166

10.10gazebo::physics::BallJoint< T > Class Template Reference	167
10.10.1 Detailed Description	167
10.10.2 Constructor & Destructor Documentation	167
10.10.2.1 BallJoint	167
10.10.2.2 ~BallJoint	167
10.10.3 Member Function Documentation	168
10.10.3.1 GetAngleCount	168
10.10.3.2 Init	168
10.10.3.3 Load	168
10.11gazebo::physics::Base Class Reference	168
10.11.1 Detailed Description	172
10.11.2 Member Enumeration Documentation	172
10.11.2.1 EntityType	172
10.11.3 Constructor & Destructor Documentation	173
10.11.3.1 Base	173
10.11.3.2 ~Base	173
10.11.4 Member Function Documentation	173
10.11.4.1 AddChild	173
10.11.4.2 AddType	173
10.11.4.3 ComputeScopedName	173
10.11.4.4 Fini	174
10.11.4.5 GetById	174
10.11.4.6 GetByName	174
10.11.4.7 GetChild	174
10.11.4.8 GetChild	174
10.11.4.9 GetChildCount	175
10.11.4.10GetId	175
10.11.4.11GetName	175
10.11.4.12GetParent	175
10.11.4.13GetParentId	175
10.11.4.14GetSaveable	176
10.11.4.15GetScopedName	176
10.11.4.16GetSDF	176
10.11.4.17GetType	176
10.11.4.18GetWorld	176
10.11.4.19HasType	176
10.11.4.20Init	177

10.11.4.21	IsSelected	177
10.11.4.22	Load	177
10.11.4.23	operator==	178
10.11.4.24	Print	178
10.11.4.25	RemoveChild	178
10.11.4.26	RemoveChild	178
10.11.4.27	RemoveChildren	179
10.11.4.28	Reset	179
10.11.4.29	Reset	179
10.11.4.30	SetName	179
10.11.4.31	SetParent	179
10.11.4.32	SetSaveable	179
10.11.4.33	SetSelected	179
10.11.4.34	SetWorld	180
10.11.4.35	Update	180
10.11.4.36	UpdateParameters	180
10.11.5	Member Data Documentation	180
10.11.5.1	children	180
10.11.5.2	parent	180
10.11.5.3	sdf	180
10.11.5.4	world	181
10.12	gazebo::math::Box Class Reference	181
10.12.1	Detailed Description	182
10.12.2	Constructor & Destructor Documentation	182
10.12.2.1	Box	182
10.12.2.2	Box	182
10.12.2.3	Box	182
10.12.2.4	~Box	182
10.12.3	Member Function Documentation	182
10.12.3.1	GetCenter	183
10.12.3.2	GetSize	183
10.12.3.3	GetXLength	183
10.12.3.4	GetYLength	183
10.12.3.5	GetZLength	183
10.12.3.6	Merge	183
10.12.3.7	operator+	184
10.12.3.8	operator+=	184

10.12.3.9 operator-	184
10.12.3.10 operator=	184
10.12.3.11 operator==	185
10.12.4 Friends And Related Function Documentation	185
10.12.4.1 operator<<	185
10.12.5 Member Data Documentation	185
10.12.5.1 max	185
10.12.5.2 min	185
10.13 gazebo::physics::BoxShape Class Reference	185
10.13.1 Detailed Description	186
10.13.2 Constructor & Destructor Documentation	187
10.13.2.1 BoxShape	187
10.13.2.2 ~BoxShape	187
10.13.3 Member Function Documentation	187
10.13.3.1 FillMsg	187
10.13.3.2 GetSize	187
10.13.3.3 Init	187
10.13.3.4 ProcessMsg	187
10.13.3.5 SetScale	188
10.13.3.6 SetSize	188
10.14 gazebo::common::Logger::Buffer Class Reference	188
10.14.1 Detailed Description	188
10.14.2 Constructor & Destructor Documentation	189
10.14.2.1 Buffer	189
10.14.2.2 ~Buffer	189
10.14.3 Member Function Documentation	189
10.14.3.1 sync	189
10.14.4 Member Data Documentation	189
10.14.4.1 color	189
10.14.4.2 type	189
10.15 gazebo::common::FileLogger::Buffer Class Reference	189
10.15.1 Detailed Description	190
10.15.2 Constructor & Destructor Documentation	190
10.15.2.1 Buffer	190
10.15.2.2 ~Buffer	190
10.15.3 Member Function Documentation	190
10.15.3.1 sync	190

10.15.4 Member Data Documentation	190
10.15.4.1 stream	190
10.16 gazebo::common::BVHLoader Class Reference	191
10.16.1 Detailed Description	191
10.16.2 Constructor & Destructor Documentation	191
10.16.2.1 BVHLoader	191
10.16.2.2 ~BVHLoader	191
10.16.3 Member Function Documentation	191
10.16.3.1 Load	191
10.17 gazebo::transport::CallbackHelper Class Reference	192
10.17.1 Detailed Description	193
10.17.2 Constructor & Destructor Documentation	193
10.17.2.1 CallbackHelper	193
10.17.2.2 ~CallbackHelper	193
10.17.3 Member Function Documentation	193
10.17.3.1 GetId	193
10.17.3.2 GetLatching	193
10.17.3.3 GetMsgType	193
10.17.3.4 HandleData	194
10.17.3.5 HandleMessage	194
10.17.3.6 IsLocal	194
10.17.3.7 SetLatching	194
10.17.4 Member Data Documentation	195
10.17.4.1 latching	195
10.18 gazebo::transport::CallbackHelperT< M > Class Template Reference	195
10.18.1 Detailed Description	196
10.18.2 Constructor & Destructor Documentation	196
10.18.2.1 CallbackHelperT	196
10.18.3 Member Function Documentation	196
10.18.3.1 GetMsgType	196
10.18.3.2 HandleData	196
10.18.3.3 HandleMessage	197
10.18.3.4 IsLocal	197
10.19 gazebo::rendering::Camera Class Reference	197
10.19.1 Detailed Description	204
10.19.2 Constructor & Destructor Documentation	204
10.19.2.1 Camera	204

10.19.2.2 ~Camera	204
10.19.3 Member Function Documentation	204
10.19.3.1 AnimationComplete	204
10.19.3.2 AttachToVisual	204
10.19.3.3 AttachToVisual	205
10.19.3.4 AttachToVisualImpl	205
10.19.3.5 AttachToVisualImpl	205
10.19.3.6 AttachToVisualImpl	206
10.19.3.7 ConnectNewImageFrame	206
10.19.3.8 CreateRenderTexture	206
10.19.3.9 DisconnectNewImageFrame	206
10.19.3.10EnableSaveFrame	207
10.19.3.11Fini	207
10.19.3.12GetAspectRatio	207
10.19.3.13GetAvgFPS	207
10.19.3.14GetCameraToViewportRay	207
10.19.3.15GetCaptureData	207
10.19.3.16GetDirection	208
10.19.3.17GetFarClip	208
10.19.3.18GetFrameFilename	208
10.19.3.19GetHFOV	208
10.19.3.20GetImageByteSize	208
10.19.3.21GetImageByteSize	208
10.19.3.22GetImageData	209
10.19.3.23GetImageDepth	209
10.19.3.24GetImageFormat	209
10.19.3.25GetImageHeight	209
10.19.3.26GetImageWidth	210
10.19.3.27GetInitialized	210
10.19.3.28GetLastRenderWallTime	210
10.19.3.29GetName	210
10.19.3.30GetNearClip	210
10.19.3.31GetOgreCamera	210
10.19.3.32GetPitchNode	211
10.19.3.33GetRenderRate	211
10.19.3.34GetRenderTexture	211
10.19.3.35GetRight	211

10.19.3.36	GetScene	211
10.19.3.37	GetSceneNode	211
10.19.3.38	GetScopedName	212
10.19.3.39	GetScreenshotPath	212
10.19.3.40	GetTextureHeight	212
10.19.3.41	GetTextureWidth	212
10.19.3.42	GetTriangleCount	212
10.19.3.43	GetUp	212
10.19.3.44	GetVFOV	213
10.19.3.45	GetViewport	213
10.19.3.46	GetViewportHeight	213
10.19.3.47	GetViewportWidth	213
10.19.3.48	GetWindowId	213
10.19.3.49	GetWorldPointOnPlane	213
10.19.3.50	GetWorldPose	214
10.19.3.51	GetWorldPosition	214
10.19.3.52	GetWorldRotation	214
10.19.3.53	GetZValue	214
10.19.3.54	Init	214
10.19.3.55	IsAnimating	215
10.19.3.56	IsVisible	215
10.19.3.57	IsVisible	215
10.19.3.58	Load	215
10.19.3.59	Load	215
10.19.3.60	MoveToPosition	215
10.19.3.61	MoveToPositions	216
10.19.3.62	PostRender	216
10.19.3.63	ReadPixelBuffer	216
10.19.3.64	Render	216
10.19.3.65	Render	216
10.19.3.66	RenderImpl	217
10.19.3.67	RotatePitch	217
10.19.3.68	RotateYaw	217
10.19.3.69	SaveFrame	217
10.19.3.70	SaveFrame	217
10.19.3.71	SetAspectRatio	218
10.19.3.72	SetCaptureData	218

10.19.3.73SetCaptureDataOnce	218
10.19.3.74SetClipDist	218
10.19.3.75SetHFOV	218
10.19.3.76SetImageHeight	218
10.19.3.77SetImageSize	219
10.19.3.78SetImageWidth	219
10.19.3.79SetName	219
10.19.3.80SetRenderRate	219
10.19.3.81SetRenderTarget	219
10.19.3.82SetSaveFramePathname	219
10.19.3.83SetScene	220
10.19.3.84SetSceneNode	220
10.19.3.85SetWindowId	220
10.19.3.86SetWorldPose	220
10.19.3.87SetWorldPosition	220
10.19.3.88SetWorldRotation	220
10.19.3.89ShowWireframe	221
10.19.3.90ToggleShowWireframe	221
10.19.3.91TrackVisual	221
10.19.3.92TrackVisualImpl	221
10.19.3.93TrackVisualImpl	221
10.19.3.94Translate	221
10.19.3.95Update	222
10.19.4 Member Data Documentation	222
10.19.4.1 animState	222
10.19.4.2 bayerFrameBuffer	222
10.19.4.3 camera	222
10.19.4.4 captureData	222
10.19.4.5 captureDataOnce	222
10.19.4.6 connections	222
10.19.4.7 imageFormat	222
10.19.4.8 imageHeight	222
10.19.4.9 imageWidth	222
10.19.4.10initialized	223
10.19.4.11lastRenderWallTime	223
10.19.4.12name	223
10.19.4.13newData	223

10.19.4.14	newImageFrame	223
10.19.4.15	onAnimationComplete	223
10.19.4.16	prevAnimTime	223
10.19.4.17	renderTarget	223
10.19.4.18	renderTexture	223
10.19.4.19	requests	223
10.19.4.20	saveCount	223
10.19.4.21	saveFrameBuffer	224
10.19.4.22	scene	224
10.19.4.23	sceneNode	224
10.19.4.24	scopedName	224
10.19.4.25	scopedUniqueName	224
10.19.4.26	screenshotPath	224
10.19.4.27	sdf	224
10.19.4.28	textureHeight	224
10.19.4.29	textureWidth	224
10.19.4.30	viewport	224
10.19.4.31	windowId	224
10.20	gazebo::rendering::CameraPrivate Class Reference	225
10.20.1	Detailed Description	226
10.20.2	Member Typedef Documentation	226
10.20.2.1	CameraCmdMsgs_L	226
10.20.3	Member Data Documentation	226
10.20.3.1	cameraCounter	226
10.20.3.2	cmdSub	226
10.20.3.3	commandMsgs	226
10.20.3.4	dIGBufferInstance	226
10.20.3.5	dIMergeInstance	226
10.20.3.6	dsGBufferInstance	226
10.20.3.7	dsMergeInstance	226
10.20.3.8	moveToPositionQueue	226
10.20.3.9	node	227
10.20.3.10	receiveMutex	227
10.20.3.11	renderPeriod	227
10.20.3.12	ssaoInstance	227
10.20.3.13	trackedVisual	227
10.20.3.14	trackVisualPID	227

10.20.3.15	trackVisualPitchPID	227
10.20.3.16	trackVisualYawPID	227
10.21	gazebo::sensors::CameraSensor Class Reference	227
10.21.1	Detailed Description	229
10.21.2	Constructor & Destructor Documentation	229
10.21.2.1	CameraSensor	229
10.21.2.2	~CameraSensor	229
10.21.3	Member Function Documentation	229
10.21.3.1	Finis	229
10.21.3.2	GetCamera	229
10.21.3.3	GetImageData	229
10.21.3.4	GetImageHeight	229
10.21.3.5	GetImageWidth	230
10.21.3.6	GetTopic	230
10.21.3.7	Init	230
10.21.3.8	IsActive	230
10.21.3.9	Load	230
10.21.3.10	Load	231
10.21.3.11	SaveFrame	231
10.21.3.12	UpdateImpl	231
10.22	gazebo::rendering::CameraVisual Class Reference	231
10.22.1	Detailed Description	232
10.22.2	Constructor & Destructor Documentation	232
10.22.2.1	CameraVisual	232
10.22.2.2	~CameraVisual	233
10.22.3	Member Function Documentation	233
10.22.3.1	Load	233
10.23	gazebo::rendering::CameraVisualPrivate Class Reference	233
10.23.1	Member Data Documentation	234
10.23.1.1	camera	234
10.23.1.2	connections	234
10.24	gazebo::common::ColladaLoader Class Reference	234
10.24.1	Detailed Description	234
10.24.2	Constructor & Destructor Documentation	235
10.24.2.1	ColladaLoader	235
10.24.2.2	~ColladaLoader	235
10.24.3	Member Function Documentation	235

10.24.3.1 Load	235
10.25 gazebo::physics::Collision Class Reference	235
10.25.1 Detailed Description	238
10.25.2 Constructor & Destructor Documentation	238
10.25.2.1 Collision	238
10.25.2.2 ~Collision	238
10.25.3 Member Function Documentation	238
10.25.3.1 AddContact	238
10.25.3.2 FillMsg	239
10.25.3.3 Fini	239
10.25.3.4 GetBoundingBox	239
10.25.3.5 GetContactsEnabled	239
10.25.3.6 GetLaserRetro	239
10.25.3.7 GetLink	239
10.25.3.8 GetMaxContacts	240
10.25.3.9 GetModel	240
10.25.3.10 GetRelativeAngularAccel	240
10.25.3.11 GetRelativeAngularVel	240
10.25.3.12 GetRelativeLinearAccel	240
10.25.3.13 GetRelativeLinearVel	241
10.25.3.14 GetShape	241
10.25.3.15 GetShapeType	241
10.25.3.16 GetState	241
10.25.3.17 GetSurface	241
10.25.3.18 GetWorldAngularAccel	242
10.25.3.19 GetWorldAngularVel	242
10.25.3.20 GetWorldLinearAccel	242
10.25.3.21 GetWorldLinearVel	242
10.25.3.22 Init	242
10.25.3.23 IsPlaceable	242
10.25.3.24 Load	243
10.25.3.25 ProcessMsg	243
10.25.3.26 SetCategoryBits	243
10.25.3.27 SetCollideBits	243
10.25.3.28 SetCollision	243
10.25.3.29 SetContactsEnabled	244
10.25.3.30 SetLaserRetro	244

10.25.3.31	setMaxContacts	244
10.25.3.32	setScale	244
10.25.3.33	setShape	244
10.25.3.34	setState	244
10.25.3.35	updateParameters	245
10.25.4	Member Data Documentation	245
10.25.4.1	link	245
10.25.4.2	placeable	245
10.25.4.3	shape	245
10.25.4.4	surface	245
10.26	gazebo::physics::CollisionState Class Reference	245
10.26.1	Detailed Description	247
10.26.2	Constructor & Destructor Documentation	247
10.26.2.1	CollisionState	247
10.26.2.2	CollisionState	247
10.26.2.3	CollisionState	247
10.26.2.4	~CollisionState	247
10.26.3	Member Function Documentation	247
10.26.3.1	FillSDF	247
10.26.3.2	GetPose	248
10.26.3.3	IsZero	248
10.26.3.4	Load	248
10.26.3.5	operator+	248
10.26.3.6	operator-	248
10.26.3.7	operator=	249
10.26.4	Friends And Related Function Documentation	249
10.26.4.1	operator<<	249
10.27	gazebo::common::Color Class Reference	249
10.27.1	Detailed Description	252
10.27.2	Member Typedef Documentation	252
10.27.2.1	ABGR	252
10.27.2.2	ARGB	252
10.27.2.3	BGRA	252
10.27.2.4	RGBA	252
10.27.3	Constructor & Destructor Documentation	252
10.27.3.1	Color	252
10.27.3.2	Color	252

10.27.3.3 Color	252
10.27.3.4 ~Color	253
10.27.4 Member Function Documentation	253
10.27.4.1 GetAsABGR	253
10.27.4.2 GetAsARGB	253
10.27.4.3 GetAsBGRA	253
10.27.4.4 GetAsHSV	253
10.27.4.5 GetAsRGBA	253
10.27.4.6 GetAsYUV	254
10.27.4.7 operator!=	254
10.27.4.8 operator*	254
10.27.4.9 operator*	254
10.27.4.10operator*=	254
10.27.4.11operator+	255
10.27.4.12operator+	255
10.27.4.13operator+=	255
10.27.4.14operator-	255
10.27.4.15operator-	256
10.27.4.16operator-=	256
10.27.4.17operator/	256
10.27.4.18operator/	256
10.27.4.19operator/=	257
10.27.4.20operator=	257
10.27.4.21operator==	257
10.27.4.22operator[]	257
10.27.4.23Reset	258
10.27.4.24Set	258
10.27.4.25SetFromABGR	258
10.27.4.26SetFromARGB	258
10.27.4.27SetFromBGRA	258
10.27.4.28SetFromHSV	259
10.27.4.29SetFromRGBA	259
10.27.4.30SetFromYUV	259
10.27.5 Friends And Related Function Documentation	259
10.27.5.1 operator<<	259
10.27.5.2 operator>>	259
10.27.6 Member Data Documentation	260

10.27.6.1 a	260
10.27.6.2 b	260
10.27.6.3 Black	260
10.27.6.4 Blue	260
10.27.6.5 g	260
10.27.6.6 Green	260
10.27.6.7 Purple	260
10.27.6.8 r	260
10.27.6.9 Red	260
10.27.6.10 White	260
10.27.6.11 Yellow	260
10.28 gazebo::rendering::COMVisual Class Reference	260
10.28.1 Detailed Description	261
10.28.2 Constructor & Destructor Documentation	261
10.28.2.1 COMVisual	261
10.28.2.2 ~COMVisual	262
10.28.3 Member Function Documentation	262
10.28.3.1 Load	262
10.28.3.2 Load	262
10.29 gazebo::rendering::COMVisualPrivate Class Reference	262
10.29.1 Detailed Description	263
10.29.2 Member Data Documentation	263
10.29.2.1 boxNode	263
10.29.2.2 crossLines	263
10.30 gazebo::event::Connection Class Reference	263
10.30.1 Detailed Description	263
10.30.2 Constructor & Destructor Documentation	264
10.30.2.1 Connection	264
10.30.2.2 Connection	264
10.30.2.3 ~Connection	264
10.30.3 Member Function Documentation	264
10.30.3.1 GetId	264
10.31 gazebo::transport::Connection Class Reference	264
10.31.1 Detailed Description	266
10.31.2 Member Typedef Documentation	266
10.31.2.1 AcceptCallback	266
10.31.2.2 ReadCallback	266

10.31.3 Constructor & Destructor Documentation	266
10.31.3.1 Connection	266
10.31.3.2 ~Connection	266
10.31.4 Member Function Documentation	266
10.31.4.1 AsyncRead	266
10.31.4.2 Cancel	267
10.31.4.3 Connect	267
10.31.4.4 ConnectToShutdown	267
10.31.4.5 DisconnectShutdown	267
10.31.4.6 EnqueueMsg	267
10.31.4.7 EnqueueMsg	268
10.31.4.8 GetId	268
10.31.4.9 GetIPWhiteList	268
10.31.4.10GetLocalAddress	268
10.31.4.11GetLocalHostname	268
10.31.4.12GetLocalPort	269
10.31.4.13GetLocalURI	269
10.31.4.14GetRemoteAddress	269
10.31.4.15GetRemoteHostname	269
10.31.4.16GetRemotePort	269
10.31.4.17GetRemoteURI	269
10.31.4.18IsOpen	270
10.31.4.19Listen	270
10.31.4.20ProcessWriteQueue	270
10.31.4.21Read	270
10.31.4.22Shutdown	270
10.31.4.23StartRead	270
10.31.4.24StopRead	270
10.31.4.25ValidateIP	271
10.32gazebo::transport::ConnectionManager Class Reference	271
10.32.1 Detailed Description	272
10.32.2 Member Function Documentation	272
10.32.2.1 Advertise	272
10.32.2.2 ConnectToRemoteHost	273
10.32.2.3 Fini	273
10.32.2.4 GetAllPublishers	273
10.32.2.5 GetTopicNamespaces	273

10.32.2.6	Init	273
10.32.2.7	IsRunning	274
10.32.2.8	RegisterTopicNamespace	274
10.32.2.9	RemoveConnection	274
10.32.2.10	Run	274
10.32.2.11	Stop	274
10.32.2.12	Subscribe	274
10.32.2.13	TriggerUpdate	274
10.32.2.14	Unadvertise	275
10.32.2.15	Unsubscribe	275
10.32.2.16	Unsubscribe	275
10.32.3	Member Data Documentation	275
10.32.3.1	eventConnections	275
10.33	gazebo::event::ConnectionPrivate Class Reference	275
10.33.1	Constructor & Destructor Documentation	276
10.33.1.1	ConnectionPrivate	276
10.33.1.2	ConnectionPrivate	276
10.33.2	Member Data Documentation	276
10.33.2.1	creationTime	276
10.33.2.2	event	276
10.33.2.3	id	276
10.34	gazebo::transport::ConnectionReadTask Class Reference	276
10.34.1	Detailed Description	277
10.34.2	Constructor & Destructor Documentation	277
10.34.2.1	ConnectionReadTask	277
10.34.3	Member Function Documentation	277
10.34.3.1	execute	277
10.35	gazebo::common::Console Class Reference	277
10.35.1	Detailed Description	278
10.35.2	Member Function Documentation	278
10.35.2.1	GetQuiet	278
10.35.2.2	SetQuiet	278
10.35.3	Member Data Documentation	278
10.35.3.1	dbg	278
10.35.3.2	err	278
10.35.3.3	log	278
10.35.3.4	msg	279

10.35.3.5 warn	279
10.36gazebo::physics::Contact Class Reference	279
10.36.1 Detailed Description	280
10.36.2 Constructor & Destructor Documentation	280
10.36.2.1 Contact	280
10.36.2.2 Contact	280
10.36.2.3 ~Contact	280
10.36.3 Member Function Documentation	280
10.36.3.1 DebugString	280
10.36.3.2 FillMsg	280
10.36.3.3 operator=	281
10.36.3.4 operator=	281
10.36.3.5 Reset	281
10.36.4 Member Data Documentation	281
10.36.4.1 collision1	281
10.36.4.2 collision2	281
10.36.4.3 count	281
10.36.4.4 depths	282
10.36.4.5 normals	282
10.36.4.6 positions	282
10.36.4.7 time	282
10.36.4.8 world	282
10.36.4.9 wrench	282
10.37gazebo::physics::ContactManager Class Reference	282
10.37.1 Detailed Description	283
10.37.2 Constructor & Destructor Documentation	283
10.37.2.1 ContactManager	283
10.37.2.2 ~ContactManager	283
10.37.3 Member Function Documentation	283
10.37.3.1 Clear	283
10.37.3.2 CreateFilter	284
10.37.3.3 CreateFilter	284
10.37.3.4 CreateFilter	284
10.37.3.5 GetContact	284
10.37.3.6 GetContactCount	284
10.37.3.7 GetContacts	285
10.37.3.8 GetFilterCount	285

10.37.3.9 HasFilter	285
10.37.3.10Init	285
10.37.3.11NewContact	285
10.37.3.12PublishContacts	285
10.37.3.13RemoveFilter	286
10.37.3.14ResetCount	286
10.38gazebo::rendering::ContactVisualPrivate::ContactPoint Class Reference	286
10.38.1 Detailed Description	286
10.38.2 Member Data Documentation	286
10.38.2.1 depth	286
10.38.2.2 normal	286
10.38.2.3 sceneNode	286
10.39gazebo::physics::ContactPublisher Class Reference	287
10.39.1 Detailed Description	287
10.39.2 Member Data Documentation	287
10.39.2.1 collisionNames	287
10.39.2.2 collisions	287
10.39.2.3 contacts	287
10.39.2.4 publisher	287
10.40gazebo::sensors::ContactSensor Class Reference	287
10.40.1 Detailed Description	289
10.40.2 Constructor & Destructor Documentation	289
10.40.2.1 ContactSensor	289
10.40.2.2 ~ContactSensor	289
10.40.3 Member Function Documentation	289
10.40.3.1 Fini	289
10.40.3.2 GetCollisionContactCount	289
10.40.3.3 GetCollisionCount	289
10.40.3.4 GetCollisionName	289
10.40.3.5 GetContacts	290
10.40.3.6 GetContacts	290
10.40.3.7 Init	291
10.40.3.8 IsActive	291
10.40.3.9 Load	291
10.40.3.10Load	291
10.40.3.11UpdateImpl	291
10.41gazebo::rendering::ContactVisual Class Reference	292

10.41.1 Detailed Description	293
10.41.2 Constructor & Destructor Documentation	293
10.41.2.1 ContactVisual	293
10.41.2.2 ~ContactVisual	293
10.41.3 Member Function Documentation	293
10.41.3.1 SetEnabled	293
10.42gazebo::rendering::ContactVisualPrivate Class Reference	293
10.42.1 Detailed Description	295
10.42.2 Member Data Documentation	295
10.42.2.1 connections	295
10.42.2.2 contactsMsg	295
10.42.2.3 contactsSub	295
10.42.2.4 enabled	295
10.42.2.5 mutex	295
10.42.2.6 node	295
10.42.2.7 points	295
10.42.2.8 receivedMsg	295
10.42.2.9 topicName	295
10.43gazebo::rendering::Conversions Class Reference	296
10.43.1 Detailed Description	296
10.43.2 Member Function Documentation	296
10.43.2.1 Convert	296
10.43.2.2 Convert	296
10.43.2.3 Convert	297
10.43.2.4 Convert	297
10.43.2.5 Convert	297
10.43.2.6 Convert	297
10.44gazebo::physics::CylinderShape Class Reference	298
10.44.1 Detailed Description	299
10.44.2 Constructor & Destructor Documentation	299
10.44.2.1 CylinderShape	299
10.44.2.2 ~CylinderShape	299
10.44.3 Member Function Documentation	299
10.44.3.1 FillMsg	299
10.44.3.2 GetLength	299
10.44.3.3 GetRadius	300
10.44.3.4 Init	300

10.44.3.5 ProcessMsg	300
10.44.3.6 SetLength	300
10.44.3.7 SetRadius	300
10.44.3.8 SetScale	300
10.44.3.9 SetSize	301
10.45gazebo::physics::DARTBallJoint Class Reference	301
10.45.1 Detailed Description	303
10.45.2 Constructor & Destructor Documentation	303
10.45.2.1 DARTBallJoint	303
10.45.2.2 ~DARTBallJoint	303
10.45.3 Member Function Documentation	304
10.45.3.1 GetAnchor	304
10.45.3.2 GetAngleImpl	304
10.45.3.3 GetGlobalAxis	304
10.45.3.4 GetHighStop	304
10.45.3.5 GetLowStop	305
10.45.3.6 GetMaxForce	305
10.45.3.7 GetVelocity	305
10.45.3.8 Init	306
10.45.3.9 Load	306
10.45.3.10SetAxis	306
10.45.3.11SetForceImpl	306
10.45.3.12SetHighStop	307
10.45.3.13SetLowStop	307
10.45.3.14SetMaxForce	307
10.45.3.15SetVelocity	307
10.45.4 Member Data Documentation	307
10.45.4.1 dtBallJoint	308
10.46gazebo::physics::DARTBoxShape Class Reference	308
10.46.1 Detailed Description	309
10.46.2 Constructor & Destructor Documentation	309
10.46.2.1 DARTBoxShape	309
10.46.2.2 ~DARTBoxShape	309
10.46.3 Member Function Documentation	309
10.46.3.1 SetSize	309
10.47gazebo::physics::DARTCollision Class Reference	309
10.47.1 Detailed Description	311

10.47.2 Constructor & Destructor Documentation	311
10.47.2.1 DARTCollision	311
10.47.2.2 ~DARTCollision	311
10.47.3 Member Function Documentation	311
10.47.3.1 Fini	311
10.47.3.2 GetBoundingBox	311
10.47.3.3 GetCategoryBits	311
10.47.3.4 GetCollideBits	312
10.47.3.5 GetDARTBodyNode	312
10.47.3.6 GetDARTCollisionShape	312
10.47.3.7 Init	312
10.47.3.8 Load	312
10.47.3.9 OnPoseChange	312
10.47.3.10SetCategoryBits	313
10.47.3.11SetCollideBits	313
10.47.3.12SetDARTCollisionShape	313
10.48gazebo::physics::DARTCylinderShape Class Reference	313
10.48.1 Detailed Description	314
10.48.2 Constructor & Destructor Documentation	314
10.48.2.1 DARTCylinderShape	314
10.48.2.2 ~DARTCylinderShape	315
10.48.3 Member Function Documentation	315
10.48.3.1 SetSize	315
10.49gazebo::physics::DARTHeightmapShape Class Reference	315
10.49.1 Detailed Description	316
10.49.2 Constructor & Destructor Documentation	316
10.49.2.1 DARTHeightmapShape	316
10.49.2.2 ~DARTHeightmapShape	317
10.49.3 Member Function Documentation	317
10.49.3.1 Init	317
10.50gazebo::physics::DARTHinge2Joint Class Reference	317
10.50.1 Detailed Description	319
10.50.2 Constructor & Destructor Documentation	319
10.50.2.1 DARTHinge2Joint	319
10.50.2.2 ~DARTHinge2Joint	319
10.50.3 Member Function Documentation	319
10.50.3.1 GetAnchor	319

10.50.3.2	GetAngleImpl	320
10.50.3.3	GetGlobalAxis	320
10.50.3.4	GetMaxForce	320
10.50.3.5	GetVelocity	321
10.50.3.6	Init	321
10.50.3.7	Load	321
10.50.3.8	SetAxis	321
10.50.3.9	SetForceImpl	321
10.50.3.10	SetMaxForce	322
10.50.3.11	SetVelocity	322
10.50.4	Member Data Documentation	322
10.50.4.1	dtUniveralJoint	322
10.51	gazebo::physics::DARTHingeJoint Class Reference	322
10.51.1	Detailed Description	324
10.51.2	Constructor & Destructor Documentation	324
10.51.2.1	DARTHingeJoint	324
10.51.2.2	~DARTHingeJoint	324
10.51.3	Member Function Documentation	324
10.51.3.1	GetAnchor	324
10.51.3.2	GetAngleImpl	325
10.51.3.3	GetGlobalAxis	325
10.51.3.4	GetMaxForce	325
10.51.3.5	GetVelocity	326
10.51.3.6	Init	326
10.51.3.7	Load	326
10.51.3.8	SetAxis	326
10.51.3.9	SetForceImpl	326
10.51.3.10	SetMaxForce	327
10.51.3.11	SetVelocity	327
10.51.4	Member Data Documentation	327
10.51.4.1	dtRevoluteJoint	327
10.52	gazebo::physics::DARTJoint Class Reference	327
10.52.1	Detailed Description	330
10.52.2	Constructor & Destructor Documentation	330
10.52.2.1	DARTJoint	330
10.52.2.2	~DARTJoint	330
10.52.3	Member Function Documentation	330

10.52.3.1 ApplyDamping	330
10.52.3.2 AreConnected	330
10.52.3.3 Attach	330
10.52.3.4 Detach	331
10.52.3.5 GetAngleCount	331
10.52.3.6 GetAttribute	331
10.52.3.7 GetDARTJoint	331
10.52.3.8 GetDARTModel	331
10.52.3.9 GetForce	332
10.52.3.10 GetForceTorque	332
10.52.3.11 GetHighStop	332
10.52.3.12 GetJointLink	333
10.52.3.13 GetLinkForce	333
10.52.3.14 GetLinkTorque	333
10.52.3.15 GetLowStop	334
10.52.3.16 GetParam	334
10.52.3.17 Init	334
10.52.3.18 Load	334
10.52.3.19 Reset	335
10.52.3.20 SetAnchor	335
10.52.3.21 SetAttribute	335
10.52.3.22 SetDamping	335
10.52.3.23 SetForce	335
10.52.3.24 SetForceImpl	336
10.52.3.25 SetHighStop	336
10.52.3.26 SetLowStop	336
10.52.3.27 SetParam	337
10.52.3.28 SetStiffness	337
10.52.3.29 SetStiffnessDamping	337
10.52.4 Member Data Documentation	337
10.52.4.1 dartPhysicsEngine	337
10.52.4.2 dtChildBodyNode	337
10.52.4.3 dtJoint	338
10.53 gazebo::physics::DARTLink Class Reference	338
10.53.1 Detailed Description	340
10.53.2 Constructor & Destructor Documentation	340
10.53.2.1 DARTLink	340

10.53.2.2 ~DARTLink	340
10.53.3 Member Function Documentation	341
10.53.3.1 AddDARTChildJoint	341
10.53.3.2 AddForce	341
10.53.3.3 AddForceAtRelativePosition	341
10.53.3.4 AddForceAtWorldPosition	341
10.53.3.5 AddRelativeForce	341
10.53.3.6 AddRelativeTorque	342
10.53.3.7 AddTorque	342
10.53.3.8 Fini	342
10.53.3.9 GetDARTBodyNode	342
10.53.3.10GetDARTModel	342
10.53.3.11GetDARTPhysics	342
10.53.3.12GetDARTWorld	343
10.53.3.13GetEnabled	343
10.53.3.14GetGravityMode	343
10.53.3.15GetKinematic	343
10.53.3.16GetWorldAngularVel	343
10.53.3.17GetWorldCoGLinearVel	344
10.53.3.18GetWorldForce	344
10.53.3.19GetWorldLinearVel	344
10.53.3.20GetWorldLinearVel	344
10.53.3.21GetWorldTorque	345
10.53.3.22Init	345
10.53.3.23Load	345
10.53.3.24OnPoseChange	345
10.53.3.25SetAngularDamping	345
10.53.3.26SetAngularVel	345
10.53.3.27SetAutoDisable	346
10.53.3.28SetDARTParentJoint	346
10.53.3.29SetEnabled	346
10.53.3.30SetForce	346
10.53.3.31SetGravityMode	346
10.53.3.32SetKinematic	346
10.53.3.33SetLinearDamping	347
10.53.3.34SetLinearVel	347
10.53.3.35SetLinkStatic	347

10.53.3.36	SetSelfCollide	347
10.53.3.37	SetTorque	347
10.53.3.38	updateDirtyPoseFromDARTTransformation	348
10.54	gazebo::physics::DARTMeshShape Class Reference	348
10.54.1	Detailed Description	349
10.54.2	Constructor & Destructor Documentation	349
10.54.2.1	DARTMeshShape	349
10.54.2.2	~DARTMeshShape	349
10.54.3	Member Function Documentation	349
10.54.3.1	Init	349
10.54.3.2	Load	349
10.54.3.3	Update	350
10.55	gazebo::physics::DARTModel Class Reference	350
10.55.1	Detailed Description	351
10.55.2	Constructor & Destructor Documentation	351
10.55.2.1	DARTModel	351
10.55.2.2	~DARTModel	351
10.55.3	Member Function Documentation	351
10.55.3.1	BackupState	351
10.55.3.2	Fini	351
10.55.3.3	GetDARTPhysics	351
10.55.3.4	GetDARTSkeleton	352
10.55.3.5	GetDARTWorld	352
10.55.3.6	Init	352
10.55.3.7	Load	352
10.55.3.8	RestoreState	352
10.55.3.9	Update	352
10.55.4	Member Data Documentation	352
10.55.4.1	dtConfig	352
10.55.4.2	dtSkeleton	352
10.55.4.3	dtVelocity	352
10.56	gazebo::physics::DARTMultiRayShape Class Reference	352
10.56.1	Detailed Description	353
10.57	gazebo::physics::DARTPhysics Class Reference	354
10.57.1	Detailed Description	355
10.57.2	Member Enumeration Documentation	356
10.57.2.1	DARTParam	356

10.57.3 Constructor & Destructor Documentation	356
10.57.3.1 DARTPhysics	356
10.57.3.2 ~DARTPhysics	356
10.57.4 Member Function Documentation	356
10.57.4.1 CreateCollision	356
10.57.4.2 CreateJoint	356
10.57.4.3 CreateLink	356
10.57.4.4 CreateModel	357
10.57.4.5 CreateShape	357
10.57.4.6 DebugPrint	357
10.57.4.7 Fini	357
10.57.4.8 GetDARTWorld	357
10.57.4.9 GetParam	357
10.57.4.10GetParam	358
10.57.4.11GetType	358
10.57.4.12Init	358
10.57.4.13InitForThread	358
10.57.4.14Load	359
10.57.4.15OnPhysicsMsg	359
10.57.4.16OnRequest	359
10.57.4.17Reset	359
10.57.4.18SetGravity	359
10.57.4.19SetParam	360
10.57.4.20SetSeed	360
10.57.4.21UpdateCollision	361
10.57.4.22UpdatePhysics	361
10.58gazebo::physics::DARTPlaneShape Class Reference	361
10.58.1 Detailed Description	362
10.58.2 Constructor & Destructor Documentation	362
10.58.2.1 DARTPlaneShape	362
10.58.2.2 ~DARTPlaneShape	362
10.58.3 Member Function Documentation	362
10.58.3.1 CreatePlane	362
10.58.3.2 SetAltitude	362
10.59gazebo::physics::DARTRayShape Class Reference	363
10.59.1 Detailed Description	364
10.60gazebo::physics::DARTScrewJoint Class Reference	364

10.60.1 Detailed Description	366
10.60.2 Constructor & Destructor Documentation	366
10.60.2.1 DARTScrewJoint	366
10.60.2.2 ~DARTScrewJoint	366
10.60.3 Member Function Documentation	366
10.60.3.1 GetAnchor	366
10.60.3.2 GetAngleImpl	366
10.60.3.3 GetGlobalAxis	367
10.60.3.4 GetHighStop	367
10.60.3.5 GetLowStop	367
10.60.3.6 GetMaxForce	367
10.60.3.7 GetThreadPitch	368
10.60.3.8 GetThreadPitch	368
10.60.3.9 GetVelocity	368
10.60.3.10 Init	369
10.60.3.11 Load	369
10.60.3.12 SetAnchor	369
10.60.3.13 SetAxis	369
10.60.3.14 SetForceImpl	369
10.60.3.15 SetMaxForce	370
10.60.3.16 SetThreadPitch	370
10.60.3.17 SetThreadPitch	370
10.60.3.18 SetVelocity	370
10.60.4 Member Data Documentation	371
10.60.4.1 dartScrewJoint	371
10.61 gazebo::physics::DARTSliderJoint Class Reference	371
10.61.1 Detailed Description	372
10.61.2 Constructor & Destructor Documentation	373
10.61.2.1 DARTSliderJoint	373
10.61.2.2 ~DARTSliderJoint	373
10.61.3 Member Function Documentation	373
10.61.3.1 GetAnchor	373
10.61.3.2 GetAngleImpl	373
10.61.3.3 GetGlobalAxis	373
10.61.3.4 GetMaxForce	374
10.61.3.5 GetVelocity	374
10.61.3.6 Init	374

10.61.3.7 Load	374
10.61.3.8 SetAxis	375
10.61.3.9 SetForceImpl	375
10.61.3.10SetMaxForce	375
10.61.3.11SetVelocity	375
10.61.4 Member Data Documentation	375
10.61.4.1 dtPrismaticJoint	376
10.62gazebo::physics::DARTSphereShape Class Reference	376
10.62.1 Detailed Description	377
10.62.2 Constructor & Destructor Documentation	377
10.62.2.1 DARTSphereShape	377
10.62.2.2 ~DARTSphereShape	377
10.62.3 Member Function Documentation	377
10.62.3.1 SetRadius	377
10.63gazebo::physics::DARTTypes Class Reference	377
10.63.1 Detailed Description	378
10.64gazebo::physics::DARTUniversalJoint Class Reference	378
10.64.1 Detailed Description	379
10.64.2 Constructor & Destructor Documentation	380
10.64.2.1 DARTUniversalJoint	380
10.64.2.2 ~DARTUniversalJoint	380
10.64.3 Member Function Documentation	380
10.64.3.1 GetAnchor	380
10.64.3.2 GetAngleImpl	380
10.64.3.3 GetGlobalAxis	380
10.64.3.4 GetMaxForce	381
10.64.3.5 GetVelocity	381
10.64.3.6 Init	381
10.64.3.7 Load	381
10.64.3.8 SetAxis	382
10.64.3.9 SetForceImpl	382
10.64.3.10SetMaxForce	382
10.64.3.11SetVelocity	382
10.64.4 Member Data Documentation	383
10.64.4.1 dtUniveralJoint	383
10.65gazebo::rendering::DepthCamera Class Reference	383
10.65.1 Detailed Description	384

10.65.2 Constructor & Destructor Documentation	384
10.65.2.1 DepthCamera	384
10.65.2.2 ~DepthCamera	385
10.65.3 Member Function Documentation	385
10.65.3.1 ConnectNewDepthFrame	385
10.65.3.2 ConnectNewRGBPointCloud	385
10.65.3.3 CreateDepthTexture	385
10.65.3.4 DisconnectNewDepthFrame	385
10.65.3.5 DisconnectNewRGBPointCloud	386
10.65.3.6 Fini	386
10.65.3.7 GetDepthData	386
10.65.3.8 Init	386
10.65.3.9 Load	386
10.65.3.10 Load	386
10.65.3.11 PostRender	386
10.65.3.12 SetDepthTarget	387
10.65.4 Member Data Documentation	387
10.65.4.1 depthTarget	387
10.65.4.2 depthTexture	387
10.65.4.3 depthViewport	387
10.66 gazebo::sensors::DepthCameraSensor Class Reference	387
10.66.1 Constructor & Destructor Documentation	388
10.66.1.1 DepthCameraSensor	388
10.66.1.2 ~DepthCameraSensor	388
10.66.2 Member Function Documentation	388
10.66.2.1 Fini	388
10.66.2.2 GetDepthCamera	389
10.66.2.3 Init	389
10.66.2.4 Load	389
10.66.2.5 Load	389
10.66.2.6 SaveFrame	389
10.66.2.7 SetActive	390
10.66.2.8 UpdateImpl	390
10.67 gazebo::util::DiagnosticManager Class Reference	390
10.67.1 Detailed Description	391
10.67.2 Member Function Documentation	392
10.67.2.1 GetLabel	392

10.67.2.2	GetLogPath	392
10.67.2.3	GetTime	392
10.67.2.4	GetTime	392
10.67.2.5	GetTimerCount	392
10.67.2.6	Init	393
10.67.2.7	Lap	393
10.67.2.8	StartTimer	393
10.67.2.9	StopTimer	393
10.68	gazebo::util::DiagnosticTimer Class Reference	394
10.68.1	Detailed Description	394
10.68.2	Constructor & Destructor Documentation	394
10.68.2.1	DiagnosticTimer	394
10.68.2.2	~DiagnosticTimer	395
10.68.3	Member Function Documentation	395
10.68.3.1	GetName	395
10.68.3.2	Lap	395
10.68.3.3	Start	395
10.68.3.4	Stop	395
10.69	gazebo::rendering::DummyPageProvider Class Reference	395
10.69.1	Detailed Description	396
10.69.2	Member Function Documentation	396
10.69.2.1	loadProceduralPage	396
10.69.2.2	prepareProceduralPage	396
10.69.2.3	unloadProceduralPage	396
10.69.2.4	unprepareProceduralPage	396
10.70	gazebo::rendering::DynamicLines Class Reference	397
10.70.1	Detailed Description	398
10.70.2	Constructor & Destructor Documentation	398
10.70.2.1	DynamicLines	398
10.70.2.2	~DynamicLines	398
10.70.3	Member Function Documentation	398
10.70.3.1	AddPoint	398
10.70.3.2	AddPoint	398
10.70.3.3	Clear	399
10.70.3.4	GetMovableType	399
10.70.3.5	getMovableType	399
10.70.3.6	GetPoint	399

10.70.3.7	GetPointCount	399
10.70.3.8	SetColor	399
10.70.3.9	SetPoint	400
10.70.3.10	Update	400
10.71	gazebo::rendering::DynamicRenderable Class Reference	400
10.71.1	Detailed Description	401
10.71.2	Constructor & Destructor Documentation	401
10.71.2.1	DynamicRenderable	401
10.71.2.2	~DynamicRenderable	401
10.71.3	Member Function Documentation	402
10.71.3.1	CreateVertexDeclaration	402
10.71.3.2	FillHardwareBuffers	402
10.71.3.3	getBoundingRadius	402
10.71.3.4	GetMovableType	402
10.71.3.5	GetOperationType	402
10.71.3.6	getSquaredViewDepth	403
10.71.3.7	Init	403
10.71.3.8	PrepareHardwareBuffers	403
10.71.3.9	SetOperationType	404
10.71.4	Member Data Documentation	404
10.71.4.1	indexBufferCapacity	404
10.71.4.2	vertexBufferCapacity	404
10.72	gazebo::physics::Entity Class Reference	404
10.72.1	Detailed Description	407
10.72.2	Constructor & Destructor Documentation	407
10.72.2.1	Entity	407
10.72.2.2	~Entity	407
10.72.3	Member Function Documentation	407
10.72.3.1	Fini	407
10.72.3.2	GetBoundingBox	407
10.72.3.3	GetChildCollision	408
10.72.3.4	GetChildLink	408
10.72.3.5	GetCollisionBoundingBox	408
10.72.3.6	GetDirtyPose	408
10.72.3.7	GetInitialRelativePose	409
10.72.3.8	GetNearestEntityBelow	409
10.72.3.9	GetParentModel	409

10.72.3.10	GetRelativeAngularAccel	409
10.72.3.11	GetRelativeAngularVel	409
10.72.3.12	GetRelativeLinearAccel	409
10.72.3.13	GetRelativeLinearVel	410
10.72.3.14	GetRelativePose	410
10.72.3.15	GetWorldAngularAccel	410
10.72.3.16	GetWorldAngularVel	410
10.72.3.17	GetWorldLinearAccel	411
10.72.3.18	GetWorldLinearVel	411
10.72.3.19	GetWorldPose	411
10.72.3.20	IsCanonicalLink	411
10.72.3.21	IsStatic	411
10.72.3.22	Load	411
10.72.3.23	OnPoseChange	412
10.72.3.24	PlaceOnEntity	412
10.72.3.25	PlaceOnNearestEntityBelow	412
10.72.3.26	Reset	412
10.72.3.27	SetAnimation	412
10.72.3.28	SetAnimation	412
10.72.3.29	SetCanonicalLink	413
10.72.3.30	SetInitialRelativePose	413
10.72.3.31	SetName	413
10.72.3.32	SetRelativePose	413
10.72.3.33	SetStatic	413
10.72.3.34	SetWorldPose	414
10.72.3.35	SetWorldTwist	414
10.72.3.36	StopAnimation	414
10.72.3.37	UpdateParameters	414
10.72.4	Member Data Documentation	414
10.72.4.1	animation	414
10.72.4.2	animationConnection	414
10.72.4.3	animationStartPose	415
10.72.4.4	connections	415
10.72.4.5	dirtyPose	415
10.72.4.6	node	415
10.72.4.7	parentEntity	415
10.72.4.8	prevAnimationTime	415

10.72.4.9 requestPub	415
10.72.4.10scale	415
10.72.4.11visPub	415
10.72.4.12visualMsg	415
10.73gazebo::event::Event Class Reference	416
10.73.1 Detailed Description	417
10.73.2 Constructor & Destructor Documentation	417
10.73.2.1 Event	417
10.73.2.2 ~Event	417
10.73.2.3 Event	417
10.73.3 Member Function Documentation	417
10.73.3.1 Disconnect	418
10.73.3.2 Disconnect	418
10.73.3.3 GetSignaled	418
10.73.4 Member Data Documentation	418
10.73.4.1 dataPtr	418
10.74gazebo::event::EventPrivate Class Reference	419
10.74.1 Constructor & Destructor Documentation	421
10.74.1.1 EventPrivate	421
10.74.2 Member Data Documentation	421
10.74.2.1 signaled	421
10.75gazebo::event::Events Class Reference	421
10.75.1 Detailed Description	424
10.75.2 Member Function Documentation	424
10.75.2.1 ConnectAddEntity	424
10.75.2.2 ConnectCreateEntity	424
10.75.2.3 ConnectDeleteEntity	424
10.75.2.4 ConnectDiagTimerStart	424
10.75.2.5 ConnectDiagTimerStop	425
10.75.2.6 ConnectPause	425
10.75.2.7 ConnectPostRender	425
10.75.2.8 ConnectPreRender	426
10.75.2.9 ConnectRender	426
10.75.2.10ConnectSetSelectedEntity	426
10.75.2.11ConnectSigInt	426
10.75.2.12ConnectStep	427
10.75.2.13ConnectStop	427

10.75.2.14	ConnectWorldCreated	427
10.75.2.15	ConnectWorldUpdateBegin	428
10.75.2.16	ConnectWorldUpdateEnd	428
10.75.2.17	DisconnectAddEntity	428
10.75.2.18	DisconnectCreateEntity	428
10.75.2.19	DisconnectDeleteEntity	428
10.75.2.20	DisconnectDiagTimerStart	429
10.75.2.21	DisconnectDiagTimerStop	429
10.75.2.22	DisconnectPause	429
10.75.2.23	DisconnectPostRender	429
10.75.2.24	DisconnectPreRender	429
10.75.2.25	DisconnectRender	430
10.75.2.26	DisconnectSetSelectedEntity	430
10.75.2.27	DisconnectSigInt	430
10.75.2.28	DisconnectStep	430
10.75.2.29	DisconnectStop	430
10.75.2.30	DisconnectWorldCreated	430
10.75.2.31	DisconnectWorldUpdateBegin	431
10.75.2.32	DisconnectWorldUpdateEnd	431
10.75.3	Member Data Documentation	431
10.75.3.1	addEntity	431
10.75.3.2	deleteEntity	431
10.75.3.3	diagTimerStart	431
10.75.3.4	diagTimerStop	431
10.75.3.5	entityCreated	431
10.75.3.6	pause	431
10.75.3.7	postRender	431
10.75.3.8	preRender	432
10.75.3.9	render	432
10.75.3.10	setSelectedEntity	432
10.75.3.11	sigInt	432
10.75.3.12	step	432
10.75.3.13	stop	432
10.75.3.14	worldCreated	432
10.75.3.15	worldUpdateBegin	432
10.75.3.16	worldUpdateEnd	432
10.76	gazebo::rendering::Events Class Reference	432

10.76.1 Detailed Description	433
10.76.2 Member Function Documentation	433
10.76.2.1 ConnectCreateScene	433
10.76.2.2 ConnectRemoveScene	433
10.76.2.3 DisconnectCreateScene	434
10.76.2.4 DisconnectRemoveScene	434
10.76.3 Member Data Documentation	434
10.76.3.1 createScene	434
10.76.3.2 removeScene	434
10.77 gazebo::event::EventT < T > Class Template Reference	434
10.77.1 Detailed Description	437
10.77.2 Member Function Documentation	437
10.77.2.1 operator()	437
10.77.2.2 operator()	437
10.77.2.3 operator()	437
10.77.2.4 operator()	437
10.77.2.5 operator()	438
10.77.2.6 operator()	438
10.77.2.7 operator()	438
10.77.2.8 operator()	439
10.77.2.9 operator()	439
10.77.2.10 operator()	439
10.77.2.11 operator()	440
10.77.2.12 Signal	440
10.77.2.13 Signal	440
10.77.2.14 Signal	440
10.77.2.15 Signal	440
10.77.2.16 Signal	441
10.77.2.17 Signal	441
10.77.2.18 Signal	441
10.77.2.19 Signal	442
10.77.2.20 Signal	442
10.77.2.21 Signal	442
10.77.2.22 Signal	443
10.78 gazebo::event::EventTPrivate < T > Class Template Reference	443
10.78.1 Member Data Documentation	444
10.78.1.1 connections	444

10.78.1.2 connectionsEraseMutex	444
10.78.1.3 connectionsToErase	444
10.79 gazebo::common::Exception Class Reference	444
10.79.1 Detailed Description	445
10.79.2 Constructor & Destructor Documentation	446
10.79.2.1 Exception	446
10.79.2.2 Exception	446
10.79.2.3 ~Exception	446
10.79.3 Member Function Documentation	446
10.79.3.1 GetErrorFile	446
10.79.3.2 GetErrorStr	446
10.79.3.3 Print	446
10.79.4 Friends And Related Function Documentation	446
10.79.4.1 operator<<	446
10.80 gazebo::common::FileLogger Class Reference	447
10.80.1 Detailed Description	448
10.80.2 Constructor & Destructor Documentation	448
10.80.2.1 FileLogger	448
10.80.2.2 ~FileLogger	448
10.80.3 Member Function Documentation	448
10.80.3.1 Init	448
10.80.3.2 operator()	448
10.80.3.3 operator()	449
10.81 gazebo::sensors::ForceTorqueSensor Class Reference	449
10.81.1 Detailed Description	450
10.81.2 Constructor & Destructor Documentation	450
10.81.2.1 ForceTorqueSensor	450
10.81.2.2 ~ForceTorqueSensor	450
10.81.3 Member Function Documentation	451
10.81.3.1 ConnectUpdate	451
10.81.3.2 DisconnectUpdate	451
10.81.3.3 Fini	451
10.81.3.4 GetForce	451
10.81.3.5 GetJoint	451
10.81.3.6 GetTopic	451
10.81.3.7 GetTorque	452
10.81.3.8 Init	452

10.81.3.9	IsActive	452
10.81.3.10	Load	452
10.81.3.11	UpdateImpl	452
10.81.4	Member Data Documentation	453
10.81.4.1	update	453
10.82	gazebo::rendering::FPSViewController Class Reference	453
10.82.1	Detailed Description	454
10.82.2	Constructor & Destructor Documentation	454
10.82.2.1	FPSViewController	454
10.82.2.2	~FPSViewController	454
10.82.3	Member Function Documentation	454
10.82.3.1	GetTypeString	454
10.82.3.2	HandleKeyPressEvent	454
10.82.3.3	HandleKeyReleaseEvent	455
10.82.3.4	HandleMouseEvent	455
10.82.3.5	Init	455
10.82.3.6	Update	455
10.83	gazebo::physics::FrictionPyramid Class Reference	455
10.83.1	Detailed Description	456
10.83.2	Constructor & Destructor Documentation	456
10.83.2.1	FrictionPyramid	456
10.83.2.2	~FrictionPyramid	456
10.83.3	Member Function Documentation	456
10.83.3.1	GetMuPrimary	456
10.83.3.2	GetMuSecondary	456
10.83.3.3	SetMuPrimary	457
10.83.3.4	SetMuSecondary	457
10.83.4	Member Data Documentation	457
10.83.4.1	direction1	457
10.84	gazebo::sensors::GaussianNoiseModel Class Reference	457
10.84.1	Detailed Description	459
10.84.2	Constructor & Destructor Documentation	459
10.84.2.1	GaussianNoiseModel	459
10.84.2.2	~GaussianNoiseModel	459
10.84.3	Member Function Documentation	459
10.84.3.1	ApplyImpl	459
10.84.3.2	Fini	459

10.84.3.3	GetBias	460
10.84.3.4	GetMean	460
10.84.3.5	GetStdDev	460
10.84.3.6	Load	460
10.84.4	Member Data Documentation	460
10.84.4.1	bias	460
10.84.4.2	mean	460
10.84.4.3	precision	461
10.84.4.4	quantized	461
10.84.4.5	stdDev	461
10.85	google::protobuf::compiler::cpp::GazeboGenerator Class Reference	461
10.85.1	Detailed Description	461
10.85.2	Constructor & Destructor Documentation	461
10.85.2.1	GazeboGenerator	461
10.85.2.2	~GazeboGenerator	461
10.85.3	Member Function Documentation	461
10.85.3.1	Generate	461
10.86	gazebo::physics::GearboxJoint< T > Class Template Reference	462
10.86.1	Detailed Description	462
10.86.2	Constructor & Destructor Documentation	462
10.86.2.1	GearboxJoint	462
10.86.2.2	~GearboxJoint	463
10.86.3	Member Function Documentation	463
10.86.3.1	GetAngleCount	463
10.86.3.2	GetGearboxRatio	463
10.86.3.3	Init	463
10.86.3.4	Load	463
10.86.3.5	SetGearboxRatio	463
10.86.4	Member Data Documentation	464
10.86.4.1	gearRatio	464
10.86.4.2	referenceBody	464
10.87	gazebo::sensors::GpsSensor Class Reference	464
10.87.1	Detailed Description	465
10.87.2	Constructor & Destructor Documentation	465
10.87.2.1	GpsSensor	465
10.87.2.2	~GpsSensor	465
10.87.3	Member Function Documentation	465

10.87.3.1	Fini	465
10.87.3.2	GetAltitude	465
10.87.3.3	GetLatitude	466
10.87.3.4	GetLongitude	466
10.87.3.5	Init	466
10.87.3.6	Load	466
10.87.3.7	Load	466
10.87.3.8	UpdateImpl	466
10.88	gazebo::rendering::GpuLaser Class Reference	467
10.88.1	Detailed Description	469
10.88.2	Constructor & Destructor Documentation	470
10.88.2.1	GpuLaser	470
10.88.2.2	~GpuLaser	470
10.88.3	Member Function Documentation	470
10.88.3.1	ConnectNewLaserFrame	470
10.88.3.2	CreateLaserTexture	470
10.88.3.3	DisconnectNewLaserFrame	470
10.88.3.4	Fini	471
10.88.3.5	GetCameraCount	471
10.88.3.6	GetCosHorzFOV	471
10.88.3.7	GetCosVertFOV	471
10.88.3.8	GetFarClip	471
10.88.3.9	GetHorzFOV	471
10.88.3.10	GetHorzHalfAngle	472
10.88.3.11	GetLaserData	472
10.88.3.12	GetNearClip	472
10.88.3.13	GetRayCountRatio	472
10.88.3.14	GetVertFOV	472
10.88.3.15	GetVertHalfAngle	472
10.88.3.16	Init	473
10.88.3.17	IsHorizontal	473
10.88.3.18	Load	473
10.88.3.19	Load	473
10.88.3.20	NotifyRenderSingleObject	473
10.88.3.21	PostRender	473
10.88.3.22	SetCameraCount	473
10.88.3.23	SetCosHorzFOV	474

10.88.3.24SetCosVertFOV	474
10.88.3.25SetFarClip	474
10.88.3.26SetHorzFOV	474
10.88.3.27SetHorzHalfAngle	474
10.88.3.28SetIsHorizontal	474
10.88.3.29SetNearClip	475
10.88.3.30SetRangeCount	475
10.88.3.31SetRayCountRatio	475
10.88.3.32SetVertFOV	475
10.88.3.33SetVertHalfAngle	475
10.88.4 Member Data Documentation	475
10.88.4.1 cameraCount	475
10.88.4.2 chfov	476
10.88.4.3 cvfov	476
10.88.4.4 far	476
10.88.4.5 hfov	476
10.88.4.6 horzHalfAngle	476
10.88.4.7 isHorizontal	476
10.88.4.8 near	476
10.88.4.9 rayCountRatio	476
10.88.4.10vertHalfAngle	476
10.88.4.11vfov	476
10.89gazebo::sensors::GpuRaySensor Class Reference	477
10.89.1 Constructor & Destructor Documentation	479
10.89.1.1 GpuRaySensor	479
10.89.1.2 ~GpuRaySensor	480
10.89.2 Member Function Documentation	480
10.89.2.1 ConnectNewLaserFrame	480
10.89.2.2 DisconnectNewLaserFrame	480
10.89.2.3 Fini	480
10.89.2.4 GetAngleMax	480
10.89.2.5 GetAngleMin	480
10.89.2.6 GetAngleResolution	480
10.89.2.7 GetCameraCount	481
10.89.2.8 GetCosHorzFOV	481
10.89.2.9 GetCosVertFOV	481
10.89.2.10GetFiducial	481

10.89.2.11	GetHorzFOV	481
10.89.2.12	GetHorzHalfAngle	482
10.89.2.13	GetLaserCamera	482
10.89.2.14	GetRange	482
10.89.2.15	GetRangeCount	482
10.89.2.16	GetRangeCountRatio	482
10.89.2.17	GetRangeMax	483
10.89.2.18	GetRangeMin	483
10.89.2.19	GetRangeResolution	483
10.89.2.20	GetRanges	483
10.89.2.21	GetRayCount	483
10.89.2.22	GetRayCountRatio	483
10.89.2.23	GetRetro	484
10.89.2.24	GetTopic	484
10.89.2.25	GetVertFOV	484
10.89.2.26	GetVertHalfAngle	484
10.89.2.27	GetVerticalAngleMax	484
10.89.2.28	GetVerticalAngleMin	485
10.89.2.29	GetVerticalAngleResolution	485
10.89.2.30	GetVerticalRangeCount	485
10.89.2.31	GetVerticalRayCount	485
10.89.2.32	Init	485
10.89.2.33	IsActive	485
10.89.2.34	IsHorizontal	486
10.89.2.35	Load	486
10.89.2.36	Load	486
10.89.2.37	SetAngleMax	486
10.89.2.38	SetAngleMin	486
10.89.2.39	SetVerticalAngleMax	486
10.89.2.40	SetVerticalAngleMin	487
10.89.2.41	UpdateImpl	487
10.89.3	Member Data Documentation	487
10.89.3.1	cameraElem	487
10.89.3.2	horzElem	487
10.89.3.3	horzRangeCount	487
10.89.3.4	horzRayCount	487
10.89.3.5	rangeCountRatio	488

10.89.3.6 rangeElem	488
10.89.3.7 scanElem	488
10.89.3.8 vertElem	488
10.89.3.9 vertRangeCount	488
10.89.3.10vertRayCount	488
10.90gazebo::rendering::Grid Class Reference	488
10.90.1 Detailed Description	489
10.90.2 Constructor & Destructor Documentation	489
10.90.2.1 Grid	489
10.90.2.2 ~Grid	489
10.90.3 Member Function Documentation	490
10.90.3.1 Enable	490
10.90.3.2 GetCellCount	490
10.90.3.3 GetCellLength	490
10.90.3.4 GetColor	490
10.90.3.5 GetHeight	490
10.90.3.6 GetLineWidth	490
10.90.3.7 GetSceneNode	491
10.90.3.8 Init	491
10.90.3.9 SetCellCount	491
10.90.3.10SetCellLength	491
10.90.3.11SetColor	491
10.90.3.12SetHeight	491
10.90.3.13SetLineWidth	491
10.90.3.14SetUserData	492
10.91gazebo::physics::Gripper Class Reference	492
10.91.1 Detailed Description	492
10.91.2 Constructor & Destructor Documentation	493
10.91.2.1 Gripper	493
10.91.2.2 ~Gripper	493
10.91.3 Member Function Documentation	493
10.91.3.1 GetName	493
10.91.3.2 Init	493
10.91.3.3 IsAttached	493
10.91.3.4 Load	493
10.91.4 Member Data Documentation	493
10.91.4.1 node	493

10.92gazebo::rendering::GUIOverlay Class Reference	494
10.92.1 Detailed Description	495
10.92.2 Constructor & Destructor Documentation	495
10.92.2.1 GUIOverlay	495
10.92.2.2 ~GUIOverlay	495
10.92.3 Member Function Documentation	495
10.92.3.1 AttachCameraToImage	495
10.92.3.2 AttachCameraToImage	495
10.92.3.3 ButtonCallback	495
10.92.3.4 CreateWindow	496
10.92.3.5 HandleKeyPressEvent	496
10.92.3.6 HandleKeyReleaseEvent	496
10.92.3.7 HandleMouseEvent	496
10.92.3.8 Hide	497
10.92.3.9 Init	497
10.92.3.10IsInitialized	497
10.92.3.11LoadLayout	497
10.92.3.12Resize	497
10.92.3.13Show	497
10.92.3.14Update	497
10.92.4 Member Data Documentation	498
10.92.4.1 callbacks	498
10.93gazebo::rendering::GUIOverlayPrivate Class Reference	498
10.93.1 Detailed Description	498
10.93.2 Member Data Documentation	498
10.93.2.1 connections	498
10.93.2.2 initialized	498
10.93.2.3 layoutFilename	498
10.93.2.4 rttImageSetCount	499
10.94gazebo::rendering::GzTerrainMatGen Class Reference	499
10.94.1 Constructor & Destructor Documentation	499
10.94.1.1 GzTerrainMatGen	499
10.94.1.2 ~GzTerrainMatGen	499
10.95gazebo::rendering::Heightmap Class Reference	499
10.95.1 Detailed Description	500
10.95.2 Constructor & Destructor Documentation	501
10.95.2.1 Heightmap	501

10.95.2.2	~Heightmap	501
10.95.3	Member Function Documentation	501
10.95.3.1	Flatten	501
10.95.3.2	GetAvgHeight	501
10.95.3.3	GetHeight	501
10.95.3.4	GetImage	502
10.95.3.5	GetMouseHit	502
10.95.3.6	GetOgreTerrain	502
10.95.3.7	GetTerrainSubdivisionCount	502
10.95.3.8	Load	502
10.95.3.9	LoadFromMsg	503
10.95.3.10	Lower	503
10.95.3.11	Raise	503
10.95.3.12	SetWireframe	503
10.95.3.13	Smooth	504
10.95.3.14	SplitHeights	504
10.96	gazebo::common::HeightmapData Class Reference	504
10.96.1	Detailed Description	505
10.96.2	Constructor & Destructor Documentation	505
10.96.2.1	~HeightmapData	505
10.96.3	Member Function Documentation	505
10.96.3.1	FillHeightMap	505
10.96.3.2	GetHeight	506
10.96.3.3	GetMaxElevation	506
10.96.3.4	GetWidth	506
10.97	gazebo::physics::HeightmapShape Class Reference	506
10.97.1	Detailed Description	508
10.97.2	Constructor & Destructor Documentation	508
10.97.2.1	HeightmapShape	508
10.97.2.2	~HeightmapShape	508
10.97.3	Member Function Documentation	509
10.97.3.1	FillMsg	509
10.97.3.2	GetHeight	509
10.97.3.3	GetImage	509
10.97.3.4	GetMaxHeight	509
10.97.3.5	GetMinHeight	509
10.97.3.6	GetPos	510

10.97.3.7	GetSize	510
10.97.3.8	GetSubSampling	510
10.97.3.9	GetURI	510
10.97.3.10	GetVertexCount	510
10.97.3.11	Init	510
10.97.3.12	Load	511
10.97.3.13	ProcessMsg	511
10.97.3.14	SetScale	511
10.97.4	Member Data Documentation	511
10.97.4.1	flipY	511
10.97.4.2	heightmapData	511
10.97.4.3	heights	511
10.97.4.4	img	511
10.97.4.5	subSampling	512
10.97.4.6	vertSize	512
10.98	gazebo::physics::Hinge2Joint< T > Class Template Reference	512
10.98.1	Detailed Description	512
10.98.2	Constructor & Destructor Documentation	512
10.98.2.1	Hinge2Joint	512
10.98.2.2	~Hinge2Joint	513
10.98.3	Member Function Documentation	513
10.98.3.1	GetAngleCount	513
10.98.3.2	Load	513
10.99	gazebo::physics::HingeJoint< T > Class Template Reference	513
10.99.1	Detailed Description	514
10.99.2	Constructor & Destructor Documentation	514
10.99.2.1	HingeJoint	514
10.99.2.2	~HingeJoint	514
10.99.3	Member Function Documentation	514
10.99.3.1	GetAngleCount	514
10.99.3.2	Init	514
10.99.3.3	Load	514
10.100	gazebo::common::Image Class Reference	515
10.100.1	Detailed Description	516
10.100.2	Member Enumeration Documentation	516
10.100.2.1	PixelFormat	516
10.100.3	Constructor & Destructor Documentation	517

10.100.3.1	Image	517
10.100.3.2	~Image	517
10.100.4	Member Function Documentation	517
10.100.4.1	ConvertPixelFormat	517
10.100.4.2	GetAvgColor	517
10.100.4.3	GetBPP	517
10.100.4.4	GetData	518
10.100.4.5	GetFilename	518
10.100.4.6	GetHeight	518
10.100.4.7	GetMaxColor	518
10.100.4.8	GetPitch	518
10.100.4.9	GetPixel	518
10.100.4.10	GetPixelFormat	519
10.100.4.11	GetRGBData	519
10.100.4.12	GetWidth	519
10.100.4.13	Load	519
10.100.4.14	Rescale	519
10.100.4.15	SavePNG	520
10.100.4.16	SetFromData	520
10.100.4.17	Valid	520
10.100	gazebo::sensors::ImageGaussianNoiseModel Class Reference	520
10.101.1	Constructor & Destructor Documentation	521
10.101.1.1	ImageGaussianNoiseModel	522
10.101.1.2	~ImageGaussianNoiseModel	522
10.101.2	Member Function Documentation	522
10.101.2.1	Fini	522
10.101.2.2	Load	522
10.101.2.3	SetCamera	522
10.101.3	Member Data Documentation	522
10.101.3.1	gaussianNoiseCompositorListener	522
10.101.3.2	gaussianNoiseInstance	523
10.102	gazebo::common::ImageHeightmap Class Reference	523
10.102.1	Detailed Description	524
10.102.2	Constructor & Destructor Documentation	524
10.102.2.1	ImageHeightmap	524
10.102.3	Member Function Documentation	524
10.102.3.1	FillHeightMap	524

10.102.3.2	GetFilename	524
10.102.3.3	GetHeight	524
10.102.3.4	GetMaxElevation	525
10.102.3.5	GetWidth	525
10.102.3.6	Load	525
10.103	gazebo::sensors::ImuSensor Class Reference	525
10.103.1	Detailed Description	527
10.103.2	Constructor & Destructor Documentation	527
10.103.2.1	ImuSensor	527
10.103.2.2	~ImuSensor	527
10.103.3	Member Function Documentation	527
10.103.3.1	Fini	527
10.103.3.2	GetAngularVelocity	527
10.103.3.3	GetImuMessage	527
10.103.3.4	GetLinearAcceleration	527
10.103.3.5	GetOrientation	528
10.103.3.6	Init	528
10.103.3.7	IsActive	528
10.103.3.8	Load	528
10.103.3.9	Load	528
10.103.3.10	SetReferencePose	528
10.103.3.11	UpdateImpl	529
10.104	gazebo::physics::Inertial Class Reference	529
10.104.1	Detailed Description	531
10.104.2	Constructor & Destructor Documentation	531
10.104.2.1	Inertial	531
10.104.2.2	Inertial	531
10.104.2.3	Inertial	531
10.104.2.4	~Inertial	532
10.104.3	Member Function Documentation	532
10.104.3.1	GetCoG	532
10.104.3.2	GetInertial	532
10.104.3.3	GetIXX	532
10.104.3.4	GetIXY	532
10.104.3.5	GetIXZ	532
10.104.3.6	GetIYY	533
10.104.3.7	GetIYZ	533

10.104.3.8	GetIZZ	533
10.104.3.9	GetMass	533
10.104.3.10	GetMOI	533
10.104.3.11	GetMOI	533
10.104.3.12	GetPose	534
10.104.3.13	GetPrincipalMoments	534
10.104.3.14	GetProductsofInertia	534
10.104.3.15	Load	534
10.104.3.16	operator+	534
10.104.3.17	operator+=	535
10.104.3.18	operator=	535
10.104.3.19	ProcessMsg	535
10.104.3.20	Reset	535
10.104.3.21	Rotate	535
10.104.3.22	SetCoG	536
10.104.3.23	SetCoG	536
10.104.3.24	SetCoG	536
10.104.3.25	SetCoG	536
10.104.3.26	SetInertiaMatrix	536
10.104.3.27	SetIXX	537
10.104.3.28	SetIXY	537
10.104.3.29	SetIXZ	537
10.104.3.30	SetIYY	537
10.104.3.31	SetIYZ	537
10.104.3.32	SetIZZ	537
10.104.3.33	SetMass	538
10.104.3.34	SetMOI	538
10.104.3.35	UpdateParameters	538
10.104.4	Friends And Related Function Documentation	538
10.104.4.1	operator<<	538
10.105	gazebo::common::InternalError Class Reference	538
10.105.1	Detailed Description	539
10.105.2	Constructor & Destructor Documentation	539
10.105.2.1	InternalError	539
10.105.2.2	InternalError	539
10.105.2.3	~InternalError	540
10.106	gazebo::transport::IOManager Class Reference	540

10.106.1	Detailed Description	540
10.106.2	Constructor & Destructor Documentation	540
10.106.2.1	IOManager	540
10.106.2.2	~IOManager	541
10.106.3	Member Function Documentation	541
10.106.3.1	DecCount	541
10.106.3.2	GetCount	541
10.106.3.3	GetIO	541
10.106.3.4	IncCount	541
10.106.3.5	Stop	541
10.107	gazebo::physics::Joint Class Reference	541
10.107.1	Detailed Description	547
10.107.2	Member Enumeration Documentation	547
10.107.2.1	Attribute	547
10.107.3	Constructor & Destructor Documentation	547
10.107.3.1	Joint	547
10.107.3.2	~Joint	547
10.107.4	Member Function Documentation	547
10.107.4.1	ApplyDamping	547
10.107.4.2	ApplyStiffnessDamping	548
10.107.4.3	AreConnected	548
10.107.4.4	Attach	548
10.107.4.5	CacheForceTorque	548
10.107.4.6	CheckAndTruncateForce	548
10.107.4.7	ConnectJointUpdate	549
10.107.4.8	Detach	549
10.107.4.9	DisconnectJointUpdate	549
10.107.4.10	FillMsg	549
10.107.4.11	Finis	549
10.107.4.12	GetAnchor	549
10.107.4.13	GetAnchorErrorPose	550
10.107.4.14	GetAngle	550
10.107.4.15	GetAngleCount	550
10.107.4.16	GetAngleImpl	550
10.107.4.17	GetAttribute	551
10.107.4.18	GetAxisFrame	551
10.107.4.19	GetChild	551

10.107.4.20	GetDamping	552
10.107.4.21	GetDampingCoefficient	552
10.107.4.22	GetEffortLimit	552
10.107.4.23	GetForce	552
10.107.4.24	GetForceTorque	553
10.107.4.25	GetGlobalAxis	553
10.107.4.26	GetHighStop	553
10.107.4.27	GetInertiaRatio	554
10.107.4.28	GetInertiaRatio	554
10.107.4.29	GetInitialAnchorPose	554
10.107.4.30	GetJointLink	554
10.107.4.31	GetLinkForce	555
10.107.4.32	GetLinkTorque	555
10.107.4.33	GetLocalAxis	555
10.107.4.34	GetLowerLimit	556
10.107.4.35	GetLowStop	556
10.107.4.36	GetMaxForce	556
10.107.4.37	GetParam	557
10.107.4.38	GetParent	557
10.107.4.39	GetParentWorldPose	557
10.107.4.40	GetSpringReferencePosition	557
10.107.4.41	GetStiffness	557
10.107.4.42	GetStopDissipation	558
10.107.4.43	GetStopStiffness	558
10.107.4.44	GetUpperLimit	558
10.107.4.45	GetVelocity	558
10.107.4.46	GetVelocityLimit	559
10.107.4.47	GetWorldEnergyPotentialSpring	559
10.107.4.48	GetWorldPose	559
10.107.4.49	Hit	559
10.107.4.50	Load	560
10.107.4.51	load	560
10.107.4.52	Reset	560
10.107.4.53	SetAnchor	561
10.107.4.54	SetAngle	561
10.107.4.55	SetAttribute	561
10.107.4.56	SetAxis	561

10.107.4.55	SetDamping	562
10.107.4.58	SetEffortLimit	562
10.107.4.59	SetForce	562
10.107.4.60	SetHighStop	562
10.107.4.63	SetLowerLimit	563
10.107.4.62	SetLowStop	563
10.107.4.65	SetMaxForce	563
10.107.4.69	SetModel	563
10.107.4.65	SetParam	564
10.107.4.66	SetProvideFeedback	564
10.107.4.67	SetState	564
10.107.4.68	SetStiffness	564
10.107.4.69	SetStiffnessDamping	565
10.107.4.70	SetStopDissipation	565
10.107.4.73	SetStopStiffness	565
10.107.4.72	SetUpperLimit	565
10.107.4.73	SetVelocity	565
10.107.4.74	Update	566
10.107.4.75	UpdateParameters	566
10.107.5	Member Data Documentation	566
10.107.5.1	anchorLink	566
10.107.5.2	anchorPos	566
10.107.5.3	anchorPose	566
10.107.5.4	applyDamping	566
10.107.5.5	axisParentModelFrame	566
10.107.5.6	childLink	567
10.107.5.7	dampingCoefficient	567
10.107.5.8	dissipationCoefficient	567
10.107.5.9	effortLimit	567
10.107.5.10	lowerLimit	567
10.107.5.11	model	567
10.107.5.12	parentAnchorPose	567
10.107.5.13	parentLink	567
10.107.5.14	provideFeedback	567
10.107.5.15	springReferencePosition	567
10.107.5.16	stiffnessCoefficient	567
10.107.5.17	upperLimit	568

10.107.5.10	UseCFMDamping	568
10.107.5.10	VelocityLimit	568
10.107.5.20	Trench	568
10.108	gazebo::physics::JointController Class Reference	568
10.108.1	Detailed Description	569
10.108.2	Constructor & Destructor Documentation	569
10.108.2.1	JointController	569
10.108.2.2	~JointController	569
10.108.3	Member Function Documentation	569
10.108.3.1	AddJoint	569
10.108.3.2	GetForces	570
10.108.3.3	GetJoints	570
10.108.3.4	GetLastUpdateTime	570
10.108.3.5	GetPositionPIDs	570
10.108.3.6	GetPositions	570
10.108.3.7	GetVelocities	570
10.108.3.8	GetVelocityPIDs	571
10.108.3.9	Reset	571
10.108.3.10	SetJointPosition	571
10.108.3.11	SetJointPosition	571
10.108.3.12	SetJointPositions	571
10.108.3.13	SetPositionTarget	571
10.108.3.14	SetVelocityTarget	572
10.108.3.15	Update	572
10.109	gazebo::physics::JointControllerPrivate Class Reference	572
10.109.1	Member Data Documentation	573
10.109.1.1	forces	573
10.109.1.2	jointCmdSub	573
10.109.1.3	joints	573
10.109.1.4	model	573
10.109.1.5	node	573
10.109.1.6	positions	573
10.109.1.7	posPids	573
10.109.1.8	prevUpdateTime	573
10.109.1.9	updatedLinks	574
10.109.1.10	velocities	574
10.109.1.11	velPids	574

10.110	gazebo::physics::JointState Class Reference	574
10.110.1	Detailed Description	575
10.110.2	Constructor & Destructor Documentation	575
10.110.2.1	JointState	575
10.110.2.2	JointState	575
10.110.2.3	JointState	576
10.110.2.4	JointState	576
10.110.2.5	~JointState	576
10.110.3	Member Function Documentation	576
10.110.3.1	FillSDF	576
10.110.3.2	GetAngle	576
10.110.3.3	GetAngleCount	577
10.110.3.4	GetAngles	577
10.110.3.5	IsZero	577
10.110.3.6	Load	577
10.110.3.7	Load	577
10.110.3.8	operator+	577
10.110.3.9	operator-	578
10.110.3.10	operator=	578
10.110.4	Friends And Related Function Documentation	578
10.110.4.1	operator<<	578
10.111	gazebo::rendering::JointVisual Class Reference	579
10.111.1	Detailed Description	579
10.111.2	Constructor & Destructor Documentation	579
10.111.2.1	JointVisual	579
10.111.2.2	~JointVisual	580
10.111.3	Member Function Documentation	580
10.111.3.1	Load	580
10.112	gazebo::rendering::JointVisualPrivate Class Reference	580
10.112.1	Detailed Description	581
10.112.2	Member Data Documentation	581
10.112.2.1	axisVisual	581
10.113	gazebo::physics::JointWrench Class Reference	581
10.113.1	Detailed Description	581
10.113.2	Member Function Documentation	582
10.113.2.1	operator+	582
10.113.2.2	operator-	582

10.113.2.3operator=	582
10.113.3Member Data Documentation	582
10.113.3.1body1Force	582
10.113.3.2body1Torque	583
10.113.3.3body2Force	583
10.113.3.4body2Torque	583
10.114gazebo::common::KeyEvent Class Reference	583
10.114.1Detailed Description	583
10.114.2Member Enumeration Documentation	584
10.114.2.1EventType	584
10.114.3Constructor & Destructor Documentation	584
10.114.3.1KeyEvent	584
10.114.4Member Data Documentation	584
10.114.4.1key	584
10.114.4.2type	584
10.115gazebo::common::KeyFrame Class Reference	584
10.115.1Detailed Description	585
10.115.2Constructor & Destructor Documentation	585
10.115.2.1KeyFrame	585
10.115.2.2~KeyFrame	585
10.115.3Member Function Documentation	585
10.115.3.1GetTime	585
10.115.4Member Data Documentation	585
10.115.4.1time	585
10.116gazebo::rendering::LaserVisual Class Reference	586
10.116.1Detailed Description	586
10.116.2Constructor & Destructor Documentation	586
10.116.2.1LaserVisual	586
10.116.2.2~LaserVisual	587
10.116.3Member Function Documentation	587
10.116.3.1SetEmissive	587
10.117gazebo::rendering::LaserVisualPrivate Class Reference	587
10.117.1Detailed Description	588
10.117.2Member Data Documentation	588
10.117.2.1connection	588
10.117.2.2laserMsg	588
10.117.2.3laserScanSub	588

10.117.2.4	mutex	588
10.117.2.5	node	588
10.117.2.6	rayFans	588
10.117.2.7	receivedMsg	589
10.118	Gazebo::rendering::Light Class Reference	589
10.118.1	Detailed Description	590
10.118.2	Constructor & Destructor Documentation	590
10.118.2.1	Light	590
10.118.2.2	~Light	591
10.118.3	Member Function Documentation	591
10.118.3.1	FillMsg	591
10.118.3.2	GetDiffuseColor	591
10.118.3.3	GetDirection	591
10.118.3.4	GetName	591
10.118.3.5	GetPosition	591
10.118.3.6	GetSpecularColor	592
10.118.3.7	GetType	592
10.118.3.8	GetVisible	592
10.118.3.9	Load	592
10.118.3.10	Load	592
10.118.3.11	loadFromMsg	592
10.118.3.12	loadFromMsg	592
10.118.3.13	OnPoseChange	593
10.118.3.14	SetAttenuation	593
10.118.3.15	SetCastShadows	593
10.118.3.16	SetDiffuseColor	593
10.118.3.17	SetDirection	593
10.118.3.18	SetLightType	593
10.118.3.19	SetName	594
10.118.3.20	SetPosition	594
10.118.3.21	SetRange	594
10.118.3.22	SetSelected	594
10.118.3.23	SetSpecularColor	594
10.118.3.24	SetSpotFalloff	594
10.118.3.25	SetSpotInnerAngle	595
10.118.3.26	SetSpotOuterAngle	595
10.118.3.27	ShowVisual	595

10.118.3.28	ToggleShowVisual	595
10.118.3.29	UpdateFromMsg	595
10.119	gazebo::physics::Link Class Reference	595
10.119.1	Detailed Description	600
10.119.2	Member Typedef Documentation	601
10.119.2.1	Visuals_M	601
10.119.3	Constructor & Destructor Documentation	601
10.119.3.1	Link	601
10.119.3.2	~Link	601
10.119.4	Member Function Documentation	601
10.119.4.1	AddChildJoint	601
10.119.4.2	AddForce	601
10.119.4.3	AddForceAtRelativePosition	601
10.119.4.4	AddForceAtWorldPosition	602
10.119.4.5	AddParentJoint	602
10.119.4.6	AddRelativeForce	602
10.119.4.7	AddRelativeTorque	602
10.119.4.8	AddTorque	602
10.119.4.9	AttachStaticModel	603
10.119.4.10	ConnectEnabled	603
10.119.4.11	DetachAllStaticModels	603
10.119.4.12	DetachStaticModel	603
10.119.4.13	DisconnectEnabled	603
10.119.4.14	FillMsg	603
10.119.4.15	Ini	604
10.119.4.16	GetAngularDamping	604
10.119.4.17	GetBoundingBox	604
10.119.4.18	GetChildJoints	604
10.119.4.19	GetChildJointsLinks	604
10.119.4.20	GetCollision	604
10.119.4.21	GetCollision	605
10.119.4.22	GetCollisionById	605
10.119.4.23	GetCollisions	605
10.119.4.24	GetEnabled	605
10.119.4.25	SetGravityMode	606
10.119.4.26	GetInertial	606
10.119.4.27	GetKinematic	606

10.119.4.28	GetLinearDamping	606
10.119.4.29	GetModel	606
10.119.4.30	GetParentJoints	606
10.119.4.31	GetParentJointsLinks	607
10.119.4.32	GetRelativeAngularAccel	607
10.119.4.33	GetRelativeAngularVel	607
10.119.4.34	GetRelativeForce	607
10.119.4.35	GetRelativeLinearAccel	607
10.119.4.36	GetRelativeLinearVel	607
10.119.4.37	GetRelativeTorque	608
10.119.4.38	GetSelfCollide	608
10.119.4.39	GetSensorCount	608
10.119.4.40	GetSensorName	608
10.119.4.41	GetWorldAngularAccel	608
10.119.4.42	GetWorldCoGLinearVel	609
10.119.4.43	GetWorldCoGPose	609
10.119.4.44	GetWorldEnergy	609
10.119.4.45	GetWorldEnergyKinetic	609
10.119.4.46	GetWorldEnergyPotential	609
10.119.4.47	GetWorldForce	610
10.119.4.48	GetWorldInertialPose	610
10.119.4.49	GetWorldInertiaMatrix	610
10.119.4.50	GetWorldLinearAccel	610
10.119.4.51	GetWorldLinearVel	610
10.119.4.52	GetWorldLinearVel	611
10.119.4.53	GetWorldLinearVel	611
10.119.4.54	GetWorldTorque	611
10.119.4.55	hit	611
10.119.4.56	load	612
10.119.4.57	onPoseChange	612
10.119.4.58	ProcessMsg	612
10.119.4.59	removeChild	612
10.119.4.60	removeChildJoint	612
10.119.4.61	removeCollision	612
10.119.4.62	removeParentJoint	612
10.119.4.63	reset	613
10.119.4.64	resetPhysicsStates	613

10.119.4.65	SetAngularAccel	613
10.119.4.66	SetAngularDamping	613
10.119.4.67	SetAngularVel	613
10.119.4.68	SetAutoDisable	613
10.119.4.69	SetCollideMode	614
10.119.4.70	SetEnabled	614
10.119.4.71	SetForce	614
10.119.4.72	SetGravityMode	614
10.119.4.73	SetInertial	614
10.119.4.74	SetKinematic	615
10.119.4.75	SetLaserRetro	615
10.119.4.76	SetLinearAccel	615
10.119.4.77	SetLinearDamping	615
10.119.4.78	SetLinearVel	615
10.119.4.79	SetLinkStatic	615
10.119.4.80	SetPublishData	616
10.119.4.81	SetScale	616
10.119.4.82	SetSelected	616
10.119.4.83	SetSelfCollide	616
10.119.4.84	SetState	616
10.119.4.85	SetTorque	617
10.119.4.86	Update	617
10.119.4.87	UpdateMass	617
10.119.4.88	UpdateParameters	617
10.119.4.89	UpdateSurface	617
10.119.5	Member Data Documentation	617
10.119.5.1	angularAccel	617
10.119.5.2	attachedModelsOffset	617
10.119.5.3	cgVisuals	617
10.119.5.4	inertial	618
10.119.5.5	initialized	618
10.119.5.6	linearAccel	618
10.119.5.7	visuals	618
10.120	gazebo::physics::LinkState Class Reference	618
10.120.1	Detailed Description	620
10.120.2	Constructor & Destructor Documentation	620
10.120.2.1	LinkState	620

10.120.2.2	LinkState	620
10.120.2.3	LinkState	620
10.120.2.4	LinkState	620
10.120.2.5	~LinkState	621
10.120.3	Member Function Documentation	621
10.120.3.1	FillSDF	621
10.120.3.2	GetAcceleration	621
10.120.3.3	GetCollisionState	621
10.120.3.4	GetCollisionState	621
10.120.3.5	GetCollisionStateCount	622
10.120.3.6	GetCollisionStates	622
10.120.3.7	GetPose	622
10.120.3.8	GetVelocity	622
10.120.3.9	GetWrench	623
10.120.3.10	Zero	623
10.120.3.11	load	623
10.120.3.12	load	623
10.120.3.13	operator+	623
10.120.3.14	operator-	624
10.120.3.15	operator=	624
10.120.3.16	SetRealTime	624
10.120.3.17	SetSimTime	624
10.120.3.18	SetWallTime	625
10.120.4	Friends And Related Function Documentation	625
10.120.4.1	operator<<	625
10.121	gazebo::common::Logger Class Reference	625
10.121.1	Detailed Description	627
10.121.2	Member Enumeration Documentation	627
10.121.2.1	LogType	627
10.121.3	Constructor & Destructor Documentation	627
10.121.3.1	Logger	627
10.121.3.2	~Logger	627
10.121.4	Member Function Documentation	627
10.121.4.1	operator()	627
10.121.4.2	operator()	628
10.121.5	Member Data Documentation	628
10.121.5.1	color	628

10.122.0	gazebo::util::LogPlay Class Reference	628
10.122.1	Member Function Documentation	629
10.122.1.1	GetChunk	629
10.122.1.2	GetChunkCount	629
10.122.1.3	GetEncoding	629
10.122.1.4	GetGazeboVersion	630
10.122.1.5	GetHeader	630
10.122.1.6	GetLogVersion	630
10.122.1.7	GetRandSeed	630
10.122.1.8	IsOpen	630
10.122.1.9	Open	631
10.122.1.10	Step	631
10.122.2	Logplay Class Reference	631
10.123.1	Detailed Description	631
10.124.0	gazebo::util::LogRecord Class Reference	631
10.124.1	Detailed Description	633
10.124.2	Member Function Documentation	633
10.124.2.1	Add	633
10.124.2.2	Finis	634
10.124.2.3	GetBasePath	634
10.124.2.4	GetBufferSize	634
10.124.2.5	GetEncoding	634
10.124.2.6	GetFilename	634
10.124.2.7	GetFileSize	634
10.124.2.8	GetFirstUpdate	635
10.124.2.9	GetPaused	635
10.124.2.10	GetRunning	635
10.124.2.11	GetRunTime	635
10.124.2.12	Wait	635
10.124.2.13	ReadyToStart	636
10.124.2.14	Notify	636
10.124.2.15	Remove	636
10.124.2.16	SetBasePath	636
10.124.2.17	SetPaused	636
10.124.2.18	Start	637
10.124.2.19	Stop	637
10.124.2.20	Write	637

10.125	<code>gazebo::Master</code> Class Reference	637
10.125.1	Detailed Description	638
10.125.2	Constructor & Destructor Documentation	638
10.125.2.1	<code>Master</code>	638
10.125.2.2	<code>~Master</code>	638
10.125.3	Member Function Documentation	638
10.125.3.1	<code>Fin</code>	638
10.125.3.2	<code>nit</code>	638
10.125.3.3	<code>Run</code>	638
10.125.3.4	<code>RunOnce</code>	638
10.125.3.5	<code>RunThread</code>	639
10.125.3.6	<code>Stop</code>	639
10.126	<code>gazebo::common::Material</code> Class Reference	639
10.126.1	Detailed Description	641
10.126.2	Member Enumeration Documentation	641
10.126.2.1	<code>BlendMode</code>	641
10.126.2.2	<code>ShadeMode</code>	642
10.126.3	Constructor & Destructor Documentation	642
10.126.3.1	<code>Material</code>	642
10.126.3.2	<code>~Material</code>	642
10.126.3.3	<code>Material</code>	642
10.126.4	Member Function Documentation	642
10.126.4.1	<code>GetAmbient</code>	642
10.126.4.2	<code>GetBlendFactors</code>	642
10.126.4.3	<code>GetBlendMode</code>	643
10.126.4.4	<code>GetDepthWrite</code>	643
10.126.4.5	<code>GetDiffuse</code>	643
10.126.4.6	<code>GetEmissive</code>	643
10.126.4.7	<code>GetLighting</code>	643
10.126.4.8	<code>GetName</code>	643
10.126.4.9	<code>GetPointSize</code>	644
10.126.4.10	<code>GetShadeMode</code>	644
10.126.4.11	<code>GetShininess</code>	644
10.126.4.12	<code>GetSpecular</code>	644
10.126.4.13	<code>GetTextureImage</code>	644
10.126.4.14	<code>GetTransparency</code>	644
10.126.4.15	<code>SetAmbient</code>	645

10.126.4.1	SetBlendFactors	645
10.126.4.1	SetBlendMode	645
10.126.4.1	SetDepthWrite	645
10.126.4.1	SetDiffuse	645
10.126.4.2	SetEmissive	645
10.126.4.2	SetLighting	646
10.126.4.2	SetPointSize	646
10.126.4.2	SetShadeMode	646
10.126.4.2	SetShininess	646
10.126.4.2	SetSpecular	646
10.126.4.2	SetTextureImage	646
10.126.4.2	SetTextureImage	647
10.126.4.2	SetTransparency	647
10.126.5	Friends And Related Function Documentation	647
10.126.5.1	operator<<	647
10.126.6	Member Data Documentation	647
10.126.6.1	ambient	647
10.126.6.2	blendMode	647
10.126.6.3	BlendModeStr	647
10.126.6.4	diffuse	647
10.126.6.5	emissive	647
10.126.6.6	name	647
10.126.6.7	pointSize	648
10.126.6.8	shadeMode	648
10.126.6.9	ShadeModeStr	648
10.126.6.10	shininess	648
10.126.6.11	specular	648
10.126.6.12	texImage	648
10.126.6.13	transparency	648
10.127	gazebo::math::Matrix3 Class Reference	648
10.127.1	Detailed Description	649
10.127.2	Constructor & Destructor Documentation	650
10.127.2.1	Matrix3	650
10.127.2.2	Matrix3	650
10.127.2.3	Matrix3	650
10.127.2.4	~Matrix3	650
10.127.3	Member Function Documentation	650

10.127.3.1operator*	650
10.127.3.2operator*	650
10.127.3.3operator*	651
10.127.3.4operator+	651
10.127.3.5operator-	651
10.127.3.6operator==	651
10.127.3.7operator[]	651
10.127.3.8operator[]	652
10.127.3.9SetCol	652
10.127.3.10SetFromAxes	652
10.127.3.11SetFromAxis	652
10.127.4Friends And Related Function Documentation	652
10.127.4.1operator*	652
10.127.4.2operator<<	653
10.127.5Member Data Documentation	653
10.127.5.1IDENTITY	653
10.127.5.2m	653
10.127.5.3ZERO	653
10.128Gazebo::math::Matrix4 Class Reference	653
10.128.1Detailed Description	655
10.128.2Constructor & Destructor Documentation	655
10.128.2.1Matrix4	655
10.128.2.2Matrix4	655
10.128.2.3Matrix4	655
10.128.2.4~Matrix4	656
10.128.3Member Function Documentation	656
10.128.3.1GetAsPose	656
10.128.3.2GetEulerRotation	656
10.128.3.3GetRotation	656
10.128.3.4GetTranslation	656
10.128.3.5Inverse	657
10.128.3.6IsAffine	657
10.128.3.7operator*	657
10.128.3.8operator*	657
10.128.3.9operator*	657
10.128.3.10operator=	658
10.128.3.11operator=	658

10.128.3.10	operator==	658
10.128.3.10	operator[]	658
10.128.3.10	operator[]	659
10.128.3.15	Set	659
10.128.3.15	SetScale	659
10.128.3.15	SetTranslate	659
10.128.3.18	TransformAffine	660
10.128.4	Friends And Related Function Documentation	660
10.128.4.1	operator<<	660
10.128.5	Member Data Documentation	660
10.128.5.1	IDENTITY	660
10.128.5.2	m	660
10.128.5.3	ZERO	660
10.129	gazebo::common::Mesh Class Reference	660
10.129.1	Detailed Description	662
10.129.2	Constructor & Destructor Documentation	662
10.129.2.1	Mesh	662
10.129.2.2	~Mesh	662
10.129.3	Member Function Documentation	662
10.129.3.1	AddMaterial	662
10.129.3.2	AddSubMesh	662
10.129.3.3	Center	663
10.129.3.4	FillArrays	663
10.129.3.5	GenSphericalTexCoord	663
10.129.3.6	GetAABB	663
10.129.3.7	GetIndexCount	663
10.129.3.8	GetMaterial	664
10.129.3.9	GetMaterialCount	664
10.129.3.10	GetMax	664
10.129.3.10	GetMin	664
10.129.3.10	GetName	664
10.129.3.10	GetNormalCount	665
10.129.3.10	GetPath	665
10.129.3.10	GetSkeleton	665
10.129.3.10	GetSubMesh	665
10.129.3.10	GetSubMesh	665
10.129.3.10	GetSubMeshCount	666

10.129.3.10	GetTexCoordCount	666
10.129.3.20	GetVertexCount	666
10.129.3.21	HasSkeleton	666
10.129.3.22	RecalculateNormals	666
10.129.3.23	Scale	666
10.129.3.23	SetName	666
10.129.3.25	SetPath	667
10.129.3.26	SetScale	667
10.129.3.27	SetSkeleton	667
10.129.3.28	Translate	667
10.130	Gazebo::common::MeshCSG Class Reference	667
10.130.1	Detailed Description	668
10.130.2	Member Enumeration Documentation	668
10.130.2.1	BooleanOperation	668
10.130.3	Constructor & Destructor Documentation	668
10.130.3.1	MeshCSG	668
10.130.3.2	~MeshCSG	668
10.130.4	Member Function Documentation	668
10.130.4.1	CreateBoolean	668
10.131	Gazebo::common::MeshLoader Class Reference	669
10.131.1	Detailed Description	669
10.131.2	Constructor & Destructor Documentation	669
10.131.2.1	MeshLoader	669
10.131.2.2	~MeshLoader	669
10.131.3	Member Function Documentation	670
10.131.3.1	Load	670
10.132	Gazebo::common::MeshManager Class Reference	670
10.132.1	Detailed Description	671
10.132.2	Member Function Documentation	671
10.132.2.1	AddMesh	671
10.132.2.2	CreateBox	672
10.132.2.3	CreateCamera	672
10.132.2.4	CreateCone	672
10.132.2.5	CreateCylinder	672
10.132.2.6	CreatePlane	672
10.132.2.7	CreatePlane	673
10.132.2.8	CreateSphere	673

10.132.2.9	CreateTube	673
10.132.2.10	GenSphericalTexCoord	674
10.132.2.10	GetMesh	674
10.132.2.10	GetMeshAABB	674
10.132.2.11	HasMesh	674
10.132.2.11	ValidFilename	674
10.132.2.11	Load	674
10.133	gazebo::physics::MeshShape Class Reference	675
10.133.1	Detailed Description	676
10.133.2	Constructor & Destructor Documentation	676
10.133.2.1	MeshShape	676
10.133.2.2	~MeshShape	676
10.133.3	Member Function Documentation	676
10.133.3.1	FillMsg	676
10.133.3.2	GetMeshURI	677
10.133.3.3	GetSize	677
10.133.3.4	Init	677
10.133.3.5	ProcessMsg	677
10.133.3.6	SetMesh	677
10.133.3.7	SetScale	678
10.133.3.8	Update	678
10.133.4	Member Data Documentation	678
10.133.4.1	mesh	678
10.133.4.2	submesh	678
10.134	gazebo::physics::Model Class Reference	678
10.134.1	Detailed Description	682
10.134.2	Constructor & Destructor Documentation	682
10.134.2.1	Model	682
10.134.2.2	~Model	682
10.134.3	Member Function Documentation	682
10.134.3.1	AttachStaticModel	682
10.134.3.2	DetachStaticModel	683
10.134.3.3	FillMsg	683
10.134.3.4	Fini	683
10.134.3.5	GetAutoDisable	683
10.134.3.6	GetBoundingBox	683
10.134.3.7	GetGripper	683

10.134.3.8	GetGripperCount	684
10.134.3.9	GetJoint	684
10.134.3.10	GetJointController	684
10.134.3.10	GetJointCount	684
10.134.3.10	GetJoints	684
10.134.3.10	GetLink	685
10.134.3.10	GetLinkById	685
10.134.3.10	GetLinks	685
10.134.3.10	GetPluginCount	685
10.134.3.10	GetRelativeAngularAccel	685
10.134.3.10	GetRelativeAngularVel	686
10.134.3.10	GetRelativeLinearAccel	686
10.134.3.20	GetRelativeLinearVel	686
10.134.3.20	GetSDF	686
10.134.3.20	GetSensorCount	686
10.134.3.20	GetWorldAngularAccel	687
10.134.3.20	GetWorldAngularVel	687
10.134.3.20	GetWorldEnergy	687
10.134.3.20	GetWorldEnergyKinetic	687
10.134.3.20	GetWorldEnergyPotential	687
10.134.3.20	GetWorldLinearAccel	688
10.134.3.20	GetWorldLinearVel	688
10.134.3.30	hit	688
10.134.3.30	load	688
10.134.3.30	loadJoints	688
10.134.3.30	loadPlugins	688
10.134.3.30	onPoseChange	689
10.134.3.30	processMsg	689
10.134.3.30	removeChild	689
10.134.3.30	reset	689
10.134.3.30	setAngularAccel	689
10.134.3.30	setAngularVel	689
10.134.3.40	setAutoDisable	689
10.134.3.40	setCollideMode	690
10.134.3.40	setEnabled	690
10.134.3.40	setGravityMode	690
10.134.3.40	setJointAnimation	690

10.134.3.45	SetJointPosition	690
10.134.3.46	SetJointPositions	691
10.134.3.47	SetLaserRetro	691
10.134.3.48	SetLinearAccel	691
10.134.3.49	SetLinearVel	691
10.134.3.50	SetLinkWorldPose	691
10.134.3.51	SetLinkWorldPose	692
10.134.3.52	SetScale	692
10.134.3.53	SetState	692
10.134.3.54	StopAnimation	692
10.134.3.55	Update	692
10.134.3.56	UpdateParameters	692
10.134.4	Member Data Documentation	693
10.134.4.1	attachedModels	693
10.134.4.2	attachedModelsOffset	693
10.134.4.3	jointPub	693
10.135	gazebo::common::ModelDatabase Class Reference	693
10.135.1	Detailed Description	694
10.136	gazebo::common::ModelDatabasePrivate Class Reference	694
10.136.1	Detailed Description	695
10.136.2	Member Typedef Documentation	695
10.136.2.1	CallbackFunc	695
10.136.3	Member Data Documentation	695
10.136.3.1	callbacksMutex	695
10.136.3.2	deprecatedCallbacks	695
10.136.3.3	modelCache	695
10.136.3.4	modelDBUpdated	695
10.136.3.5	startCacheMutex	696
10.136.3.6	stop	696
10.136.3.7	updateCacheCompleteCondition	696
10.136.3.8	updateCacheCondition	696
10.136.3.9	updateCacheThread	696
10.136.3.10	updateMutex	696
10.137	gazebo::ModelPlugin Class Reference	696
10.137.1	Detailed Description	697
10.137.2	Constructor & Destructor Documentation	697
10.137.2.1	ModelPlugin	697

10.137.2.2~ModelPlugin	697
10.137.3 Member Function Documentation	697
10.137.3.1Init	697
10.137.3.2Load	697
10.137.3.3Reset	698
10.138 gazebo::physics::ModelState Class Reference	698
10.138.1 Detailed Description	700
10.138.2 Constructor & Destructor Documentation	700
10.138.2.1 ModelState	700
10.138.2.2 ModelState	700
10.138.2.3 ModelState	700
10.138.2.4 ModelState	700
10.138.2.5 ~ModelState	701
10.138.3 Member Function Documentation	701
10.138.3.1 FillSDF	701
10.138.3.2 GetJointState	701
10.138.3.3 GetJointState	701
10.138.3.4 GetJointStateCount	702
10.138.3.5 GetJointStates	702
10.138.3.6 GetJointStates	702
10.138.3.7 GetLinkState	702
10.138.3.8 GetLinkStateCount	703
10.138.3.9 GetLinkStates	703
10.138.3.10 GetLinkStates	703
10.138.3.11 GetPose	703
10.138.3.12 HasJointState	703
10.138.3.13 HasLinkState	704
10.138.3.14 Zero	704
10.138.3.15 Load	704
10.138.3.16 Load	704
10.138.3.17 operator+	704
10.138.3.18 operator-	705
10.138.3.19 operator=	705
10.138.3.20 SetRealTime	705
10.138.3.21 SetSimTime	705
10.138.3.22 SetWallTime	706
10.138.4 Friends And Related Function Documentation	706

10.138.4.1operator<<	706
10.139.0 gazebo::common::MouseEvent Class Reference	706
10.139.1 Detailed Description	707
10.139.2 Member Enumeration Documentation	707
10.139.2.1 Buttons	707
10.139.2.2 EventType	708
10.139.3 Constructor & Destructor Documentation	708
10.139.3.1 MouseEvent	708
10.139.4 Member Data Documentation	708
10.139.4.1 alt	708
10.139.4.2 button	708
10.139.4.3 buttons	708
10.139.4.4 control	708
10.139.4.5 dragging	708
10.139.4.6 moveScale	708
10.139.4.7 pos	708
10.139.4.8 pressPos	709
10.139.4.9 prevPos	709
10.139.4.10 scroll	709
10.139.4.11 shift	709
10.139.4.12 type	709
10.140.0 gazebo::rendering::MovableText Class Reference	709
10.140.1 Detailed Description	711
10.140.2 Member Enumeration Documentation	711
10.140.2.1 HorizAlign	711
10.140.2.2 VertAlign	711
10.140.3 Constructor & Destructor Documentation	711
10.140.3.1 MovableText	711
10.140.3.2 ~MovableText	711
10.140.4 Member Function Documentation	711
10.140.4.1 _setupGeometry	711
10.140.4.2 _updateColors	711
10.140.4.3 GetAABB	711
10.140.4.4 GetBaseline	712
10.140.4.5 getBoundingRadius	712
10.140.4.6 GetCharHeight	712
10.140.4.7 GetColor	712

10.140.4.8	GetFont	712
10.140.4.9	getLights	712
10.140.4.10	GetMaterial	712
10.140.4.11	getRenderOperation	712
10.140.4.12	GetShowOnTop	712
10.140.4.13	GetSpaceWidth	712
10.140.4.14	GetSquaredViewDepth	713
10.140.4.15	GetText	713
10.140.4.16	GetWorldTransforms	713
10.140.4.17	Load	713
10.140.4.18	SetBaseline	713
10.140.4.19	SetCharHeight	713
10.140.4.20	SetColor	713
10.140.4.21	SetFontName	714
10.140.4.22	SetShowOnTop	714
10.140.4.23	SetSpaceWidth	714
10.140.4.24	SetText	714
10.140.4.25	SetTextAlignment	714
10.140.4.26	Update	714
10.140.4.27	visitRenderables	715
10.141	gazebo::common::MovingWindowFilter< T > Class Template Reference	715
10.141.1	Detailed Description	715
10.141.2	Constructor & Destructor Documentation	716
10.141.2.1	MovingWindowFilter	716
10.141.3	Member Data Documentation	716
10.141.3.1	dataPtr	716
10.142	gazebo::common::MovingWindowFilterPrivate< T > Class Template Reference	716
10.142.1	Member Data Documentation	716
10.142.1.1	Isamples	716
10.142.1.2	sum	717
10.142.1.3	valHistory	717
10.142.1.4	valIter	717
10.142.1.5	valWindowSize	717
10.143	gazebo::msgs::MsgFactory Class Reference	717
10.143.1	Detailed Description	717
10.143.2	Member Function Documentation	717
10.143.2.1	GetMsgTypes	717

10.143.2.2	NewMsg	718
10.143.2.3	RegisterMsg	718
10.144	Gazebo::sensors::MultiCameraSensor Class Reference	718
10.144.1	Detailed Description	720
10.144.2	Constructor & Destructor Documentation	720
10.144.2.1	MultiCameraSensor	720
10.144.2.2	~MultiCameraSensor	720
10.144.3	Member Function Documentation	720
10.144.3.1	Finis	720
10.144.3.2	GetCamera	720
10.144.3.3	GetCameraCount	721
10.144.3.4	GetImageData	721
10.144.3.5	GetImageHeight	721
10.144.3.6	GetImageWidth	721
10.144.3.7	GetTopic	722
10.144.3.8	Init	722
10.144.3.9	IsActive	722
10.144.3.10	Load	722
10.144.3.11	SaveFrame	722
10.144.3.12	UpdateImpl	723
10.145	Gazebo::physics::MultiRayShape Class Reference	723
10.145.1	Detailed Description	726
10.145.2	Constructor & Destructor Documentation	726
10.145.2.1	MultiRayShape	726
10.145.2.2	~MultiRayShape	726
10.145.3	Member Function Documentation	726
10.145.3.1	AddRay	726
10.145.3.2	ConnectNewLaserScans	726
10.145.3.3	DisconnectNewLaserScans	727
10.145.3.4	FillMsg	727
10.145.3.5	GetFiducial	727
10.145.3.6	GetMaxAngle	727
10.145.3.7	GetMaxRange	727
10.145.3.8	GetMinAngle	728
10.145.3.9	GetMinRange	728
10.145.3.10	GetRange	728
10.145.3.11	GetResRange	728

10.145.3.10	GetRetro	728
10.145.3.11	GetSampleCount	729
10.145.3.12	GetScanResolution	729
10.145.3.13	GetVerticalMaxAngle	729
10.145.3.14	GetVerticalMinAngle	729
10.145.3.15	GetVerticalSampleCount	729
10.145.3.16	GetVerticalScanResolution	729
10.145.3.17	Init	730
10.145.3.18	ProcessMsg	730
10.145.3.19	SetScale	730
10.145.3.20	Update	730
10.145.3.21	UpdateRays	730
10.145.4	Member Data Documentation	730
10.145.4.1	horzElem	730
10.145.4.2	newLaserScans	730
10.145.4.3	offset	731
10.145.4.4	rangeElem	731
10.145.4.5	rayElem	731
10.145.4.6	rays	731
10.145.4.7	scanElem	731
10.145.4.8	vertElem	731
10.146	Gazebo::transport::Node Class Reference	731
10.146.1	Detailed Description	732
10.146.2	Constructor & Destructor Documentation	733
10.146.2.1	Node	733
10.146.2.2	~Node	733
10.146.3	Member Function Documentation	733
10.146.3.1	Advertise	733
10.146.3.2	DecodeTopicName	733
10.146.3.3	EncodeTopicName	733
10.146.3.4	Finis	734
10.146.3.5	GetId	734
10.146.3.6	GetMsgType	734
10.146.3.7	GetTopicNamespace	734
10.146.3.8	HandleData	734
10.146.3.9	HandleMessage	735
10.146.3.10	HasLatchedSubscriber	735

10.146.3.11hit	735
10.146.3.12InsertLatchedMsg	735
10.146.3.13InsertLatchedMsg	735
10.146.3.14ProcessIncoming	736
10.146.3.15ProcessPublishers	736
10.146.3.16Publish	736
10.146.3.17RemoveCallback	736
10.146.3.18Subscribe	736
10.146.3.19Subscribe	736
10.146.3.20Subscribe	737
10.146.3.21Subscribe	737
10.147.0 gazebo::common::NodeAnimation Class Reference	738
10.147.1 Detailed Description	738
10.147.2 Constructor & Destructor Documentation	739
10.147.2.1 NodeAnimation	739
10.147.2.2 ~NodeAnimation	739
10.147.3 Member Function Documentation	739
10.147.3.1 AddKeyFrame	739
10.147.3.2 AddKeyFrame	739
10.147.3.3 GetFrameAt	739
10.147.3.4 GetFrameCount	739
10.147.3.5 GetKeyFrame	740
10.147.3.6 GetKeyFrame	740
10.147.3.7 GetLength	740
10.147.3.8 GetName	740
10.147.3.9 GetTimeAtX	740
10.147.3.10 Scale	741
10.147.3.11 SetName	741
10.147.4 Member Data Documentation	741
10.147.4.1 keyFrames	741
10.147.4.2 length	741
10.147.4.3 name	741
10.148.0 gazebo::common::NodeAssignment Class Reference	741
10.148.1 Detailed Description	742
10.148.2 Constructor & Destructor Documentation	742
10.148.2.1 NodeAssignment	742
10.148.3 Member Data Documentation	742

10.148.3.1nodeIndex	742
10.148.3.2vertexIndex	742
10.148.3.3weight	742
10.149 gazebo::common::NodeTransform Class Reference	742
10.149.1 Detailed Description	744
10.149.2 Member Enumeration Documentation	744
10.149.2.1 TransformType	744
10.149.3 Constructor & Destructor Documentation	744
10.149.3.1 NodeTransform	744
10.149.3.2 NodeTransform	744
10.149.3.3 ~NodeTransform	745
10.149.4 Member Function Documentation	745
10.149.4.1 Get	745
10.149.4.2 GetSID	745
10.149.4.3 GetType	745
10.149.4.4 operator()	745
10.149.4.5 operator*	745
10.149.4.6 operator*	746
10.149.4.7 PrintSource	746
10.149.4.8 RecalculateMatrix	746
10.149.4.9 Set	746
10.149.4.10 SetComponent	746
10.149.4.11 SetSID	746
10.149.4.12 SetSourceValues	747
10.149.4.13 SetSourceValues	747
10.149.4.14 SetSourceValues	747
10.149.4.15 SetType	747
10.149.5 Member Data Documentation	747
10.149.5.1 sid	747
10.149.5.2 source	747
10.149.5.3 transform	747
10.149.5.4 type	747
10.150 gazebo::sensors::Noise Class Reference	748
10.150.1 Detailed Description	749
10.150.2 Member Enumeration Documentation	749
10.150.2.1 NoiseType	749
10.150.3 Constructor & Destructor Documentation	749

10.150.3.1	Noise	749
10.150.3.2	~Noise	749
10.150.4	Member Function Documentation	749
10.150.4.1	Apply	749
10.150.4.2	ApplyImpl	750
10.150.4.3	Finis	750
10.150.4.4	GetNoiseType	750
10.150.4.5	Load	750
10.150.4.6	SetCamera	751
10.150.4.7	SetCustomNoiseCallback	751
10.150	gazebo::sensors::NoiseFactory Class Reference	751
10.151.1	Detailed Description	751
10.151.2	Member Function Documentation	751
10.151.2.1	NewNoiseModel	751
10.150	gazebo::common::NumericAnimation Class Reference	752
10.152.1	Detailed Description	753
10.152.2	Constructor & Destructor Documentation	753
10.152.2.1	NumericAnimation	753
10.152.2.2	~NumericAnimation	753
10.152.3	Member Function Documentation	753
10.152.3.1	CreateKeyFrame	753
10.152.3.2	GetInterpolatedKeyFrame	753
10.150	gazebo::common::NumericKeyFrame Class Reference	754
10.153.1	Detailed Description	754
10.153.2	Constructor & Destructor Documentation	754
10.153.2.1	NumericKeyFrame	754
10.153.2.2	~NumericKeyFrame	755
10.153.3	Member Function Documentation	755
10.153.3.1	GetValue	755
10.153.3.2	SetValue	755
10.153.4	Member Data Documentation	755
10.153.4.1	value	755
10.150	gazebo::util::OpenAL Class Reference	755
10.154.1	Detailed Description	756
10.154.2	Member Function Documentation	756
10.154.2.1	CreateSink	756
10.154.2.2	CreateSource	757

10.154.2.3	Fin	757
10.154.2.4	Load	757
10.155	gazebo::util::OpenALSink Class Reference	757
10.155.1	Detailed Description	757
10.155.2	Constructor & Destructor Documentation	758
10.155.2.1	OpenALSink	758
10.155.2.2	~OpenALSink	758
10.155.3	Member Function Documentation	758
10.155.3.1	SetPose	758
10.155.3.2	SetVelocity	758
10.156	gazebo::util::OpenALSource Class Reference	758
10.156.1	Detailed Description	759
10.156.2	Constructor & Destructor Documentation	759
10.156.2.1	OpenALSource	759
10.156.2.2	~OpenALSource	760
10.156.3	Member Function Documentation	760
10.156.3.1	FillBufferFromFile	760
10.156.3.2	FillBufferFromPCM	760
10.156.3.3	GetCollisionNames	760
10.156.3.4	GetOnContact	760
10.156.3.5	HasCollisionName	761
10.156.3.6	IsPlaying	761
10.156.3.7	Load	761
10.156.3.8	Pause	761
10.156.3.9	Play	761
10.156.3.10	Rewind	761
10.156.3.11	SetGain	761
10.156.3.12	SetLoop	762
10.156.3.13	SetPitch	762
10.156.3.14	SetPose	762
10.156.3.15	SetVelocity	762
10.156.3.16	Stop	763
10.157	gazebo::rendering::OrbitViewController Class Reference	763
10.157.1	Detailed Description	764
10.157.2	Constructor & Destructor Documentation	764
10.157.2.1	OrbitViewController	764
10.157.2.2	~OrbitViewController	764

10.157.3	Member Function Documentation	764
10.157.3.1	GetFocalPoint	764
10.157.3.2	GetTypeString	765
10.157.3.3	HandleKeyPressEvent	765
10.157.3.4	HandleKeyReleaseEvent	765
10.157.3.5	HandleMouseEvent	765
10.157.3.6	Init	765
10.157.3.7	Init	765
10.157.3.8	SetDistance	766
10.157.3.9	SetFocalPoint	766
10.157.3.10	Update	766
10.158	Gazebo::physics::PhysicsEngine Class Reference	766
10.158.1	Detailed Description	770
10.158.2	Constructor & Destructor Documentation	770
10.158.2.1	PhysicsEngine	770
10.158.2.2	~PhysicsEngine	770
10.158.3	Member Function Documentation	770
10.158.3.1	CreateCollision	770
10.158.3.2	CreateCollision	770
10.158.3.3	CreateJoint	771
10.158.3.4	CreateLink	771
10.158.3.5	CreateModel	771
10.158.3.6	CreateShape	771
10.158.3.7	DebugPrint	772
10.158.3.8	Finis	772
10.158.3.9	GetAutoDisableFlag	772
10.158.3.10	GetContactManager	772
10.158.3.11	GetContactMaxCorrectingVel	772
10.158.3.12	GetContactSurfaceLayer	772
10.158.3.13	GetGravity	773
10.158.3.14	GetMaxContacts	773
10.158.3.15	GetMaxStepSize	773
10.158.3.16	GetParam	773
10.158.3.17	GetPhysicsUpdateMutex	773
10.158.3.18	GetRealTimeUpdateRate	774
10.158.3.19	GetSORPGSIters	774
10.158.3.20	GetSORPGSPreconIters	774

10.158.3.20	GetSORPGSW	774
10.158.3.20	GetTargetRealTimeFactor	774
10.158.3.23	GetType	774
10.158.3.24	GetUpdatePeriod	775
10.158.3.25	GetWorldCFM	775
10.158.3.26	GetWorldERP	775
10.158.3.27	Init	775
10.158.3.28	InitForThread	775
10.158.3.29	Load	775
10.158.3.30	OnPhysicsMsg	776
10.158.3.30	OnRequest	776
10.158.3.32	Reset	776
10.158.3.33	SetAutoDisableFlag	776
10.158.3.33	SetContactMaxCorrectingVel	776
10.158.3.35	SetContactSurfaceLayer	776
10.158.3.36	SetGravity	777
10.158.3.37	SetMaxContacts	777
10.158.3.38	SetMaxStepSize	777
10.158.3.39	SetParam	777
10.158.3.40	SetRealTimeUpdateRate	778
10.158.3.41	SetSeed	778
10.158.3.42	SetSORPGSIters	779
10.158.3.43	SetSORPGSPreconIters	779
10.158.3.43	SetSORPGSW	779
10.158.3.45	SetTargetRealTimeFactor	779
10.158.3.46	SetWorldCFM	779
10.158.3.47	SetWorldERP	780
10.158.3.48	UpdateCollision	780
10.158.3.49	UpdatePhysics	780
10.158.4	Member Data Documentation	780
10.158.4.1	contactManager	780
10.158.4.2	maxStepSize	780
10.158.4.3	node	780
10.158.4.4	physicsSub	780
10.158.4.5	physicsUpdateMutex	780
10.158.4.6	realTimeUpdateRate	780
10.158.4.7	requestSub	781

10.158.4.8	responsePub	781
10.158.4.9	sdf	781
10.158.4.10	targetRealTimeFactor	781
10.158.4.11	World	781
10.159	gazebo::physics::PhysicsFactory Class Reference	781
10.159.1	Detailed Description	781
10.159.2	Member Function Documentation	782
10.159.2.1	IsRegistered	782
10.159.2.2	NewPhysicsEngine	782
10.159.2.3	RegisterAll	782
10.159.2.4	RegisterPhysicsEngine	782
10.160	gazebo::common::PID Class Reference	782
10.160.1	Detailed Description	784
10.160.2	Constructor & Destructor Documentation	784
10.160.2.1	PID	784
10.160.2.2	~PID	784
10.160.3	Member Function Documentation	784
10.160.3.1	GetCmd	784
10.160.3.2	GetCmdMax	784
10.160.3.3	GetCmdMin	784
10.160.3.4	GetDGain	785
10.160.3.5	GetErrors	785
10.160.3.6	GetIGain	785
10.160.3.7	GetIMax	785
10.160.3.8	GetIMin	785
10.160.3.9	GetPGain	786
10.160.3.10	it	786
10.160.3.11	operator=	786
10.160.3.12	reset	786
10.160.3.13	SetCmd	786
10.160.3.14	SetCmdMax	787
10.160.3.15	SetCmdMin	787
10.160.3.16	SetDGain	787
10.160.3.17	SetIGain	787
10.160.3.18	SetIMax	787
10.160.3.19	SetIMin	787
10.160.3.20	SetPGain	788

10.160.3.2	Update	788
10.160	gazebo::math::Plane Class Reference	788
10.161.1	Detailed Description	789
10.161.2	Constructor & Destructor Documentation	789
10.161.2.1	Plane	789
10.161.2.2	Plane	789
10.161.2.3	Plane	789
10.161.2.4	~Plane	789
10.161.3	Member Function Documentation	789
10.161.3.1	Distance	789
10.161.3.2	operator=	790
10.161.3.3	Set	790
10.161.4	Member Data Documentation	790
10.161.4.1	id	790
10.161.4.2	normal	790
10.161.4.3	size	790
10.160	gazebo::physics::PlaneShape Class Reference	791
10.162.1	Detailed Description	792
10.162.2	Constructor & Destructor Documentation	792
10.162.2.1	PlaneShape	792
10.162.2.2	~PlaneShape	792
10.162.3	Member Function Documentation	792
10.162.3.1	CreatePlane	792
10.162.3.2	FillMsg	792
10.162.3.3	GetNormal	793
10.162.3.4	GetSize	793
10.162.3.5	Init	793
10.162.3.6	ProcessMsg	793
10.162.3.7	SetAltitude	793
10.162.3.8	SetNormal	793
10.162.3.9	SetScale	794
10.162.3.10	SetSize	794
10.160	gazebo::PluginT< T > Class Template Reference	794
10.163.1	Detailed Description	795
10.163.2	Member Typedef Documentation	795
10.163.2.1	TPtr	795
10.163.3	Constructor & Destructor Documentation	795

10.163.3.1	PluginT	795
10.163.3.2	~PluginT	795
10.163.4	Member Function Documentation	796
10.163.4.1	Create	796
10.163.4.2	GetFilename	796
10.163.4.3	GetHandle	796
10.163.4.4	GetType	796
10.163.5	Member Data Documentation	796
10.163.5.1	filename	796
10.163.5.2	handle	796
10.163.5.3	type	796
10.164	gazebo::math::Pose Class Reference	797
10.164.1	Detailed Description	798
10.164.2	Constructor & Destructor Documentation	799
10.164.2.1	Pose	799
10.164.2.2	Pose	799
10.164.2.3	Pose	799
10.164.2.4	Pose	799
10.164.2.5	~Pose	799
10.164.3	Member Function Documentation	799
10.164.3.1	CoordPoseSolve	799
10.164.3.2	CoordPositionAdd	800
10.164.3.3	CoordPositionAdd	800
10.164.3.4	CoordPositionSub	800
10.164.3.5	CoordRotationAdd	800
10.164.3.6	CoordRotationSub	801
10.164.3.7	Correct	801
10.164.3.8	GetInverse	801
10.164.3.9	IsFinite	801
10.164.3.10	operator!=	801
10.164.3.11	operator*	801
10.164.3.12	operator+	802
10.164.3.13	operator+=	802
10.164.3.14	operator-	802
10.164.3.15	operator-	802
10.164.3.16	operator-=	803
10.164.3.17	operator=	803

10.164.3.1	operator==	803
10.164.3.1	Reset	803
10.164.3.2	RotatePositionAboutOrigin	803
10.164.3.2	Round	804
10.164.3.2	Set	804
10.164.3.2	Set	804
10.164.3.2	Set	804
10.164.4	Friends And Related Function Documentation	804
10.164.4.1	operator<<	804
10.164.4.2	operator>>	805
10.164.5	Member Data Documentation	805
10.164.5.1	pos	805
10.164.5.2	rot	805
10.164.5.3	Zero	805
10.165	Gazebo::common::PoseAnimation Class Reference	805
10.165.1	Detailed Description	806
10.165.2	Constructor & Destructor Documentation	806
10.165.2.1	PoseAnimation	806
10.165.2.2	~PoseAnimation	807
10.165.3	Member Function Documentation	807
10.165.3.1	BuildInterpolationSplines	807
10.165.3.2	CreateKeyFrame	807
10.165.3.3	GetInterpolatedKeyFrame	807
10.165.3.4	GetInterpolatedKeyFrame	807
10.166	Gazebo::common::PoseKeyFrame Class Reference	808
10.166.1	Detailed Description	808
10.166.2	Constructor & Destructor Documentation	809
10.166.2.1	PoseKeyFrame	809
10.166.2.2	~PoseKeyFrame	809
10.166.3	Member Function Documentation	809
10.166.3.1	GetRotation	809
10.166.3.2	GetTranslation	809
10.166.3.3	SetRotation	809
10.166.3.4	SetTranslation	809
10.166.4	Member Data Documentation	810
10.166.4.1	rotate	810
10.166.4.2	translate	810

10.167	gazebo::rendering::Projector Class Reference	810
10.167.1	Detailed Description	811
10.167.2	Constructor & Destructor Documentation	811
10.167.2.1	Projector	811
10.167.2.2	~Projector	811
10.167.3	Member Function Documentation	811
10.167.3.1	GetParent	811
10.167.3.2	Load	811
10.167.3.3	Load	811
10.167.3.4	Load	812
10.167.3.5	SetEnabled	812
10.167.3.6	SetTexture	812
10.167.3.7	Toggle	812
10.168	gazebo::transport::Publication Class Reference	812
10.168.1	Detailed Description	813
10.168.2	Constructor & Destructor Documentation	814
10.168.2.1	Publication	814
10.168.2.2	~Publication	814
10.168.3	Member Function Documentation	814
10.168.3.1	AddPublisher	814
10.168.3.2	AddSubscription	814
10.168.3.3	AddSubscription	814
10.168.3.4	AddTransport	814
10.168.3.5	GetCallbackCount	815
10.168.3.6	GetLocallyAdvertised	815
10.168.3.7	GetMsgType	815
10.168.3.8	GetNodeCount	815
10.168.3.9	GetPrevMsg	815
10.168.3.10	GetRemoteSubscriptionCount	815
10.168.3.11	GetTransportCount	816
10.168.3.12	IsTransport	816
10.168.3.13	LocalPublish	816
10.168.3.14	Publish	816
10.168.3.15	RemovePublisher	816
10.168.3.16	RemoveSubscription	817
10.168.3.17	RemoveSubscription	817
10.168.3.18	RemoveTransport	817

10.168.3.1	SetLocallyAdvertised	817
10.168.3.2	SetPrevMsg	817
10.169	gazebo::transport::PublicationTransport Class Reference	818
10.169.1	Detailed Description	818
10.169.2	Constructor & Destructor Documentation	818
10.169.2.1	PublicationTransport	818
10.169.2.2	~PublicationTransport	819
10.169.3	Member Function Documentation	819
10.169.3.1	AddCallback	819
10.169.3.2	Finis	819
10.169.3.3	GetConnection	819
10.169.3.4	GetMsgType	819
10.169.3.5	GetTopic	819
10.169.3.6	dnit	819
10.170	gazebo::transport::Publisher Class Reference	820
10.170.1	Detailed Description	821
10.170.2	Constructor & Destructor Documentation	821
10.170.2.1	Publisher	821
10.170.2.2	~Publisher	821
10.170.3	Member Function Documentation	821
10.170.3.1	Finis	821
10.170.3.2	GetMsgType	821
10.170.3.3	GetOutgoingCount	821
10.170.3.4	GetPrevMsg	821
10.170.3.5	GetPrevMsgPtr	822
10.170.3.6	GetTopic	822
10.170.3.7	HasConnections	822
10.170.3.8	Publish	822
10.170.3.9	Publish	822
10.170.3.10	SendMessage	823
10.170.3.11	SetNode	823
10.170.3.12	SetPublication	823
10.170.3.13	WaitForConnection	823
10.170.3.14	WaitForConnection	823
10.171	gazebo::transport::PublishTask Class Reference	823
10.171.1	Detailed Description	824
10.171.2	Constructor & Destructor Documentation	824

10.171.2.1	PublishTask	824
10.171.3	Member Function Documentation	824
10.171.3.1	execute	824
10.172	Gazebo::math::Quaternion Class Reference	824
10.172.1	Detailed Description	827
10.172.2	Constructor & Destructor Documentation	827
10.172.2.1	Quaternion	827
10.172.2.2	Quaternion	827
10.172.2.3	Quaternion	827
10.172.2.4	Quaternion	828
10.172.2.5	Quaternion	828
10.172.2.6	Quaternion	828
10.172.2.7	~Quaternion	828
10.172.3	Member Function Documentation	828
10.172.3.1	Correct	828
10.172.3.2	Dot	828
10.172.3.3	EulerToQuaternion	829
10.172.3.4	EulerToQuaternion	829
10.172.3.5	GetAsAxis	829
10.172.3.6	GetAsEuler	829
10.172.3.7	GetAsMatrix3	829
10.172.3.8	GetAsMatrix4	830
10.172.3.9	GetExp	830
10.172.3.10	GetInverse	830
10.172.3.11	GetLog	830
10.172.3.12	GetPitch	830
10.172.3.13	GetRoll	830
10.172.3.14	GetXAxis	831
10.172.3.15	GetYaw	831
10.172.3.16	GetYAxis	831
10.172.3.17	GetZAxis	831
10.172.3.18	Invert	831
10.172.3.19	IsFinite	831
10.172.3.20	Normalize	831
10.172.3.21	operator!=	832
10.172.3.22	operator*	832
10.172.3.23	operator*	832

10.172.3.24	operator*	832
10.172.3.25	operator*= operator*	832
10.172.3.26	operator+ operator+	833
10.172.3.27	operator+=	833
10.172.3.28	operator- operator-	833
10.172.3.29	operator- operator-	833
10.172.3.30	operator-=	834
10.172.3.31	operator=	834
10.172.3.32	operator==	834
10.172.3.33	RotateVector	834
10.172.3.34	RotateVectorReverse	835
10.172.3.35	Round	835
10.172.3.36	Scale	835
10.172.3.37	Set	835
10.172.3.38	SetFromAxis	835
10.172.3.39	SetFromAxis	836
10.172.3.40	SetFromEuler	836
10.172.3.41	SetFromEuler	836
10.172.3.42	SetToIdentity	836
10.172.3.43	Slerp	836
10.172.3.44	Squad	836
10.172.4	Friends And Related Function Documentation	837
10.172.4.1	operator<<	837
10.172.4.2	operator>>	837
10.172.5	Member Data Documentation	837
10.172.5.1w		837
10.172.5.2x		837
10.172.5.3y		838
10.172.5.4z		838
10.173	Gazebo::math::Rand Class Reference	838
10.173.1	Detailed Description	838
10.173.2	Member Function Documentation	838
10.173.2.1	GetDbfNormal	838
10.173.2.2	GetDbfUniform	839
10.173.2.3	GetIntNormal	839
10.173.2.4	GetIntUniform	839
10.173.2.5	GetSeed	839

10.173.2.6	SetSeed	839
10.174	gazebo::transport::RawCallbackHelper Class Reference	840
10.174.1	Detailed Description	840
10.174.2	Constructor & Destructor Documentation	841
10.174.2.1	RawCallbackHelper	841
10.174.3	Member Function Documentation	841
10.174.3.1	GetMsgType	841
10.174.3.2	HandleData	841
10.174.3.3	HandleMessage	841
10.174.3.4	IsLocal	842
10.175	gazebo::sensors::RaySensor Class Reference	842
10.175.1	Detailed Description	844
10.175.2	Constructor & Destructor Documentation	844
10.175.2.1	RaySensor	844
10.175.2.2	~RaySensor	844
10.175.3	Member Function Documentation	844
10.175.3.1	Fini	844
10.175.3.2	GetAngleMax	844
10.175.3.3	GetAngleMin	844
10.175.3.4	GetAngleResolution	845
10.175.3.5	GetFiducial	845
10.175.3.6	GetLaserShape	845
10.175.3.7	GetRange	845
10.175.3.8	GetRangeCount	846
10.175.3.9	GetRangeMax	846
10.175.3.10	GetRangeMin	846
10.175.3.11	GetRangeResolution	846
10.175.3.12	GetRanges	846
10.175.3.13	GetRayCount	846
10.175.3.14	GetRetro	847
10.175.3.15	GetTopic	847
10.175.3.16	GetVerticalAngleMax	847
10.175.3.17	GetVerticalAngleMin	847
10.175.3.18	GetVerticalAngleResolution	848
10.175.3.19	GetVerticalRangeCount	848
10.175.3.20	GetVerticalRayCount	848
10.175.3.21	hit	848

10.175.3.23	Active	848
10.175.3.23	bad	848
10.175.3.24	updateImpl	849
10.176	Gazebo::physics::RayShape Class Reference	849
10.176.1	Detailed Description	851
10.176.2	Constructor & Destructor Documentation	851
10.176.2.1	RayShape	851
10.176.2.2	RayShape	851
10.176.2.3	~RayShape	851
10.176.3	Member Function Documentation	851
10.176.3.1	FillMsg	851
10.176.3.2	GetFiducial	852
10.176.3.3	GetGlobalPoints	852
10.176.3.4	GetIntersection	852
10.176.3.5	GetLength	852
10.176.3.6	GetRelativePoints	852
10.176.3.7	GetRetro	853
10.176.3.8	Init	853
10.176.3.9	ProcessMsg	853
10.176.3.10	SetFiducial	853
10.176.3.11	SetLength	853
10.176.3.12	SetPoints	853
10.176.3.13	SetRetro	854
10.176.3.14	SetScale	854
10.176.3.15	Update	854
10.176.4	Member Data Documentation	854
10.176.4.1	contactFiducial	854
10.176.4.2	contactLen	854
10.176.4.3	contactRetro	854
10.176.4.4	globalEndPos	854
10.176.4.5	globalStartPos	854
10.176.4.6	relativeEndPos	854
10.176.4.7	relativeStartPos	855
10.177	Gazebo::rendering::RenderEngine Class Reference	855
10.177.1	Detailed Description	856
10.177.2	Member Enumeration Documentation	856
10.177.2.1	RenderPathType	856

10.177.3	Member Function Documentation	857
10.177.3.1	AddResourcePath	857
10.177.3.2	CreateScene	857
10.177.3.3	Fini	857
10.177.3.4	GetRenderPathType	857
10.177.3.5	GetScene	857
10.177.3.6	GetScene	857
10.177.3.7	GetSceneCount	858
10.177.3.8	GetWindowManager	858
10.177.3.9	Init	858
10.177.3.10	Load	858
10.177.3.11	RemoveScene	858
10.177.4	Member Data Documentation	858
10.177.4.1	dummyContext	858
10.177.4.2	dummyDisplay	859
10.177.4.3	dummyWindowId	859
10.177.4.4	root	859
10.178	gazebo::sensors::RFIDSensor Class Reference	859
10.178.1	Detailed Description	860
10.178.2	Constructor & Destructor Documentation	860
10.178.2.1	RFIDSensor	860
10.178.2.2	~RFIDSensor	860
10.178.3	Member Function Documentation	860
10.178.3.1	AddTag	860
10.178.3.2	Fini	860
10.178.3.3	Init	860
10.178.3.4	Load	860
10.178.3.5	Load	861
10.178.3.6	UpdateImpl	861
10.179	gazebo::sensors::RFIDTag Class Reference	861
10.179.1	Detailed Description	862
10.179.2	Constructor & Destructor Documentation	863
10.179.2.1	RFIDTag	863
10.179.2.2	~RFIDTag	863
10.179.3	Member Function Documentation	863
10.179.3.1	Fini	863
10.179.3.2	GetTagPose	863

10.179.3.3	Init	863
10.179.3.4	Load	863
10.179.3.5	Load	863
10.179.3.6	UpdateImpl	864
10.180	gazebo::rendering::RFIDTagVisual Class Reference	864
10.180.1	Detailed Description	865
10.180.2	Constructor & Destructor Documentation	865
10.180.2.1	RFIDTagVisual	865
10.180.2.2	~RFIDTagVisual	865
10.181	gazebo::rendering::RFIDTagVisualPrivate Class Reference	865
10.181.1	Detailed Description	866
10.181.2	Member Data Documentation	866
10.181.2.1	node	866
10.181.2.2	fidSub	866
10.182	gazebo::rendering::RFIDVisual Class Reference	867
10.182.1	Detailed Description	867
10.182.2	Constructor & Destructor Documentation	867
10.182.2.1	RFIDVisual	867
10.182.2.2	~RFIDVisual	868
10.183	gazebo::rendering::RFIDVisualPrivate Class Reference	868
10.183.1	Detailed Description	868
10.183.2	Member Data Documentation	868
10.183.2.1	node	869
10.183.2.2	fidSub	869
10.184	gazebo::physics::Road Class Reference	869
10.184.1	Detailed Description	869
10.184.2	Constructor & Destructor Documentation	870
10.184.2.1	Road	870
10.184.2.2	~Road	870
10.184.3	Member Function Documentation	870
10.184.3.1	Init	870
10.184.3.2	Load	870
10.185	Road Class Reference	870
10.185.1	Detailed Description	870
10.186	gazebo::rendering::Road2d Class Reference	871
10.186.1	Constructor & Destructor Documentation	871
10.186.1.1	Road2d	871

10.186.1.2~Road2d	871
10.186.2 Member Function Documentation	871
10.186.2.1Load	871
10.187 gazebo::math::RotationSpline Class Reference	871
10.187.1 Detailed Description	872
10.187.2 Constructor & Destructor Documentation	872
10.187.2.1RotationSpline	872
10.187.2.2~RotationSpline	872
10.187.3 Member Function Documentation	873
10.187.3.1AddPoint	873
10.187.3.2Clear	873
10.187.3.3GetNumPoints	873
10.187.3.4GetPoint	873
10.187.3.5Interpolate	873
10.187.3.6Interpolate	874
10.187.3.7RecalcTangents	874
10.187.3.8SetAutoCalculate	874
10.187.3.9UpdatePoint	874
10.187.4 Member Data Documentation	875
10.187.4.1autoCalc	875
10.187.4.2points	875
10.187.4.3tangents	875
10.188 gazebo::rendering::RTShaderSystem Class Reference	875
10.188.1 Detailed Description	876
10.188.2 Member Enumeration Documentation	877
10.188.2.1LightingModel	877
10.188.3 Member Function Documentation	877
10.188.3.1AddScene	877
10.188.3.2ApplyShadows	877
10.188.3.3AttachEntity	877
10.188.3.4AttachViewport	877
10.188.3.5Clear	878
10.188.3.6DetachEntity	878
10.188.3.7DetachViewport	878
10.188.3.8Fini	878
10.188.3.9GenerateShaders	878
10.188.3.10GetPSSMShadowCameraSetup	878

10.188.3.11hit	878
10.188.3.12RemoveScene	879
10.188.3.13RemoveShadows	879
10.188.3.14SetPerPixelLighting	879
10.188.3.15UpdateShaders	879
10.189.0gazebo::rendering::Scene Class Reference	879
10.189.1Detailed Description	883
10.189.2Member Enumeration Documentation	883
10.189.2.1SkyXMode	883
10.189.3Constructor & Destructor Documentation	883
10.189.3.1Scene	883
10.189.3.2~Scene	883
10.189.4Member Function Documentation	883
10.189.4.1AddVisual	883
10.189.4.2Clear	883
10.189.4.3CloneVisual	884
10.189.4.4CreateCamera	884
10.189.4.5CreateDepthCamera	884
10.189.4.6CreateGpuLaser	884
10.189.4.7CreateGrid	885
10.189.4.8CreateUserCamera	885
10.189.4.9DrawLine	885
10.189.4.10GetAmbientColor	885
10.189.4.11GetBackgroundColor	886
10.189.4.12GetCamera	886
10.189.4.13GetCamera	886
10.189.4.14GetCameraCount	886
10.189.4.15GetFirstContact	886
10.189.4.16GetGrid	887
10.189.4.17GetGridCount	887
10.189.4.18GetHeightBelowPoint	887
10.189.4.19GetHeightmap	887
10.189.4.20GetId	887
10.189.4.21GetIdString	888
10.189.4.22GetInitialized	888
10.189.4.23GetLight	888
10.189.4.24GetLight	888

10.189.4.26	GetLightCount	888
10.189.4.26	GetManager	888
10.189.4.27	GetModelVisualAt	889
10.189.4.28	GetName	889
10.189.4.29	GetSelectedVisual	889
10.189.4.30	GetShadowsEnabled	889
10.189.4.30	GetShowClouds	889
10.189.4.32	GetSimTime	890
10.189.4.33	GetUserCamera	890
10.189.4.33	GetUserCameraCount	890
10.189.4.35	GetVisual	890
10.189.4.36	GetVisual	890
10.189.4.37	GetVisualAt	891
10.189.4.38	GetVisualAt	891
10.189.4.39	GetVisualBelow	891
10.189.4.40	GetVisualCount	892
10.189.4.40	GetVisualsBelowPoint	892
10.189.4.40	GetWorldVisual	892
10.189.4.40	Hit	892
10.189.4.44	Load	892
10.189.4.45	Load	892
10.189.4.46	PreRender	892
10.189.4.47	PrintSceneGraph	893
10.189.4.48	RemoveCamera	893
10.189.4.49	RemoveProjectors	893
10.189.4.50	RemoveVisual	893
10.189.4.53	SelectVisual	893
10.189.4.52	SetAmbientColor	893
10.189.4.53	SetBackgroundColor	893
10.189.4.53	SetFog	894
10.189.4.55	SetGrid	894
10.189.4.56	SetShadowsEnabled	894
10.189.4.57	SetSkyXMode	894
10.189.4.58	SetTransparent	894
10.189.4.59	SetVisible	895
10.189.4.60	SetWireframe	895
10.189.4.63	ShowClouds	895

10.189.4.62	ShowCollisions	895
10.189.4.63	ShowCOMs	895
10.189.4.64	ShowContacts	895
10.189.4.65	ShowJoints	896
10.189.4.66	SnapVisualToNearestBelow	896
10.189.4.67	StripSceneName	896
10.189.5	Member Data Documentation	896
10.189.5.1	skyx	896
10.190	Gazebo::physics::ScrewJoint< T > Class Template Reference	896
10.190.1	Detailed Description	897
10.190.2	Constructor & Destructor Documentation	897
10.190.2.1	ScrewJoint	897
10.190.2.2	~ScrewJoint	897
10.190.3	Member Function Documentation	897
10.190.3.1	GetAngleCount	898
10.190.3.2	GetThreadPitch	898
10.190.3.3	GetThreadPitch	898
10.190.3.4	Init	898
10.190.3.5	Load	898
10.190.3.6	SetThreadPitch	899
10.190.3.7	SetThreadPitch	899
10.190.4	Member Data Documentation	899
10.190.4.1	threadPitch	899
10.191	Gazebo::rendering::SelectionObj Class Reference	899
10.191.1	Detailed Description	901
10.191.2	Member Enumeration Documentation	901
10.191.2.1	SelectionMode	901
10.191.3	Constructor & Destructor Documentation	901
10.191.3.1	SelectionObj	901
10.191.3.2	~SelectionObj	902
10.191.4	Member Function Documentation	902
10.191.4.1	Attach	902
10.191.4.2	Detach	902
10.191.4.3	GetMode	902
10.191.4.4	GetState	902
10.191.4.5	Load	902
10.191.4.6	SetGlobal	902

10.191.4.7	SetMode	902
10.191.4.8	SetMode	903
10.191.4.9	SetState	903
10.191.4.10	SetState	903
10.191.4.11	UpdateSize	903
10.192	gazebo::rendering::SelectionObjPrivate Class Reference	903
10.192.1	Detailed Description	905
10.192.2	Member Data Documentation	905
10.192.2.1	maxScale	905
10.192.2.2	minScale	905
10.192.2.3	mode	905
10.192.2.4	rotVisual	905
10.192.2.5	rotXVisual	905
10.192.2.6	rotYVisual	906
10.192.2.7	rotZVisual	906
10.192.2.8	scaleVisual	906
10.192.2.9	scaleXVisual	906
10.192.2.10	scaleYVisual	906
10.192.2.11	scaleZVisual	906
10.192.2.12	selectedVis	906
10.192.2.13	state	906
10.192.2.14	transVisual	906
10.192.2.15	transXVisual	906
10.192.2.16	transYVisual	906
10.192.2.17	transZVisual	907
10.192.2.18	AxisMat	907
10.192.2.19	AxisMatOverlay	907
10.192.2.20	AxisMat	907
10.192.2.21	AxisMatOverlay	907
10.192.2.22	AxisMat	907
10.192.2.23	AxisMatOverlay	907
10.193	gazebo::sensors::Sensor Class Reference	907
10.193.1	Detailed Description	910
10.193.2	Constructor & Destructor Documentation	911
10.193.2.1	Sensor	911
10.193.2.2	~Sensor	911
10.193.3	Member Function Documentation	911

10.193.3.1ConnectUpdated	911
10.193.3.2DisconnectUpdated	911
10.193.3.3FillMsg	911
10.193.3.4Fini	912
10.193.3.5GetCategory	912
10.193.3.6GetId	912
10.193.3.7GetLastMeasurementTime	912
10.193.3.8GetLastUpdateTime	912
10.193.3.9GetName	913
10.193.3.10GetNoise	913
10.193.3.11GetParentId	913
10.193.3.12GetParentName	913
10.193.3.13GetPose	913
10.193.3.14GetScopedName	914
10.193.3.15GetTopic	914
10.193.3.16GetType	914
10.193.3.17GetUpdateRate	914
10.193.3.18GetVisualize	914
10.193.3.19GetWorldName	914
10.193.3.20Hit	915
10.193.3.21IsActive	915
10.193.3.22IsBad	915
10.193.3.23IsBad	915
10.193.3.24NeedsUpdate	916
10.193.3.25ResetLastUpdateTime	916
10.193.3.26SetActive	916
10.193.3.27SetParent	916
10.193.3.28SetParent	916
10.193.3.29SetUpdateRate	916
10.193.3.30Update	917
10.193.3.31UpdateImpl	917
10.193.4Member Data Documentation	917
10.193.4.1active	917
10.193.4.2connections	917
10.193.4.3lastMeasurementTime	917
10.193.4.4lastUpdateTime	917
10.193.4.5node	918

10.193.4.6	noises	918
10.193.4.7	parentId	918
10.193.4.8	parentName	918
10.193.4.9	plugins	918
10.193.4.10	pose	918
10.193.4.11	poseSub	918
10.193.4.12	scene	918
10.193.4.13	stf	918
10.193.4.14	updatePeriod	918
10.193.4.15	world	918
10.194	SensorFactor Class Reference	919
10.194.1	Detailed Description	919
10.195	gazebo::sensors::SensorFactory Class Reference	919
10.195.1	Member Function Documentation	919
10.195.1.1	GetSensorTypes	919
10.195.1.2	NewSensor	919
10.195.1.3	RegisterAll	920
10.195.1.4	RegisterSensor	920
10.196	gazebo::sensors::SensorManager Class Reference	920
10.196.1	Detailed Description	922
10.196.2	Member Function Documentation	922
10.196.2.1	CreateSensor	922
10.196.2.2	CreateSensor	922
10.196.2.3	Finis	922
10.196.2.4	GetSensor	922
10.196.2.5	GetSensors	923
10.196.2.6	GetSensorTypes	923
10.196.2.7	Init	923
10.196.2.8	RemoveSensor	923
10.196.2.9	RemoveSensors	923
10.196.2.10	ResetLastUpdateTimes	923
10.196.2.11	RunThreads	923
10.196.2.12	SensorsInitialized	924
10.196.2.13	Stop	924
10.196.2.14	Update	924
10.197	gazebo::SensorPlugin Class Reference	924
10.197.1	Detailed Description	925

10.197.2	Constructor & Destructor Documentation	925
10.197.2.1	SensorPlugin	925
10.197.2.2	~SensorPlugin	925
10.197.3	Member Function Documentation	925
10.197.3.1	Init	925
10.197.3.2	Load	925
10.197.3.3	Reset	926
10.198	Gazebo::Server Class Reference	926
10.198.1	Constructor & Destructor Documentation	926
10.198.1.1	Server	926
10.198.1.2	~Server	926
10.198.2	Member Function Documentation	927
10.198.2.1	Fini	927
10.198.2.2	GetInitialized	927
10.198.2.3	LoadFile	927
10.198.2.4	LoadString	927
10.198.2.5	ParseArgs	927
10.198.2.6	PreLoad	928
10.198.2.7	PrintUsage	928
10.198.2.8	Run	928
10.198.2.9	SetParams	928
10.198.2.10	Stop	928
10.199	Gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg Class Reference	928
10.199.1	Detailed Description	929
10.199.2	Member Function Documentation	929
10.199.2.1	defaultVpParams	929
10.199.2.2	generateFragmentProgram	929
10.199.2.3	generateVertexProgram	929
10.199.2.4	generateVertexProgramSource	929
10.199.2.5	generateVpDynamicShadows	929
10.199.2.6	generateVpDynamicShadowsParams	929
10.199.2.7	generateVpFooter	930
10.199.2.8	generateVpHeader	930
10.200	Gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL Class Reference	930
10.200.1	Detailed Description	931
10.200.2	Member Function Documentation	931
10.200.2.1	defaultVpParams	931

10.200.2.2	generateFpDynamicShadows	931
10.200.2.3	generateFpDynamicShadowsHelpers	931
10.200.2.4	generateFpDynamicShadowsParams	931
10.200.2.5	generateFpFooter	931
10.200.2.6	generateFpHeader	931
10.200.2.7	generateFpLayer	931
10.200.2.8	generateFragmentProgram	931
10.200.2.9	generateFragmentProgramSource	932
10.200.2.10	generateVertexProgram	932
10.200.2.11	generateVertexProgramSource	932
10.200.2.12	generateVpDynamicShadows	932
10.200.2.13	generateVpDynamicShadowsParams	932
10.200.2.14	generateVpFooter	932
10.200.2.15	generateVpHeader	932
10.200.2.16	updateParams	932
10.200.2.17	updateVpParams	932
10.201	gazebo::physics::Shape Class Reference	932
10.201.1	Detailed Description	934
10.201.2	Constructor & Destructor Documentation	934
10.201.2.1	Shape	934
10.201.2.2	~Shape	934
10.201.3	Member Function Documentation	934
10.201.3.1	FillMsg	934
10.201.3.2	GetScale	934
10.201.3.3	Init	935
10.201.3.4	ProcessMsg	935
10.201.3.5	SetScale	935
10.201.4	Member Data Documentation	935
10.201.4.1	collisionParent	935
10.201.4.2	scale	935
10.202	gazebo::physics::SimbodyBallJoint Class Reference	936
10.202.1	Detailed Description	937
10.202.2	Constructor & Destructor Documentation	937
10.202.2.1	SimbodyBallJoint	937
10.202.2.2	~SimbodyBallJoint	937
10.202.3	Member Function Documentation	938
10.202.3.1	GetAnchor	938

10.202.3.2	GetAngleImpl	938
10.202.3.3	GetAxis	938
10.202.3.4	GetGlobalAxis	938
10.202.3.5	GetHighStop	938
10.202.3.6	GetLowStop	939
10.202.3.7	GetMaxForce	939
10.202.3.8	GetVelocity	939
10.202.3.9	Load	940
10.202.3.10	SetAxis	940
10.202.3.11	SetForceImpl	940
10.202.3.12	SetHighStop	940
10.202.3.13	SetLowStop	941
10.202.3.14	SetMaxForce	941
10.202.3.15	SetVelocity	941
10.203	gazebo::physics::SimbodyBoxShape Class Reference	941
10.203.1	Detailed Description	942
10.203.2	Constructor & Destructor Documentation	942
10.203.2.1	SimbodyBoxShape	942
10.203.2.2	~SimbodyBoxShape	943
10.203.3	Member Function Documentation	943
10.203.3.1	SetSize	943
10.204	gazebo::physics::SimbodyCollision Class Reference	943
10.204.1	Detailed Description	945
10.204.2	Constructor & Destructor Documentation	945
10.204.2.1	SimbodyCollision	945
10.204.2.2	~SimbodyCollision	945
10.204.3	Member Function Documentation	945
10.204.3.1	GetBoundingBox	945
10.204.3.2	GetCollisionShape	945
10.204.3.3	Load	945
10.204.3.4	OnPoseChange	945
10.204.3.5	SetCategoryBits	946
10.204.3.6	SetCollideBits	946
10.204.3.7	SetCollisionShape	946
10.205	gazebo::physics::SimbodyCylinderShape Class Reference	946
10.205.1	Detailed Description	947
10.205.2	Constructor & Destructor Documentation	947

10.205.2.1	SimbodyCylinderShape	947
10.205.2.2	~SimbodyCylinderShape	948
10.205.3	Member Function Documentation	948
10.205.3.1	SetSize	948
10.206	gazebo::physics::SimbodyHeightmapShape Class Reference	948
10.206.1	Detailed Description	949
10.206.2	Constructor & Destructor Documentation	949
10.206.2.1	SimbodyHeightmapShape	949
10.206.2.2	~SimbodyHeightmapShape	950
10.206.3	Member Function Documentation	950
10.206.3.1	Init	950
10.207	gazebo::physics::SimbodyHinge2Joint Class Reference	950
10.207.1	Detailed Description	952
10.207.2	Constructor & Destructor Documentation	952
10.207.2.1	SimbodyHinge2Joint	952
10.207.2.2	~SimbodyHinge2Joint	952
10.207.3	Member Function Documentation	952
10.207.3.1	GetAnchor	952
10.207.3.2	GetAngleImpl	953
10.207.3.3	GetAxis	953
10.207.3.4	GetGlobalAxis	953
10.207.3.5	GetMaxForce	953
10.207.3.6	GetVelocity	953
10.207.3.7	Load	954
10.207.3.8	SetAxis	954
10.207.3.9	SetForceImpl	954
10.207.3.10	SetMaxForce	954
10.207.3.11	SetVelocity	955
10.208	gazebo::physics::SimbodyHingeJoint Class Reference	955
10.208.1	Detailed Description	957
10.208.2	Constructor & Destructor Documentation	957
10.208.2.1	SimbodyHingeJoint	957
10.208.2.2	~SimbodyHingeJoint	957
10.208.3	Member Function Documentation	957
10.208.3.1	GetAngleImpl	957
10.208.3.2	GetGlobalAxis	958
10.208.3.3	GetMaxForce	958

10.208.3.4	GetVelocity	958
10.208.3.5	Load	958
10.208.3.6	RestoreSimbodyState	959
10.208.3.7	SaveSimbodyState	959
10.208.3.8	SetAxis	959
10.208.3.9	SetForceImpl	959
10.208.3.10	SetMaxForce	959
10.208.3.11	SetVelocity	960
10.209	Gazebo::physics::SimbodyJoint Class Reference	960
10.209.1	Detailed Description	962
10.209.2	Constructor & Destructor Documentation	962
10.209.2.1	SimbodyJoint	962
10.209.2.2	~SimbodyJoint	963
10.209.3	Member Function Documentation	963
10.209.3.1	AreConnected	963
10.209.3.2	CacheForceTorque	963
10.209.3.3	Detach	963
10.209.3.4	GetAnchor	963
10.209.3.5	GetAttribute	964
10.209.3.6	GetForce	964
10.209.3.7	GetForceTorque	964
10.209.3.8	GetHighStop	965
10.209.3.9	GetJointLink	965
10.209.3.10	GetLinkForce	965
10.209.3.11	GetLinkTorque	965
10.209.3.12	GetLowStop	966
10.209.3.13	GetParam	966
10.209.3.14	Load	966
10.209.3.15	Reset	967
10.209.3.16	RestoreSimbodyState	967
10.209.3.17	SaveSimbodyState	967
10.209.3.18	SetAnchor	967
10.209.3.19	SetAttribute	967
10.209.3.20	SetAttribute	967
10.209.3.21	SetAxis	968
10.209.3.22	SetDamping	968
10.209.3.23	SetForce	968

10.209.3.23	SetForceImpl	968
10.209.3.24	SetHighStop	969
10.209.3.25	SetLowStop	969
10.209.3.26	SetParam	969
10.209.3.27	SetStiffness	970
10.209.3.28	SetStiffnessDamping	970
10.209.4	Member Data Documentation	970
10.209.4.1	constraint	970
10.209.4.2	damper	970
10.209.4.3	defxAB	970
10.209.4.4	isReversed	970
10.209.4.5	limitForce	971
10.209.4.6	mobod	971
10.209.4.7	mustBreakLoopHere	971
10.209.4.8	physicsInitialized	971
10.209.4.9	simbodyPhysics	971
10.209.4.10	spring	971
10.209.4.11	World	971
10.209.4.12	CB	971
10.209.4.13	PA	971
10.210	gazebo::physics::SimbodyLink Class Reference	972
10.210.1	Detailed Description	974
10.210.2	Constructor & Destructor Documentation	974
10.210.2.1	SimbodyLink	974
10.210.2.2	~SimbodyLink	974
10.210.3	Member Function Documentation	974
10.210.3.1	AddForce	974
10.210.3.2	AddForceAtRelativePosition	974
10.210.3.3	AddForceAtWorldPosition	975
10.210.3.4	AddRelativeForce	975
10.210.3.5	AddRelativeTorque	975
10.210.3.6	AddTorque	975
10.210.3.7	Fini	976
10.210.3.8	GetEffectiveMassProps	976
10.210.3.9	GetEnabled	976
10.210.3.10	GetGravityMode	976
10.210.3.11	GetMassProperties	976

10.210.3.10	GetWorldAngularVel	976
10.210.3.10	GetWorldCoGLinearVel	976
10.210.3.10	GetWorldForce	977
10.210.3.10	GetWorldLinearVel	977
10.210.3.10	GetWorldLinearVel	977
10.210.3.10	GetWorldTorque	977
10.210.3.10	Init	978
10.210.3.10	Load	978
10.210.3.20	OnPoseChange	978
10.210.3.20	RestoreSimbodyState	978
10.210.3.20	SaveSimbodyState	978
10.210.3.20	SetAngularDamping	978
10.210.3.20	SetAngularVel	978
10.210.3.20	SetAutoDisable	978
10.210.3.20	SetDirtyPose	979
10.210.3.20	SetEnabled	979
10.210.3.20	SetForce	979
10.210.3.20	SetGravityMode	979
10.210.3.30	SetLinearDamping	979
10.210.3.30	SetLinearVel	979
10.210.3.30	SetLinkStatic	980
10.210.3.30	SetSelfCollide	980
10.210.3.30	SetTorque	980
10.210.4	Member Data Documentation	980
10.210.4.1	masterMobod	980
10.210.4.2	mustBeBaseLink	980
10.210.4.3	physicsInitialized	980
10.210.4.4	slaveMobods	980
10.210.4.5	slaveWelds	981
10.210	gazebo::physics::SimbodyMeshShape Class Reference	981
10.211.1	Detailed Description	982
10.211.2	Constructor & Destructor Documentation	982
10.211.2.1	SimbodyMeshShape	982
10.211.2.2	~SimbodyMeshShape	982
10.211.3	Member Function Documentation	982
10.211.3.1	Init	982
10.211.3.2	Load	982

10.212	<code>gazebo::physics::SimbodyModel</code> Class Reference	982
10.212.1	Detailed Description	983
10.212.2	Constructor & Destructor Documentation	983
10.212.2.1	<code>SimbodyModel</code>	983
10.212.2.2	<code>~SimbodyModel</code>	984
10.212.3	Member Function Documentation	984
10.212.3.1	<code>Init</code>	984
10.212.3.2	<code>Load</code>	984
10.213	<code>gazebo::physics::SimbodyMultiRayShape</code> Class Reference	984
10.213.1	Detailed Description	985
10.213.2	Constructor & Destructor Documentation	986
10.213.2.1	<code>SimbodyMultiRayShape</code>	986
10.213.2.2	<code>~SimbodyMultiRayShape</code>	986
10.213.3	Member Function Documentation	986
10.213.3.1	<code>AddRay</code>	986
10.213.3.2	<code>UpdateRays</code>	986
10.214	<code>gazebo::physics::SimbodyPhysics</code> Class Reference	986
10.214.1	Detailed Description	989
10.214.2	Constructor & Destructor Documentation	989
10.214.2.1	<code>SimbodyPhysics</code>	989
10.214.2.2	<code>~SimbodyPhysics</code>	989
10.214.3	Member Function Documentation	989
10.214.3.1	<code>CreateCollision</code>	989
10.214.3.2	<code>CreateJoint</code>	989
10.214.3.3	<code>CreateLink</code>	990
10.214.3.4	<code>CreateModel</code>	990
10.214.3.5	<code>CreateShape</code>	990
10.214.3.6	<code>DebugPrint</code>	990
10.214.3.7	<code>Finis</code>	990
10.214.3.8	<code>GetDynamicsWorld</code>	991
10.214.3.9	<code>GetParam</code>	991
10.214.3.10	<code>GetPose</code>	991
10.214.3.11	<code>GetType</code>	991
10.214.3.12	<code>GetTypeString</code>	991
10.214.3.13	<code>GetTypeString</code>	992
10.214.3.14	<code>Init</code>	992
10.214.3.15	<code>InitForThread</code>	992

10.214.3.16	hitModel	992
10.214.3.17	load	992
10.214.3.18	onPhysicsMsg	992
10.214.3.19	onRequest	993
10.214.3.20	pose2Transform	993
10.214.3.21	quadToQuad	993
10.214.3.22	quadToQuad	993
10.214.3.23	reset	994
10.214.3.24	setGravity	994
10.214.3.25	setParam	994
10.214.3.26	setSeed	995
10.214.3.27	transform2Pose	995
10.214.3.28	updateCollision	996
10.214.3.29	updatePhysics	996
10.214.3.30	vec3ToVector3	996
10.214.3.31	vector3ToVec3	996
10.214.4	Member Data Documentation	996
10.214.4.1	contact	996
10.214.4.2	discreteForces	996
10.214.4.3	forces	997
10.214.4.4	gravity	997
10.214.4.5	integ	997
10.214.4.6	matter	997
10.214.4.7	simbodyPhysicsInitialized	997
10.214.4.8	simbodyPhysicsStepped	997
10.214.4.9	system	997
10.214.4.10	tracker	997
10.215	gazebo::physics::SimbodyPlaneShape Class Reference	997
10.215.1	Detailed Description	998
10.215.2	Constructor & Destructor Documentation	998
10.215.2.1	SimbodyPlaneShape	998
10.215.2.2	~SimbodyPlaneShape	999
10.215.3	Member Function Documentation	999
10.215.3.1	CreatePlane	999
10.215.3.2	SetAltitude	999
10.216	gazebo::physics::SimbodyRayShape Class Reference	999
10.216.1	Detailed Description	1000

10.216.2	Constructor & Destructor Documentation	1001
10.216.2.1	SimbodyRayShape	1001
10.216.2.2	SimbodyRayShape	1001
10.216.2.3	~SimbodyRayShape	1001
10.216.3	Member Function Documentation	1001
10.216.3.1	GetIntersection	1001
10.216.3.2	SetPoints	1001
10.216.3.3	Update	1002
10.217	gazebo::physics::SimbodyScrewJoint Class Reference	1002
10.217.1	Detailed Description	1004
10.217.2	Constructor & Destructor Documentation	1004
10.217.2.1	SimbodyScrewJoint	1004
10.217.2.2	~SimbodyScrewJoint	1004
10.217.3	Member Function Documentation	1004
10.217.3.1	GetAngleImpl	1004
10.217.3.2	GetAttribute	1004
10.217.3.3	GetGlobalAxis	1005
10.217.3.4	GetHighStop	1005
10.217.3.5	GetLowStop	1005
10.217.3.6	GetMaxForce	1005
10.217.3.7	GetParam	1006
10.217.3.8	GetThreadPitch	1006
10.217.3.9	GetThreadPitch	1006
10.217.3.10	GetVelocity	1007
10.217.3.11	load	1007
10.217.3.12	GetAttribute	1007
10.217.3.13	SetAxis	1007
10.217.3.14	SetForceImpl	1008
10.217.3.15	SetHighStop	1008
10.217.3.16	SetLowStop	1008
10.217.3.17	SetMaxForce	1008
10.217.3.18	SetParam	1009
10.217.3.19	SetThreadPitch	1009
10.217.3.20	SetThreadPitch	1009
10.217.3.21	SetVelocity	1009
10.218	gazebo::physics::SimbodySliderJoint Class Reference	1010
10.218.1	Detailed Description	1011

10.218.2	Constructor & Destructor Documentation	1011
10.218.2.1	SimbodySliderJoint	1011
10.218.2.2	~SimbodySliderJoint	1011
10.218.3	Member Function Documentation	1011
10.218.3.1	GetAngleImpl	1012
10.218.3.2	GetGlobalAxis	1012
10.218.3.3	GetMaxForce	1012
10.218.3.4	GetVelocity	1012
10.218.3.5	Load	1013
10.218.3.6	SetAxis	1013
10.218.3.7	SetForceImpl	1013
10.218.3.8	SetMaxForce	1013
10.218.3.9	SetVelocity	1014
10.219	Gazebo::physics::SimbodySphereShape Class Reference	1014
10.219.1	Detailed Description	1015
10.219.2	Constructor & Destructor Documentation	1015
10.219.2.1	SimbodySphereShape	1015
10.219.2.2	~SimbodySphereShape	1015
10.219.3	Member Function Documentation	1015
10.219.3.1	SetRadius	1015
10.220	Gazebo::physics::SimbodyUniversalJoint Class Reference	1016
10.220.1	Detailed Description	1017
10.220.2	Constructor & Destructor Documentation	1017
10.220.2.1	SimbodyUniversalJoint	1017
10.220.2.2	~SimbodyUniversalJoint	1017
10.220.3	Member Function Documentation	1017
10.220.3.1	GetAnchor	1018
10.220.3.2	GetAngleImpl	1018
10.220.3.3	GetAxis	1018
10.220.3.4	GetGlobalAxis	1018
10.220.3.5	GetMaxForce	1018
10.220.3.6	GetVelocity	1019
10.220.3.7	Load	1019
10.220.3.8	SetAxis	1019
10.220.3.9	SetForceImpl	1019
10.220.3.10	SetMaxForce	1020
10.220.3.11	SetVelocity	1020

10.220	<code>gazebo::sensors::SimTimeEvent</code> Class Reference	1020
10.221	1. Detailed Description	1021
10.221	2. Member Data Documentation	1021
10.221.2	1. <code>condition</code>	1021
10.221.2	2. <code>time</code>	1021
10.220	<code>gazebo::sensors::SimTimeEventHandler</code> Class Reference	1021
10.222	1. Detailed Description	1021
10.222	2. Constructor & Destructor Documentation	1021
10.222.2	1. <code>SimTimeEventHandler</code>	1021
10.222.2	2. <code>~SimTimeEventHandler</code>	1021
10.222	3. Member Function Documentation	1022
10.222.3	1. <code>AddRelativeEvent</code>	1022
10.223	<code>SingletonT< T ></code> Class Template Reference	1022
10.223	1. Detailed Description	1024
10.223	2. Constructor & Destructor Documentation	1024
10.223.2	1. <code>SingletonT</code>	1024
10.223.2	2. <code>~SingletonT</code>	1024
10.223	3. Member Function Documentation	1024
10.223.3	1. <code>Instance</code>	1024
10.224	<code>gazebo::common::Skeleton</code> Class Reference	1024
10.224	1. Detailed Description	1026
10.224	2. Constructor & Destructor Documentation	1026
10.224.2	1. <code>Skeleton</code>	1026
10.224.2	2. <code>~Skeleton</code>	1026
10.224.2	3. <code>~Skeleton</code>	1026
10.224	3. Member Function Documentation	1026
10.224.3	1. <code>AddAnimation</code>	1026
10.224.3	2. <code>AddVertNodeWeight</code>	1027
10.224.3	3. <code>BuildNodeMap</code>	1027
10.224.3	4. <code>GetAnimation</code>	1027
10.224.3	5. <code>GetBindShapeTransform</code>	1027
10.224.3	6. <code>GetNodeByHandle</code>	1027
10.224.3	7. <code>GetNodeById</code>	1027
10.224.3	8. <code>GetNodeByName</code>	1028
10.224.3	9. <code>GetNodes</code>	1028
10.224.3	10. <code>GetNumAnimations</code>	1028
10.224.3	11. <code>GetNumJoints</code>	1028

10.224.3.10	GetNumNodes	1028
10.224.3.11	GetNumVertNodeWeights	1029
10.224.3.12	GetRootNode	1029
10.224.3.13	GetVertNodeWeight	1029
10.224.3.14	PrintTransforms	1029
10.224.3.15	Scale	1029
10.224.3.16	SetBindShapeTransform	1029
10.224.3.17	SetNumVertAttached	1030
10.224.3.18	SetRootNode	1030
10.224.4	Member Data Documentation	1030
10.224.4.1	animations	1030
10.224.4.2	bindShapeTransform	1030
10.224.4.3	nodes	1030
10.224.4.4	drawNW	1030
10.224.4.5	root	1030
10.225	gazebo::common::SkeletonAnimation Class Reference	1031
10.225.1	Detailed Description	1032
10.225.2	Constructor & Destructor Documentation	1032
10.225.2.1	SkeletonAnimation	1032
10.225.2.2	~SkeletonAnimation	1032
10.225.3	Member Function Documentation	1032
10.225.3.1	AddKeyFrame	1032
10.225.3.2	AddKeyFrame	1032
10.225.3.3	GetLength	1032
10.225.3.4	GetName	1033
10.225.3.5	GetNodeCount	1033
10.225.3.6	GetNodePoseAt	1033
10.225.3.7	GetPoseAt	1033
10.225.3.8	GetPoseAtX	1034
10.225.3.9	HasNode	1034
10.225.3.10	Scale	1034
10.225.3.11	SetName	1034
10.225.4	Member Data Documentation	1034
10.225.4.1	animations	1034
10.225.4.2	length	1035
10.225.4.3	name	1035
10.226	gazebo::common::SkeletonNode Class Reference	1035

10.226.1	Detailed Description	1037
10.226.2	Member Enumeration Documentation	1037
10.226.2.1	SkeletonNodeType	1037
10.226.3	Constructor & Destructor Documentation	1037
10.226.3.1	SkeletonNode	1037
10.226.3.2	SkeletonNode	1038
10.226.3.3	~SkeletonNode	1038
10.226.4	Member Function Documentation	1038
10.226.4.1	AddChild	1038
10.226.4.2	AddRawTransform	1038
10.226.4.3	GetChild	1038
10.226.4.4	GetChildById	1038
10.226.4.5	GetChildByName	1039
10.226.4.6	GetChildCount	1039
10.226.4.7	GetHandle	1039
10.226.4.8	GetId	1039
10.226.4.9	GetInverseBindTransform	1039
10.226.4.10	GetModelTransform	1040
10.226.4.11	GetName	1040
10.226.4.12	GetNumRawTrans	1040
10.226.4.13	GetParent	1040
10.226.4.14	GetRawTransform	1040
10.226.4.15	GetRawTransforms	1040
10.226.4.16	GetTransform	1041
10.226.4.17	GetTransforms	1041
10.226.4.18	Joint	1041
10.226.4.19	RootNode	1041
10.226.4.20	Reset	1041
10.226.4.21	SetHandle	1041
10.226.4.22	SetId	1042
10.226.4.23	SetInitialTransform	1042
10.226.4.24	SetInverseBindTransform	1042
10.226.4.25	SetModelTransform	1042
10.226.4.26	SetName	1042
10.226.4.27	SetParent	1042
10.226.4.28	SetTransform	1043
10.226.4.29	SetType	1043

10.226.4.3	UpdateChildrenTransforms	1043
10.226.5	Member Data Documentation	1043
10.226.5.1	children	1043
10.226.5.2	handle	1043
10.226.5.3	d	1043
10.226.5.4	initialTransform	1043
10.226.5.5	invBindTransform	1043
10.226.5.6	modelTransform	1043
10.226.5.7	name	1044
10.226.5.8	parent	1044
10.226.5.9	rawTransforms	1044
10.226.5.10	ransform	1044
10.226.5.11	type	1044
10.227	Gazebo::physics::SliderJoint< T > Class Template Reference	1044
10.227.1	Detailed Description	1044
10.227.2	Constructor & Destructor Documentation	1045
10.227.2.1	SliderJoint	1045
10.227.2.2	~SliderJoint	1045
10.227.3	Member Function Documentation	1045
10.227.3.1	GetAngleCount	1045
10.227.3.2	Load	1045
10.228	Gazebo::rendering::GzTerrainMatGen::SM2Profile Class Reference	1045
10.228.1	Detailed Description	1046
10.228.2	Constructor & Destructor Documentation	1046
10.228.2.1	SM2Profile	1046
10.228.2.2	~SM2Profile	1046
10.228.3	Member Function Documentation	1046
10.228.3.1	addTechnique	1046
10.228.3.2	generate	1046
10.228.3.3	generateForCompositeMap	1046
10.228.3.4	updateParams	1046
10.228.3.5	updateParamsForCompositeMap	1046
10.229	Gazebo::sensors::SonarSensor Class Reference	1047
10.229.1	Detailed Description	1048
10.229.2	Constructor & Destructor Documentation	1048
10.229.2.1	SonarSensor	1048
10.229.2.2	~SonarSensor	1048

10.229.3	Member Function Documentation	1048
10.229.3.1	ConnectUpdate	1048
10.229.3.2	DisconnectUpdate	1048
10.229.3.3	Finis	1049
10.229.3.4	GetRadius	1049
10.229.3.5	GetRange	1049
10.229.3.6	GetRangeMax	1049
10.229.3.7	GetRangeMin	1049
10.229.3.8	GetTopic	1050
10.229.3.9	init	1050
10.229.3.10	IsActive	1050
10.229.3.11	load	1050
10.229.3.12	updateImpl	1050
10.229.4	Member Data Documentation	1051
10.229.4.1	update	1051
10.230	gazebo::rendering::SonarVisual Class Reference	1051
10.230.1	Detailed Description	1052
10.230.2	Constructor & Destructor Documentation	1052
10.230.2.1	SonarVisual	1052
10.230.2.2	~SonarVisual	1052
10.230.3	Member Function Documentation	1052
10.230.3.1	Load	1052
10.231	gazebo::rendering::SonarVisualPrivate Class Reference	1052
10.231.1	Detailed Description	1053
10.231.2	Member Data Documentation	1054
10.231.2.1	coneNode	1054
10.231.2.2	connections	1054
10.231.2.3	mutex	1054
10.231.2.4	node	1054
10.231.2.5	receivedMsg	1054
10.231.2.6	sonarMsg	1054
10.231.2.7	sonarRay	1054
10.231.2.8	sonarSub	1054
10.232	gazebo::physics::SphereShape Class Reference	1054
10.232.1	Detailed Description	1055
10.232.2	Constructor & Destructor Documentation	1056
10.232.2.1	SphereShape	1056

10.232.2.2~SphereShape	1056
10.232.3 Member Function Documentation	1056
10.232.3.1FillMsg	1056
10.232.3.2GetRadius	1056
10.232.3.3Init	1056
10.232.3.4ProcessMsg	1056
10.232.3.5SetRadius	1057
10.232.3.6SetScale	1057
10.233 gazebo::common::SphericalCoordinates Class Reference	1057
10.233.1 Detailed Description	1058
10.233.2 Member Enumeration Documentation	1058
10.233.2.1SurfaceType	1058
10.233.3 Constructor & Destructor Documentation	1059
10.233.3.1SphericalCoordinates	1059
10.233.3.2SphericalCoordinates	1059
10.233.3.3SphericalCoordinates	1059
10.233.3.4~SphericalCoordinates	1059
10.233.4 Member Function Documentation	1059
10.233.4.1Convert	1059
10.233.4.2Distance	1059
10.233.4.3GetElevationReference	1060
10.233.4.4GetHeadingOffset	1060
10.233.4.5GetLatitudeReference	1060
10.233.4.6GetLongitudeReference	1060
10.233.4.7GetSurfaceType	1061
10.233.4.8GlobalFromLocal	1061
10.233.4.9SetElevationReference	1061
10.233.4.10SetHeadingOffset	1061
10.233.4.11SetLatitudeReference	1061
10.233.4.12SetLongitudeReference	1061
10.233.4.13SetSurfaceType	1062
10.233.4.14SphericalFromLocal	1062
10.234 gazebo::common::SphericalCoordinatesPrivate Class Reference	1062
10.234.1 Detailed Description	1063
10.234.2 Member Data Documentation	1063
10.234.2.1elevationReference	1063
10.234.2.2headingOffset	1063

10.234.2.3	attitudeReference	1063
10.234.2.4	longitudeReference	1063
10.234.2.5	surfaceType	1063
10.235	gazebo::math::Spline Class Reference	1063
10.235.1	Detailed Description	1064
10.235.2	Constructor & Destructor Documentation	1064
10.235.2.1	Spline	1064
10.235.2.2	~Spline	1064
10.235.3	Member Function Documentation	1065
10.235.3.1	AddPoint	1065
10.235.3.2	Clear	1065
10.235.3.3	GetPoint	1065
10.235.3.4	GetPointCount	1065
10.235.3.5	GetTangent	1065
10.235.3.6	GetTension	1065
10.235.3.7	Interpolate	1066
10.235.3.8	Interpolate	1066
10.235.3.9	RecalcTangents	1066
10.235.3.10	SetAutoCalculate	1066
10.235.3.11	SetTension	1067
10.235.3.12	UpdatePoint	1067
10.235.4	Member Data Documentation	1067
10.235.4.1	autoCalc	1067
10.235.4.2	coeffs	1067
10.235.4.3	points	1067
10.235.4.4	tangents	1067
10.235.4.5	tension	1067
10.236	gazebo::physics::State Class Reference	1068
10.236.1	Detailed Description	1069
10.236.2	Constructor & Destructor Documentation	1069
10.236.2.1	State	1069
10.236.2.2	State	1069
10.236.2.3	~State	1069
10.236.3	Member Function Documentation	1070
10.236.3.1	GetName	1070
10.236.3.2	GetRealTime	1070
10.236.3.3	GetSimTime	1070

10.236.3.4	GetWallTime	1070
10.236.3.5	Load	1070
10.236.3.6	operator-	1071
10.236.3.7	operator=	1071
10.236.3.8	SetName	1071
10.236.3.9	SetRealTime	1071
10.236.3.10	SetSimTime	1071
10.236.3.11	SetWallTime	1072
10.236.4	Member Data Documentation	1072
10.236.4.1	name	1072
10.236.4.2	realTime	1072
10.236.4.3	simTime	1072
10.236.4.4	wallTime	1072
10.237	gazebo::common::STLLoader Class Reference	1072
10.237.1	Detailed Description	1073
10.237.2	Constructor & Destructor Documentation	1073
10.237.2.1	STLLoader	1073
10.237.2.2	~STLLoader	1073
10.237.3	Member Function Documentation	1073
10.237.3.1	Load	1073
10.238	gazebo::common::SubMesh Class Reference	1074
10.238.1	Detailed Description	1076
10.238.2	Member Enumeration Documentation	1076
10.238.2.1	PrimitiveType	1076
10.238.3	Constructor & Destructor Documentation	1076
10.238.3.1	SubMesh	1076
10.238.3.2	SubMesh	1077
10.238.3.3	~SubMesh	1077
10.238.4	Member Function Documentation	1077
10.238.4.1	AddIndex	1077
10.238.4.2	AddNodeAssignment	1077
10.238.4.3	AddNormal	1077
10.238.4.4	AddNormal	1077
10.238.4.5	AddTexCoord	1078
10.238.4.6	AddVertex	1078
10.238.4.7	AddVertex	1078
10.238.4.8	Center	1078

10.238.4.9	CopyNormals	1078
10.238.4.10	CopyVertices	1078
10.238.4.11	FillArrays	1079
10.238.4.12	GenSphericalTexCoord	1079
10.238.4.13	GetIndex	1079
10.238.4.14	GetIndexCount	1079
10.238.4.15	GetMaterialIndex	1079
10.238.4.16	GetMax	1079
10.238.4.17	GetMaxIndex	1079
10.238.4.18	GetMin	1080
10.238.4.19	GetName	1080
10.238.4.20	GetNodeAssignment	1080
10.238.4.21	GetNodeAssignmentsCount	1080
10.238.4.22	GetNormal	1080
10.238.4.23	GetNormalCount	1080
10.238.4.24	GetPrimitiveType	1080
10.238.4.25	GetTexCoord	1081
10.238.4.26	GetTexCoordCount	1081
10.238.4.27	GetVertex	1081
10.238.4.28	GetVertexCount	1081
10.238.4.29	GetVertexIndex	1081
10.238.4.30	HasVertex	1081
10.238.4.31	RecalculateNormals	1082
10.238.4.32	Scale	1082
10.238.4.33	SetIndexCount	1082
10.238.4.34	SetMaterialIndex	1082
10.238.4.35	SetName	1082
10.238.4.36	SetNormal	1082
10.238.4.37	SetNormalCount	1083
10.238.4.38	SetPrimitiveType	1083
10.238.4.39	SetScale	1083
10.238.4.40	SetSubMeshCenter	1083
10.238.4.41	SetTexCoord	1083
10.238.4.42	SetTexCoordCount	1084
10.238.4.43	SetVertex	1084
10.238.4.44	SetVertexCount	1084
10.238.4.45	Translate	1084

10.239	<code>gazebo::transport::SubscribeOptions</code> Class Reference	1084
10.239.1	Detailed Description	1085
10.239.2	Constructor & Destructor Documentation	1085
10.239.2.1	<code>SubscribeOptions</code>	1085
10.239.3	Member Function Documentation	1085
10.239.3.1	<code>GetLatching</code>	1085
10.239.3.2	<code>GetMsgType</code>	1085
10.239.3.3	<code>GetNode</code>	1085
10.239.3.4	<code>GetTopic</code>	1086
10.239.3.5	<code>init</code>	1086
10.239.3.6	<code>init</code>	1086
10.240	<code>gazebo::transport::Subscriber</code> Class Reference	1086
10.240.1	Detailed Description	1087
10.240.2	Constructor & Destructor Documentation	1087
10.240.2.1	<code>Subscriber</code>	1087
10.240.2.2	<code>~Subscriber</code>	1087
10.240.3	Member Function Documentation	1087
10.240.3.1	<code>GetCallbackId</code>	1087
10.240.3.2	<code>GetTopic</code>	1087
10.240.3.3	<code>SetCallbackId</code>	1087
10.240.3.4	<code>Unsubscribe</code>	1087
10.241	<code>gazebo::transport::SubscriptionTransport</code> Class Reference	1088
10.241.1	Detailed Description	1088
10.241.2	Constructor & Destructor Documentation	1089
10.241.2.1	<code>SubscriptionTransport</code>	1089
10.241.2.2	<code>~SubscriptionTransport</code>	1089
10.241.3	Member Function Documentation	1089
10.241.3.1	<code>GetConnection</code>	1089
10.241.3.2	<code>HandleData</code>	1089
10.241.3.3	<code>HandleMessage</code>	1089
10.241.3.4	<code>init</code>	1090
10.241.3.5	<code>isLocal</code>	1090
10.242	<code>gazebo::physics::SurfaceParams</code> Class Reference	1090
10.242.1	Detailed Description	1091
10.242.2	Constructor & Destructor Documentation	1091
10.242.2.1	<code>SurfaceParams</code>	1091
10.242.2.2	<code>~SurfaceParams</code>	1091

10.242.3	Member Function Documentation	1091
10.242.3.1	FillMsg	1091
10.242.3.2	Load	1091
10.242.3.3	ProcessMsg	1091
10.242.4	Member Data Documentation	1092
10.242.4.1	collideWithoutContact	1092
10.242.4.2	collideWithoutContactBitmask	1092
10.243	Gazebo::common::SystemPaths Class Reference	1092
10.243.1	Detailed Description	1094
10.243.2	Member Function Documentation	1094
10.243.2.1	AddGazeboPaths	1094
10.243.2.2	AddModelPaths	1094
10.243.2.3	AddOgrePaths	1094
10.243.2.4	AddPluginPaths	1094
10.243.2.5	AddSearchPathSuffix	1094
10.243.2.6	ClearGazeboPaths	1095
10.243.2.7	ClearModelPaths	1095
10.243.2.8	ClearOgrePaths	1095
10.243.2.9	ClearPluginPaths	1095
10.243.2.10	FindFile	1095
10.243.2.11	FindFileURI	1095
10.243.2.12	GetDefaultTestPath	1096
10.243.2.13	GetGazeboPaths	1096
10.243.2.14	GetLogPath	1096
10.243.2.15	GetModelPaths	1096
10.243.2.16	GetOgrePaths	1096
10.243.2.17	GetPluginPaths	1096
10.243.2.18	GetTmpInstancePath	1097
10.243.2.19	GetTmpPath	1097
10.243.2.20	GetWorldPathExtension	1097
10.243.3	Member Data Documentation	1097
10.243.3.1	gazeboPathsFromEnv	1097
10.243.3.2	modelPathsFromEnv	1097
10.243.3.3	ogrePathsFromEnv	1097
10.243.3.4	pluginPathsFromEnv	1097
10.244	Gazebo::SystemPlugin Class Reference	1098
10.244.1	Detailed Description	1098

10.244.2	Constructor & Destructor Documentation	1098
10.244.2.1	SystemPlugin	1098
10.244.2.2	~SystemPlugin	1099
10.244.3	Member Function Documentation	1099
10.244.3.1	Init	1099
10.244.3.2	Load	1099
10.244.3.3	Reset	1099
10.245	Gazebo::common::Time Class Reference	1099
10.245.1	Detailed Description	1103
10.245.2	Constructor & Destructor Documentation	1103
10.245.2.1	Time	1103
10.245.2.2	Time	1103
10.245.2.3	Time	1103
10.245.2.4	Time	1104
10.245.2.5	Time	1104
10.245.2.6	Time	1104
10.245.2.7	~Time	1104
10.245.3	Member Function Documentation	1104
10.245.3.1	Double	1104
10.245.3.2	Float	1104
10.245.3.3	GetWallTime	1105
10.245.3.4	GetWallTimeAsISOString	1105
10.245.3.5	MicToNano	1105
10.245.3.6	MilToNano	1105
10.245.3.7	MSleep	1105
10.245.3.8	NSleep	1106
10.245.3.9	operator!=	1106
10.245.3.10	operator!=	1106
10.245.3.11	operator!=	1106
10.245.3.12	operator!=	1107
10.245.3.13	operator*	1107
10.245.3.14	operator*	1107
10.245.3.15	operator*	1107
10.245.3.16	operator*=	1108
10.245.3.17	operator*=	1108
10.245.3.18	operator*=	1108
10.245.3.19	operator+	1108

10.245.3.20	operator+	1109
10.245.3.21	operator+	1109
10.245.3.22	operator+=	1109
10.245.3.23	operator+=	1109
10.245.3.24	operator+=	1110
10.245.3.25	operator-	1110
10.245.3.26	operator-	1110
10.245.3.27	operator-	1110
10.245.3.28	operator-=	1111
10.245.3.29	operator-=	1111
10.245.3.30	operator-=	1111
10.245.3.31	operator/	1111
10.245.3.32	operator/	1112
10.245.3.33	operator/	1112
10.245.3.34	operator/=	1112
10.245.3.35	operator/=	1112
10.245.3.36	operator/=	1113
10.245.3.37	operator<	1113
10.245.3.38	operator<	1113
10.245.3.39	operator<	1113
10.245.3.40	operator<	1114
10.245.3.41	operator<=	1114
10.245.3.42	operator<=	1114
10.245.3.43	operator<=	1114
10.245.3.44	operator<=	1115
10.245.3.45	operator=	1115
10.245.3.46	operator=	1115
10.245.3.47	operator=	1115
10.245.3.48	operator==	1116
10.245.3.49	operator==	1116
10.245.3.50	operator==	1116
10.245.3.51	operator==	1116
10.245.3.52	operator>	1117
10.245.3.53	operator>	1117
10.245.3.54	operator>	1117
10.245.3.55	operator>	1117
10.245.3.56	operator>=	1118

10.245.3.57	operator>=	1118
10.245.3.58	operator>=	1118
10.245.3.59	operator>=	1118
10.245.3.60	SecToNano	1119
10.245.3.61	Set	1119
10.245.3.62	Set	1119
10.245.3.63	SetToWallTime	1119
10.245.3.64	sleep	1119
10.245.4	Friends And Related Function Documentation	1120
10.245.4.1	operator<<	1120
10.245.4.2	operator>>	1120
10.245.5	Member Data Documentation	1120
10.245.5.1	Insec	1120
10.245.5.2	Sec	1120
10.245.5.3	Zero	1120
10.246	gazebo::common::Timer Class Reference	1120
10.246.1	Detailed Description	1121
10.246.2	Constructor & Destructor Documentation	1121
10.246.2.1	Timer	1121
10.246.2.2	~Timer	1122
10.246.3	Member Function Documentation	1122
10.246.3.1	GetElapsed	1122
10.246.3.2	GetRunning	1122
10.246.3.3	Start	1122
10.246.3.4	Stop	1122
10.246.4	Friends And Related Function Documentation	1122
10.246.4.1	operator<<	1122
10.247	gazebo::transport::TopicManager Class Reference	1122
10.247.1	Detailed Description	1124
10.247.2	Member Typedef Documentation	1124
10.247.2.1	SubNodeMap	1124
10.247.3	Member Function Documentation	1124
10.247.3.1	AddNode	1124
10.247.3.2	AddNodeToProcess	1125
10.247.3.3	Advertise	1125
10.247.3.4	ClearBuffers	1125
10.247.3.5	ConnectPubToSub	1125

10.247.3.6	ConnectSubscribers	1125
10.247.3.7	ConnectSubToPub	1126
10.247.3.8	DisconnectPubFromSub	1126
10.247.3.9	DisconnectSubFromPub	1126
10.247.3.10	FindPublication	1126
10.247.3.11	Init	1126
10.247.3.12	GetTopicNamespaces	1127
10.247.3.13	Init	1127
10.247.3.14	Advertised	1127
10.247.3.15	PauseIncoming	1127
10.247.3.16	ProcessNodes	1127
10.247.3.17	Publish	1127
10.247.3.18	RegisterTopicNamespace	1128
10.247.3.19	RemoveNode	1128
10.247.3.20	Subscribe	1128
10.247.3.21	Unadvertise	1128
10.247.3.22	Unadvertise	1128
10.247.3.23	Unsubscribe	1129
10.247.3.24	UpdatePublications	1129
10.248	gazebo::physics::TrajectoryInfo Class Reference	1129
10.248.1	Detailed Description	1130
10.248.2	Constructor & Destructor Documentation	1130
10.248.2.1	TrajectoryInfo	1130
10.248.3	Member Data Documentation	1130
10.248.3.1	duration	1130
10.248.3.2	endTime	1130
10.248.3.3	id	1130
10.248.3.4	startTime	1130
10.248.3.5	translated	1130
10.248.3.6	type	1131
10.249	gazebo::rendering::TransmitterVisual Class Reference	1131
10.249.1	Detailed Description	1131
10.249.2	Constructor & Destructor Documentation	1132
10.249.2.1	TransmitterVisual	1132
10.249.2.2	~TransmitterVisual	1132
10.249.3	Member Function Documentation	1132
10.249.3.1	Load	1132

10.249.3.2Update	1132
10.250gazebo::rendering::TransmitterVisualPrivate Class Reference	1132
10.250.1Detailed Description	1133
10.250.2Member Data Documentation	1134
10.250.2.1connections	1134
10.250.2.2gridMsg	1134
10.250.2.3sFirst	1134
10.250.2.4mutex	1134
10.250.2.5node	1134
10.250.2.6points	1134
10.250.2.7receivedMsg	1134
10.250.2.8signalPropagationSub	1134
10.250.2.9vectorLink	1134
10.251gazebo::physics::UniversalJoint< T > Class Template Reference	1135
10.251.1Detailed Description	1135
10.251.2Member Enumeration Documentation	1135
10.251.2.1AxisIndex	1135
10.251.3Constructor & Destructor Documentation	1136
10.251.3.1UniversalJoint	1136
10.251.3.2~UniversalJoint	1136
10.251.4Member Function Documentation	1136
10.251.4.1GetAngleCount	1136
10.251.4.2Init	1136
10.251.4.3Load	1136
10.252gazebo::common::UpdateInfo Class Reference	1136
10.252.1Detailed Description	1137
10.252.2Member Data Documentation	1137
10.252.2.1realTime	1137
10.252.2.2simTime	1137
10.252.2.3worldName	1137
10.253gazebo::rendering::UserCamera Class Reference	1137
10.253.1Detailed Description	1140
10.253.2Constructor & Destructor Documentation	1140
10.253.2.1UserCamera	1140
10.253.2.2~UserCamera	1140
10.253.3Member Function Documentation	1140
10.253.3.1AnimationComplete	1140

10.253.3.2	AttachToVisualImpl	1140
10.253.3.3	EnableViewController	1141
10.253.3.4	Finis	1141
10.253.3.5	GetAvgFPS	1141
10.253.3.6	GetGUIOverlay	1141
10.253.3.7	GetImageHeight	1141
10.253.3.8	GetImageWidth	1141
10.253.3.9	GetTriangleCount	1142
10.253.3.10	GetViewControllerTypeString	1142
10.253.3.11	GetVisual	1142
10.253.3.12	GetVisual	1142
10.253.3.13	HandleKeyPressEvent	1142
10.253.3.14	HandleKeyReleaseEvent	1143
10.253.3.15	HandleMouseEvent	1143
10.253.3.16	Init	1143
10.253.3.17	CameraSetInWorldFile	1143
10.253.3.18	Load	1143
10.253.3.19	Load	1143
10.253.3.20	MoveToPosition	1143
10.253.3.21	MoveToVisual	1144
10.253.3.22	MoveToVisual	1144
10.253.3.23	PostRender	1144
10.253.3.24	Resize	1144
10.253.3.25	SetFocalPoint	1144
10.253.3.26	SetRenderTarget	1145
10.253.3.27	SetUseSDFPose	1145
10.253.3.28	SetViewController	1145
10.253.3.29	SetViewController	1145
10.253.3.30	SetViewportDimensions	1145
10.253.3.31	SetWorldPose	1146
10.253.3.32	TrackVisualImpl	1146
10.253.3.33	Update	1146
10.254	gazebo::rendering::UserCameraPrivate Class Reference	1146
10.254.1	Detailed Description	1147
10.254.2	Member Data Documentation	1147
10.254.2.1	ffpsViewController	1147
10.254.2.2	gui	1147

10.254.2.3	CameraSetInWorldFile	1147
10.254.2.4	OrbitViewController	1147
10.254.2.5	SelectionBuffer	1147
10.254.2.6	ViewController	1147
10.255	Gazebo::math::Vector2d Class Reference	1147
10.255.1	Detailed Description	1149
10.255.2	Constructor & Destructor Documentation	1149
10.255.2.1	Vector2d	1149
10.255.2.2	Vector2d	1149
10.255.2.3	Vector2d	1149
10.255.2.4	~Vector2d	1150
10.255.3	Member Function Documentation	1150
10.255.3.1	Cross	1150
10.255.3.2	Distance	1150
10.255.3.3	IsFinite	1150
10.255.3.4	Normalize	1150
10.255.3.5	operator!=	1150
10.255.3.6	operator*	1151
10.255.3.7	operator*	1151
10.255.3.8	operator*=	1151
10.255.3.9	operator*=	1151
10.255.3.10	operator+	1152
10.255.3.11	operator+=	1152
10.255.3.12	operator-	1152
10.255.3.13	operator-=	1152
10.255.3.14	operator/	1153
10.255.3.15	operator/	1153
10.255.3.16	operator/=	1153
10.255.3.17	operator/=	1154
10.255.3.18	operator=	1154
10.255.3.19	operator=	1154
10.255.3.20	operator==	1154
10.255.3.21	operator[]	1155
10.255.3.22	Set	1155
10.255.4	Friends And Related Function Documentation	1155
10.255.4.1	operator<<	1155
10.255.4.2	operator>>	1155

10.255.5	Member Data Documentation	1155
10.255.5.1x	1156
10.255.5.2y	1156
10.256	gazebo::math::Vector2i Class Reference	1156
10.256.1	Detailed Description	1157
10.256.2	Constructor & Destructor Documentation	1158
10.256.2.1	Vector2i	1158
10.256.2.2	Vector2i	1158
10.256.2.3	Vector2i	1158
10.256.2.4	~Vector2i	1158
10.256.3	Member Function Documentation	1158
10.256.3.1	Cross	1158
10.256.3.2	Distance	1158
10.256.3.3	IsFinite	1159
10.256.3.4	Normalize	1159
10.256.3.5	operator!=	1159
10.256.3.6	operator*	1159
10.256.3.7	operator*	1159
10.256.3.8	operator*=	1160
10.256.3.9	operator*=	1160
10.256.3.10	operator+	1160
10.256.3.11	operator+=	1161
10.256.3.12	operator-	1161
10.256.3.13	operator-=	1161
10.256.3.14	operator/	1161
10.256.3.15	operator/	1162
10.256.3.16	operator/=	1162
10.256.3.17	operator/=	1162
10.256.3.18	operator=	1163
10.256.3.19	operator=	1163
10.256.3.20	operator==	1163
10.256.3.21	operator[]	1163
10.256.3.22	Set	1164
10.256.4	Friends And Related Function Documentation	1164
10.256.4.1	operator<<	1164
10.256.4.2	operator>>	1164
10.256.5	Member Data Documentation	1164

10.256.5.1x	1164
10.256.5.2y	1164
10.257 gazebo::math::Vector3 Class Reference	1165
10.257.1 Detailed Description	1167
10.257.2 Constructor & Destructor Documentation	1167
10.257.2.1 Vector3	1167
10.257.2.2 Vector3	1168
10.257.2.3 Vector3	1168
10.257.2.4 ~Vector3	1168
10.257.3 Member Function Documentation	1168
10.257.3.1 Correct	1168
10.257.3.2 Cross	1168
10.257.3.3 Distance	1168
10.257.3.4 Distance	1169
10.257.3.5 Dot	1169
10.257.3.6 Equal	1169
10.257.3.7 GetAbs	1169
10.257.3.8 GetDistToLine	1169
10.257.3.9 GetLength	1170
10.257.3.10 GetMax	1170
10.257.3.11 GetMin	1170
10.257.3.12 GetNormal	1170
10.257.3.13 GetPerpendicular	1170
10.257.3.14 GetRounded	1171
10.257.3.15 GetSquaredLength	1171
10.257.3.16 GetSum	1171
10.257.3.17 Finite	1171
10.257.3.18 Normalize	1171
10.257.3.19 operator!=	1171
10.257.3.20 operator*	1172
10.257.3.21 operator*	1172
10.257.3.22 operator*=	1172
10.257.3.23 operator*=	1172
10.257.3.24 operator+	1173
10.257.3.25 operator+=	1173
10.257.3.26 operator-	1173
10.257.3.27 operator-	1173

10.257.3.28	operator-=	1174
10.257.3.29	operator/	1174
10.257.3.30	operator/	1174
10.257.3.31	operator/=	1174
10.257.3.32	operator/=	1175
10.257.3.33	operator=	1175
10.257.3.34	operator=	1175
10.257.3.35	operator==	1175
10.257.3.36	operator[]	1176
10.257.3.37	bound	1176
10.257.3.38	bound	1176
10.257.3.39	set	1176
10.257.3.40	setToMax	1176
10.257.3.41	setToMin	1176
10.257.4	Friends And Related Function Documentation	1176
10.257.4.1	operator*	1177
10.257.4.2	operator<<	1177
10.257.4.3	operator>>	1177
10.257.5	Member Data Documentation	1177
10.257.5.1	One	1177
10.257.5.2	UnitX	1177
10.257.5.3	UnitY	1178
10.257.5.4	UnitZ	1178
10.257.5.5	x	1178
10.257.5.6	y	1178
10.257.5.7	z	1178
10.257.5.8	Zero	1178
10.258	Gazebo::math::Vector4 Class Reference	1178
10.258.1	Detailed Description	1180
10.258.2	Constructor & Destructor Documentation	1180
10.258.2.1	Vector4	1180
10.258.2.2	Vector4	1180
10.258.2.3	Vector4	1181
10.258.2.4	~Vector4	1181
10.258.3	Member Function Documentation	1181
10.258.3.1	Distance	1181
10.258.3.2	GetLength	1181

10.258.3.3	GetSquaredLength	1181
10.258.3.4	isFinite	1181
10.258.3.5	Normalize	1181
10.258.3.6	operator!=	1182
10.258.3.7	operator*	1182
10.258.3.8	operator*	1182
10.258.3.9	operator*	1182
10.258.3.10	operator*=	1183
10.258.3.11	operator*=	1183
10.258.3.12	operator+	1183
10.258.3.13	operator+=	1183
10.258.3.14	operator-	1184
10.258.3.15	operator-=	1184
10.258.3.16	operator/	1184
10.258.3.17	operator/	1185
10.258.3.18	operator/=	1185
10.258.3.19	operator/=	1185
10.258.3.20	operator=	1185
10.258.3.21	operator=	1186
10.258.3.22	operator==	1186
10.258.3.23	operator[]	1186
10.258.3.24	set	1186
10.258.4	Friends And Related Function Documentation	1186
10.258.4.1	operator<<	1187
10.258.4.2	operator>>	1187
10.258.5	Member Data Documentation	1187
10.258.5.1	w	1187
10.258.5.2	x	1187
10.258.5.3	y	1187
10.258.5.4	z	1187
10.259	gazebo::common::Video Class Reference	1188
10.259.1	Detailed Description	1188
10.259.2	Constructor & Destructor Documentation	1188
10.259.2.1	Video	1188
10.259.2.2	~Video	1188
10.259.3	Member Function Documentation	1188
10.259.3.1	GetHeight	1188

10.259.3.2	getNextFrame	1189
10.259.3.3	getWidth	1189
10.259.3.4	Load	1189
10.260	gazebo::rendering::VideoVisual Class Reference	1189
10.260.1	Detailed Description	1190
10.260.2	Constructor & Destructor Documentation	1190
10.260.2.1	VideoVisual	1190
10.260.2.2	~VideoVisual	1190
10.261	gazebo::rendering::VideoVisualPrivate Class Reference	1191
10.261.1	Detailed Description	1191
10.261.2	Member Data Documentation	1191
10.261.2.1	connections	1192
10.261.2.2	height	1192
10.261.2.3	imageBuffer	1192
10.261.2.4	texture	1192
10.261.2.5	video	1192
10.261.2.6	width	1192
10.262	gazebo::rendering::ViewController Class Reference	1192
10.262.1	Detailed Description	1194
10.262.2	Constructor & Destructor Documentation	1194
10.262.2.1	ViewController	1194
10.262.2.2	~ViewController	1194
10.262.3	Member Function Documentation	1194
10.262.3.1	GetTypeString	1194
10.262.3.2	HandleKeyPressEvent	1194
10.262.3.3	HandleKeyReleaseEvent	1194
10.262.3.4	HandleMouseEvent	1195
10.262.3.5	init	1195
10.262.3.6	init	1195
10.262.3.7	setEnabled	1195
10.262.3.8	update	1195
10.262.4	Member Data Documentation	1195
10.262.4.1	camera	1196
10.262.4.2	enabled	1196
10.262.4.3	typeString	1196
10.263	gazebo::rendering::Visual Class Reference	1196
10.263.1	Detailed Description	1202

10.263.2	Constructor & Destructor Documentation	1202
10.263.2.1	Visual	1202
10.263.2.2	Visual	1202
10.263.2.3	Visual	1202
10.263.2.4	Visual	1202
10.263.2.5	Visual	1202
10.263.3	Member Function Documentation	1202
10.263.3.1	AttachAxes	1203
10.263.3.2	AttachLineVertex	1203
10.263.3.3	AttachMesh	1203
10.263.3.4	AttachObject	1203
10.263.3.5	AttachVisual	1203
10.263.3.6	ClearParent	1203
10.263.3.7	Clone	1203
10.263.3.8	CreateDynamicLine	1204
10.263.3.9	DeleteDynamicLine	1204
10.263.3.10	DetachObjects	1204
10.263.3.11	DetachVisual	1204
10.263.3.12	DetachVisual	1204
10.263.3.13	DisableTrackVisual	1205
10.263.3.14	EnableTrackVisual	1205
10.263.3.15	End	1205
10.263.3.16	GetAttachedObjectCount	1205
10.263.3.17	GetBoundingBox	1205
10.263.3.18	GetChild	1205
10.263.3.19	GetChildCount	1205
10.263.3.20	GetHighlighted	1206
10.263.3.21	GetId	1206
10.263.3.22	GetMaterialName	1206
10.263.3.23	GetMeshName	1206
10.263.3.24	GetName	1206
10.263.3.25	GetNormalMap	1206
10.263.3.26	GetParent	1207
10.263.3.27	GetPose	1207
10.263.3.28	GetPosition	1207
10.263.3.29	GetRootVisual	1207
10.263.3.30	GetRotation	1207

10.263.3.30	GetScale	1207
10.263.3.30	GetScene	1208
10.263.3.33	GetSceneNode	1208
10.263.3.34	GetShaderType	1208
10.263.3.35	GetSubMeshName	1208
10.263.3.36	GetTransparency	1208
10.263.3.37	GetVisibilityFlags	1208
10.263.3.38	GetVisible	1209
10.263.3.39	GetWorldPose	1209
10.263.3.40	HasAttachedObject	1209
10.263.3.41	hit	1209
10.263.3.42	InsertMesh	1209
10.263.3.43	InsertMesh	1209
10.263.3.44	Plane	1210
10.263.3.45	Static	1210
10.263.3.46	Load	1210
10.263.3.47	Load	1210
10.263.3.48	LoadFromMsg	1210
10.263.3.49	LoadPlugin	1210
10.263.3.50	MakeStatic	1211
10.263.3.51	MoveToPosition	1211
10.263.3.52	MoveToPositions	1211
10.263.3.53	RemovePlugin	1211
10.263.3.54	SetAmbient	1211
10.263.3.55	SetCastShadows	1212
10.263.3.56	SetDiffuse	1212
10.263.3.57	SetEmissive	1212
10.263.3.58	SetHighlighted	1212
10.263.3.59	SetId	1212
10.263.3.60	SetLighting	1212
10.263.3.61	SetMaterial	1213
10.263.3.62	SetName	1213
10.263.3.63	SetNormalMap	1213
10.263.3.64	SetPose	1213
10.263.3.65	SetPosition	1213
10.263.3.66	SetRibbonTrail	1213
10.263.3.67	SetRotation	1214

10.263.3.68	SetScale	1214
10.263.3.69	SetScene	1214
10.263.3.70	SetShaderType	1214
10.263.3.71	SetSkeletonPose	1214
10.263.3.72	SetSpecular	1215
10.263.3.73	SetTransparency	1215
10.263.3.74	SetVisibilityFlags	1215
10.263.3.75	SetVisible	1215
10.263.3.76	SetWireframe	1215
10.263.3.77	SetWorldPose	1216
10.263.3.78	SetWorldPosition	1216
10.263.3.79	SetWorldRotation	1216
10.263.3.80	ShowBoundingBox	1216
10.263.3.81	ShowCollision	1216
10.263.3.82	ShowCOM	1216
10.263.3.83	ShowJoints	1216
10.263.3.84	ShowSkeleton	1217
10.263.3.85	ToggleVisible	1217
10.263.3.86	Update	1217
10.263.3.87	UpdateFromMsg	1217
10.263.4	Member Data Documentation	1217
10.263.4.1	dataPtr	1217
10.264	gazebo::VisualPlugin Class Reference	1217
10.264.1	Detailed Description	1218
10.264.2	Constructor & Destructor Documentation	1218
10.264.2.1	VisualPlugin	1218
10.264.3	Member Function Documentation	1218
10.264.3.1	Init	1218
10.264.3.2	Load	1219
10.264.3.3	Reset	1219
10.265	gazebo::rendering::VisualPrivate Class Reference	1219
10.265.1	Detailed Description	1222
10.265.2	Member Data Documentation	1222
10.265.2.1	animState	1222
10.265.2.2	boundingBox	1222
10.265.2.3	children	1222
10.265.2.4	d	1222

10.265.2.5	initialized	1222
10.265.2.6	isStatic	1222
10.265.2.7	lighting	1222
10.265.2.8	lines	1223
10.265.2.9	lineVertices	1223
10.265.2.10	materialName	1223
10.265.2.11	name	1223
10.265.2.12	onAnimationComplete	1223
10.265.2.13	origMaterialName	1223
10.265.2.14	parent	1223
10.265.2.15	plugins	1223
10.265.2.16	preRenderConnection	1223
10.265.2.17	prevAnimTime	1223
10.265.2.18	roboTrail	1223
10.265.2.19	scale	1224
10.265.2.20	scene	1224
10.265.2.21	sceneNode	1224
10.265.2.22	surf	1224
10.265.2.23	skeleton	1224
10.265.2.24	staticGeom	1224
10.265.2.25	transparency	1224
10.265.2.26	useRTShader	1224
10.265.2.27	visible	1224
10.265.2.28	visualIdCount	1224
10.266	Gazebo::rendering::WindowManager Class Reference	1225
10.266.1	Detailed Description	1225
10.266.2	Constructor & Destructor Documentation	1225
10.266.2.1	WindowManager	1225
10.266.2.2	~WindowManager	1225
10.266.3	Member Function Documentation	1226
10.266.3.1	CreateWindow	1226
10.266.3.2	Fini	1226
10.266.3.3	GetAvgFPS	1226
10.266.3.4	GetTriangleCount	1226
10.266.3.5	GetWindow	1226
10.266.3.6	Moved	1227
10.266.3.7	Resize	1227

10.266.3.8SetCamera	1227
10.267 gazebo::rendering::WireBox Class Reference	1227
10.267.1 Detailed Description	1228
10.267.2 Constructor & Destructor Documentation	1228
10.267.2.1 WireBox	1228
10.267.2.2 ~WireBox	1228
10.267.3 Member Function Documentation	1228
10.267.3.1 GetBox	1228
10.267.3.2 GetVisible	1228
10.267.3.3 Init	1228
10.267.3.4 SetVisible	1229
10.268 gazebo::rendering::WireBoxPrivate Class Reference	1229
10.268.1 Detailed Description	1229
10.268.2 Member Data Documentation	1229
10.268.2.1 lines	1229
10.268.2.2 parent	1229
10.269 gazebo::sensors::WirelessReceiver Class Reference	1230
10.269.1 Detailed Description	1231
10.269.2 Constructor & Destructor Documentation	1231
10.269.2.1 WirelessReceiver	1231
10.269.2.2 ~WirelessReceiver	1231
10.269.3 Member Function Documentation	1231
10.269.3.1 Fini	1231
10.269.3.2 GetMaxFreqFiltered	1231
10.269.3.3 GetMinFreqFiltered	1231
10.269.3.4 GetSensitivity	1231
10.269.3.5 Init	1232
10.269.3.6 Load	1232
10.270 gazebo::sensors::WirelessTransceiver Class Reference	1232
10.270.1 Detailed Description	1233
10.270.2 Constructor & Destructor Documentation	1233
10.270.2.1 WirelessTransceiver	1233
10.270.2.2 ~WirelessTransceiver	1233
10.270.3 Member Function Documentation	1234
10.270.3.1 Fini	1234
10.270.3.2 GetGain	1234
10.270.3.3 GetPower	1234

10.270.3.4	GetTopic	1234
10.270.3.5	Init	1234
10.270.3.6	Load	1234
10.270.4	Member Data Documentation	1235
10.270.4.1	gain	1235
10.270.4.2	parentEntity	1235
10.270.4.3	power	1235
10.270.4.4	pub	1235
10.270.4.5	referencePose	1235
10.271	gazebo::sensors::WirelessTransmitter Class Reference	1235
10.271.1	Detailed Description	1237
10.271.2	Constructor & Destructor Documentation	1237
10.271.2.1	WirelessTransmitter	1237
10.271.2.2	~WirelessTransmitter	1237
10.271.3	Member Function Documentation	1237
10.271.3.1	GetESSID	1237
10.271.3.2	GetFreq	1237
10.271.3.3	GetSignalStrength	1237
10.271.3.4	Init	1238
10.271.3.5	Load	1238
10.271.3.6	UpdateImpl	1238
10.271.4	Member Data Documentation	1238
10.271.4.1	freq	1238
10.271.4.2	ModelStdDesv	1238
10.271.4.3	NEmpy	1239
10.271.4.4	NObstacle	1239
10.272	gazebo::physics::World Class Reference	1239
10.272.1	Detailed Description	1241
10.272.2	Constructor & Destructor Documentation	1242
10.272.2.1	World	1242
10.272.2.2	~World	1242
10.272.3	Member Function Documentation	1242
10.272.3.1	Clear	1242
10.272.3.2	ClearModels	1242
10.272.3.3	DisableAllModels	1242
10.272.3.4	EnableAllModels	1242
10.272.3.5	EnablePhysicsEngine	1242

10.272.3.6	Fin	1243
10.272.3.7	GetByName	1243
10.272.3.8	GetEnablePhysicsEngine	1243
10.272.3.9	GetEntity	1243
10.272.3.10	GetEntityBelowPoint	1243
10.272.3.10	GetIterations	1244
10.272.3.10	GetModel	1244
10.272.3.10	GetModel	1244
10.272.3.10	GetModelBelowPoint	1244
10.272.3.10	GetModelCount	1245
10.272.3.10	GetModels	1245
10.272.3.10	GetName	1245
10.272.3.10	GetPauseTime	1245
10.272.3.10	GetPhysicsEngine	1245
10.272.3.20	GetRealTime	1246
10.272.3.20	GetRunning	1246
10.272.3.20	GetSceneMsg	1246
10.272.3.20	GetSelectedEntity	1246
10.272.3.20	GetSetWorldPoseMutex	1246
10.272.3.20	GetSimTime	1246
10.272.3.20	GetSphericalCoordinates	1247
10.272.3.20	GetStartTime	1247
10.272.3.20	Get	1247
10.272.3.20	InsertModelFile	1247
10.272.3.30	InsertModelSDF	1247
10.272.3.30	InsertModelString	1247
10.272.3.30	Loaded	1248
10.272.3.30	Paused	1248
10.272.3.30	Load	1248
10.272.3.30	LoadPlugin	1248
10.272.3.30	PrintEntityTree	1248
10.272.3.30	PublishModelPose	1248
10.272.3.30	RemovePlugin	1249
10.272.3.30	Reset	1249
10.272.3.40	ResetEntities	1249
10.272.3.40	ResetTime	1249
10.272.3.40	Run	1249

10.272.3.4	RunBlocking	1249
10.272.3.4	Save	1250
10.272.3.4	SetPaused	1250
10.272.3.4	SetSimTime	1250
10.272.3.4	SetState	1250
10.272.3.4	Step	1250
10.272.3.4	StepWorld	1250
10.272.3.5	Stop	1251
10.272.3.5	StripWorldName	1251
10.272.3.5	UpdateStateSDF	1251
10.272.4	Member Data Documentation	1251
10.272.4.1	dirtyPoses	1251
10.273	Gazebo::WorldPlugin Class Reference	1251
10.273.1	Detailed Description	1252
10.273.2	Constructor & Destructor Documentation	1252
10.273.2.1	WorldPlugin	1252
10.273.2.2	~WorldPlugin	1252
10.273.3	Member Function Documentation	1253
10.273.3.1	Init	1253
10.273.3.2	Load	1253
10.273.3.3	Reset	1253
10.274	Gazebo::physics::WorldState Class Reference	1253
10.274.1	Detailed Description	1254
10.274.2	Constructor & Destructor Documentation	1255
10.274.2.1	WorldState	1255
10.274.2.2	WorldState	1255
10.274.2.3	WorldState	1255
10.274.2.4	~WorldState	1255
10.274.3	Member Function Documentation	1255
10.274.3.1	FillSDF	1255
10.274.3.2	GetModelState	1255
10.274.3.3	GetModelStateCount	1256
10.274.3.4	GetModelStates	1256
10.274.3.5	GetModelStates	1256
10.274.3.6	HasModelState	1256
10.274.3.7	IsZero	1257
10.274.3.8	Load	1257

10.274.3.9	Load	1257
10.274.3.10	operator+	1257
10.274.3.11	operator-	1257
10.274.3.12	operator=	1258
10.274.3.13	SetRealTime	1258
10.274.3.14	SetSimTime	1258
10.274.3.15	SetWallTime	1258
10.274.3.16	SetWorld	1258
10.274.4	Friends And Related Function Documentation	1259
10.274.4.1	operator<<	1259
10.275	Gazebo::rendering::WrenchVisual Class Reference	1259
10.275.1	Detailed Description	1260
10.275.2	Constructor & Destructor Documentation	1260
10.275.2.1	WrenchVisual	1260
10.275.2.2	~WrenchVisual	1260
10.275.3	Member Function Documentation	1260
10.275.3.1	Load	1260
10.275.3.2	SetEnabled	1260
10.276	Gazebo::rendering::WrenchVisualPrivate Class Reference	1261
10.276.1	Detailed Description	1262
10.276.2	Member Data Documentation	1262
10.276.2.1	coneXNode	1262
10.276.2.2	coneYNode	1262
10.276.2.3	coneZNode	1262
10.276.2.4	connections	1262
10.276.2.5	enabled	1262
10.276.2.6	forceLine	1262
10.276.2.7	forceNode	1262
10.276.2.8	mutex	1263
10.276.2.9	node	1263
10.276.2.10	receivedMsg	1263
10.276.2.11	wrenchMsg	1263
10.276.2.12	wrenchSub	1263
11	File Documentation	1265
11.1	Actor.hh File Reference	1265
11.2	Angle.hh File Reference	1266

11.2.1 Macro Definition Documentation	1268
11.2.1.1 GZ_DTOR	1268
11.2.1.2 GZ_NORMALIZE	1268
11.2.1.3 GZ_RTOD	1268
11.3 Animation.hh File Reference	1269
11.4 ArrowVisual.hh File Reference	1270
11.5 ArrowVisualPrivate.hh File Reference	1271
11.6 Assert.hh File Reference	1271
11.6.1 Macro Definition Documentation	1272
11.6.1.1 GZ_ASSERT	1272
11.7 AudioDecoder.hh File Reference	1272
11.8 AxisVisual.hh File Reference	1274
11.9 AxisVisualPrivate.hh File Reference	1274
11.10BallJoint.hh File Reference	1275
11.11Base.hh File Reference	1276
11.12Base64.hh File Reference	1277
11.12.1 Function Documentation	1278
11.12.1.1 Base64Decode	1278
11.12.1.2 Base64Encode	1278
11.13Box.hh File Reference	1278
11.14BoxShape.hh File Reference	1279
11.15BVHLoader.hh File Reference	1280
11.15.1 Macro Definition Documentation	1282
11.15.1.1 X_POSITION	1282
11.15.1.2 X_ROTATION	1282
11.15.1.3 Y_POSITION	1282
11.15.1.4 Y_ROTATION	1282
11.15.1.5 Z_POSITION	1282
11.15.1.6 Z_ROTATION	1282
11.16CallbackHelper.hh File Reference	1282
11.17Camera.hh File Reference	1283
11.18CameraPrivate.hh File Reference	1285
11.19CameraSensor.hh File Reference	1285
11.20CameraVisual.hh File Reference	1286
11.21CameraVisualPrivate.hh File Reference	1287
11.22cegui.h File Reference	1287
11.23ColladaLoader.hh File Reference	1288

11.24Collision.hh File Reference 1289

11.25CollisionState.hh File Reference 1290

11.26Color.hh File Reference 1291

11.27Commonface.hh File Reference 1292

11.28CommonTypes.hh File Reference 1294

 11.28.1 Detailed Description 1295

 11.28.2 Macro Definition Documentation 1295

 11.28.2.1 GAZEBO_DEPRECATED 1295

 11.28.2.2 GAZEBO_FORCEINLINE 1295

 11.28.2.3 NULL 1295

11.29COMVisual.hh File Reference 1296

11.30COMVisualPrivate.hh File Reference 1296

11.31Connection.hh File Reference 1297

 11.31.1 Macro Definition Documentation 1299

 11.31.1.1 HEADER_LENGTH 1299

11.32ConnectionManager.hh File Reference 1299

11.33Console.hh File Reference 1300

11.34Contact.hh File Reference 1301

 11.34.1 Macro Definition Documentation 1302

 11.34.1.1 MAX_COLLIDE_RETURNS 1302

 11.34.1.2 MAX_CONTACT_JOINTS 1302

11.35ContactManager.hh File Reference 1303

11.36ContactSensor.hh File Reference 1304

11.37ContactVisual.hh File Reference 1304

11.38ContactVisualPrivate.hh File Reference 1305

11.39Conversions.hh File Reference 1306

11.40CylinderShape.hh File Reference 1307

11.41dart_inc.h File Reference 1308

11.42DARTBallJoint.hh File Reference 1308

11.43DARTBoxShape.hh File Reference 1309

11.44DARTCollision.hh File Reference 1310

11.45DARTCylinderShape.hh File Reference 1311

11.46DARTHeightmapShape.hh File Reference 1311

11.47DARTHinge2Joint.hh File Reference 1312

11.48DARTHingeJoint.hh File Reference 1312

11.49DARTJoint.hh File Reference 1313

11.50DARTLink.hh File Reference 1313

11.51DARTMeshShape.hh File Reference	1314
11.52DARTModel.hh File Reference	1315
11.53DARTMultiRayShape.hh File Reference	1315
11.54DARTPhysics.hh File Reference	1316
11.55DARTPlaneShape.hh File Reference	1316
11.56DARTRayShape.hh File Reference	1317
11.57DARTScrewJoint.hh File Reference	1318
11.58DARTSliderJoint.hh File Reference	1318
11.59DARTSphereShape.hh File Reference	1319
11.60DARTTypes.hh File Reference	1319
11.60.1 Detailed Description	1320
11.61DARTUniversalJoint.hh File Reference	1320
11.62Dem.hh File Reference	1321
11.63DemPrivate.hh File Reference	1322
11.64DepthCamera.hh File Reference	1323
11.65DepthCameraSensor.hh File Reference	1324
11.66Diagnostics.hh File Reference	1324
11.67DynamicLines.hh File Reference	1325
11.68DynamicRenderable.hh File Reference	1326
11.69Entity.hh File Reference	1327
11.70Event.hh File Reference	1328
11.71Events.hh File Reference	1329
11.72Exception.hh File Reference	1330
11.73ffmpeg_inc.h File Reference	1332
11.74ForceTorqueSensor.hh File Reference	1332
11.75FPSViewController.hh File Reference	1333
11.76GaussianNoiseModel.hh File Reference	1334
11.77gazebo.hh File Reference	1334
11.78gazebo_core.hh File Reference	1336
11.79GazeboGenerator.hh File Reference	1337
11.80GearboxJoint.hh File Reference	1338
11.81GpsSensor.hh File Reference	1339
11.82GpuLaser.hh File Reference	1339
11.83GpuRaySensor.hh File Reference	1340
11.84Grid.hh File Reference	1341
11.85Gripper.hh File Reference	1341
11.86GUIOverlay.hh File Reference	1342

11.87	GUIOverlayPrivate.hh File Reference	1343
11.88	Heightmap.hh File Reference	1344
11.89	HeightmapData.hh File Reference	1345
11.90	HeightmapShape.hh File Reference	1346
11.91	Helpers.hh File Reference	1347
11.91.1	Macro Definition Documentation	1349
11.91.1.1	GZ_DBL_MAX	1349
11.91.1.2	GZ_DBL_MIN	1349
11.91.1.3	GZ_FLT_MAX	1349
11.91.1.4	GZ_FLT_MIN	1349
11.91.1.5	GZ_INT32_MAX	1349
11.91.1.6	GZ_INT32_MIN	1349
11.91.1.7	GZ_UINT32_MAX	1349
11.91.1.8	GZ_UINT32_MIN	1350
11.92	Hinge2Joint.hh File Reference	1350
11.93	HingeJoint.hh File Reference	1351
11.94	Image.hh File Reference	1352
11.95	ImageHeightmap.hh File Reference	1352
11.96	ImuSensor.hh File Reference	1354
11.97	Inertial.hh File Reference	1354
11.98	IOManager.hh File Reference	1355
11.99	Joint.hh File Reference	1356
11.99.1	Macro Definition Documentation	1357
11.99.1.1	MAX_JOINT_AXIS	1357
11.100	JointController.hh File Reference	1358
11.101	JointControllerPrivate.hh File Reference	1359
11.102	JointState.hh File Reference	1359
11.103	JointVisual.hh File Reference	1360
11.104	JointVisualPrivate.hh File Reference	1361
11.105	JointWrench.hh File Reference	1362
11.106	KeyEvent.hh File Reference	1363
11.107	KeyFrame.hh File Reference	1364
11.108	LaserVisual.hh File Reference	1365
11.109	LaserVisualPrivate.hh File Reference	1366
11.110	Light.hh File Reference	1367
11.111	Link.hh File Reference	1367
11.112	LinkState.hh File Reference	1369

11.113	logPlay.hh File Reference	1370
11.114	logRecord.hh File Reference	1370
11.114.1	Macro Definition Documentation	1371
11.114.1.1	IGZ_LOG_VERSION	1371
11.115	mainpage.html File Reference	1371
11.116	MapShape.hh File Reference	1371
11.117	Master.hh File Reference	1372
11.118	Material.hh File Reference	1373
11.119	Material.hh File Reference	1375
11.120	MathTypes.hh File Reference	1375
11.120.1	Detailed Description	1375
11.121	Matrix3.hh File Reference	1376
11.122	Matrix4.hh File Reference	1376
11.123	Mesh.hh File Reference	1377
11.124	MeshCSG.hh File Reference	1378
11.124.1	Typedef Documentation	1380
11.124.1.1	IGPtrArray	1380
11.124.1.2	GtsSurface	1380
11.125	MeshLoader.hh File Reference	1380
11.126	MeshManager.hh File Reference	1381
11.127	MeshShape.hh File Reference	1383
11.128	Model.hh File Reference	1384
11.129	ModelDatabase.hh File Reference	1385
11.129.1	Macro Definition Documentation	1387
11.129.1.1	IGZ_MODEL_DB_MANIFEST_FILENAME	1387
11.129.1.2	GZ_MODEL_MANIFEST_FILENAME	1387
11.130	ModelDatabasePrivate.hh File Reference	1387
11.131	ModelState.hh File Reference	1388
11.132	MouseEvent.hh File Reference	1389
11.133	MovableText.hh File Reference	1390
11.134	MovingWindowFilter.hh File Reference	1391
11.135	MsgFactory.hh File Reference	1392
11.136	msgsgs.hh File Reference	1393
11.137	MultiCameraSensor.hh File Reference	1396
11.138	MultiRayShape.hh File Reference	1396
11.139	Node.hh File Reference	1397
11.140	Noise.hh File Reference	1398

11.140gre_gazebo.h File Reference 1399

11.140OpenAL.hh File Reference 1401

11.140OrbitViewController.hh File Reference 1401

11.140PhysicsEngine.hh File Reference 1402

11.140PhysicsFactory.hh File Reference 1403

11.140PhysicsIface.hh File Reference 1404

11.140PhysicsTypes.hh File Reference 1406

 11.147.1Detailed Description 1408

 11.147.2Macro Definition Documentation 1409

 11.147.2.1GZ_ALL_COLLIDE 1409

 11.147.2.2GZ_FIXED_COLLIDE 1409

 11.147.2.3GZ_GHOST_COLLIDE 1409

 11.147.2.4GZ_NONE_COLLIDE 1409

 11.147.2.5GZ_SENSOR_COLLIDE 1409

11.140PID.hh File Reference 1409

11.140Plane.hh File Reference 1410

11.150PlaneShape.hh File Reference 1411

11.150Plugin.hh File Reference 1412

 11.151.1Macro Definition Documentation 1414

 11.151.1.1GZ_REGISTER_MODEL_PLUGIN 1414

 11.151.1.2GZ_REGISTER_SENSOR_PLUGIN 1415

 11.151.1.3GZ_REGISTER_SYSTEM_PLUGIN 1415

 11.151.1.4GZ_REGISTER_VISUAL_PLUGIN 1415

 11.151.1.5GZ_REGISTER_WORLD_PLUGIN 1416

11.150Pose.hh File Reference 1416

11.150Projector.hh File Reference 1417

11.150Publication.hh File Reference 1417

11.150PublicationTransport.hh File Reference 1419

11.150Publisher.hh File Reference 1420

11.150Quaternion.hh File Reference 1421

11.150Rand.hh File Reference 1422

11.150RaySensor.hh File Reference 1424

11.160RayShape.hh File Reference 1425

11.160RenderEngine.hh File Reference 1426

11.160RenderEvents.hh File Reference 1426

11.160RenderingIface.hh File Reference 1427

11.160RenderTypes.hh File Reference 1428

11.164. Macro Definition Documentation	1430
11.164.1.1GZ_VISIBILITY_ALL	1430
11.164.1.2GZ_VISIBILITY_GUI	1430
11.164.1.3GZ_VISIBILITY_SELECTABLE	1430
11.164.1.4GZ_VISIBILITY_SELECTION	1430
11.165. FIDSensor.hh File Reference	1430
11.166. FIDTag.hh File Reference	1431
11.167. FIDTagVisual.hh File Reference	1432
11.168. FIDTagVisualPrivate.hh File Reference	1432
11.169. FIDVisual.hh File Reference	1433
11.170. FIDVisualPrivate.hh File Reference	1433
11.171. Road.hh File Reference	1434
11.172. Road2d.hh File Reference	1435
11.173. RotationSpline.hh File Reference	1436
11.174. RTShaderSystem.hh File Reference	1437
11.175. Scene.hh File Reference	1438
11.176. ScrewJoint.hh File Reference	1439
11.177. SelectionObj.hh File Reference	1440
11.178. SelectionObjPrivate.hh File Reference	1442
11.179. Sensor.hh File Reference	1442
11.180. SensorFactory.hh File Reference	1443
11.181. SensorManager.hh File Reference	1444
11.182. SensorsIface.hh File Reference	1445
11.183. SensorTypes.hh File Reference	1447
11.183. Detailed Description	1449
11.184. Server.hh File Reference	1449
11.185. Shape.hh File Reference	1450
11.186. Simbody_inc.h File Reference	1451
11.187. SimbodyBallJoint.hh File Reference	1451
11.188. SimbodyBoxShape.hh File Reference	1452
11.189. SimbodyCollision.hh File Reference	1452
11.190. SimbodyCylinderShape.hh File Reference	1453
11.191. SimbodyHeightmapShape.hh File Reference	1453
11.192. SimbodyHinge2Joint.hh File Reference	1454
11.193. SimbodyHingeJoint.hh File Reference	1455
11.194. SimbodyJoint.hh File Reference	1455
11.195. SimbodyLink.hh File Reference	1456

11.196 **SimbodyMeshShape**.hh File Reference 1457

11.197 **SimbodyModel**.hh File Reference 1457

11.198 **SimbodyMultiRayShape**.hh File Reference 1458

11.199 **SimbodyPhysics**.hh File Reference 1458

11.200 **SimbodyPlaneShape**.hh File Reference 1459

11.201 **SimbodyRayShape**.hh File Reference 1460

11.202 **SimbodyScrewJoint**.hh File Reference 1460

11.203 **SimbodySliderJoint**.hh File Reference 1461

11.204 **SimbodySphereShape**.hh File Reference 1462

11.205 **SimbodyTypes**.hh File Reference 1462

 11.205.1 Detailed Description 1463

11.206 **SimbodyUniversalJoint**.hh File Reference 1464

11.207 **SingletonT**.hh File Reference 1464

11.208 **skeleton**.hh File Reference 1465

11.209 **skeletonAnimation**.hh File Reference 1467

11.210 **SliderJoint**.hh File Reference 1468

11.211 **SonarSensor**.hh File Reference 1469

11.212 **SonarVisual**.hh File Reference 1470

11.213 **SonarVisualPrivate**.hh File Reference 1471

11.214 **SphereShape**.hh File Reference 1471

11.215 **SphericalCoordinates**.hh File Reference 1472

11.216 **SphericalCoordinatesPrivate**.hh File Reference 1473

11.217 **Spline**.hh File Reference 1475

11.218 **State**.hh File Reference 1476

11.219 **STLLoader**.hh File Reference 1477

 11.219.1 Macro Definition Documentation 1479

 11.219.1.1 **COR3_MAX** 1479

 11.219.1.2 **FACE_MAX** 1479

 11.219.1.3 **LINE_MAX_LEN** 1479

 11.219.1.4 **ORDER_MAX** 1479

11.220 **SubscribeOptions**.hh File Reference 1479

11.221 **Subscriber**.hh File Reference 1480

11.222 **SubscriptionTransport**.hh File Reference 1481

11.223 **SurfaceParams**.hh File Reference 1482

11.224 **system**.hh File Reference 1484

 11.224.1 Macro Definition Documentation 1484

 11.224.1.1 **GAZEBO_HIDDEN** 1484

11.224.1.2GAZEBO_VISIBLE	1484
11.225SystemPaths.hh File Reference	1484
11.225.1Macro Definition Documentation	1486
11.225.1.1GetCurrentDir	1486
11.225.1.2LINUX	1486
11.226Time.hh File Reference	1486
11.227Timer.hh File Reference	1487
11.228TopicManager.hh File Reference	1488
11.229TransmitterVisual.hh File Reference	1489
11.230TransmitterVisualPrivate.hh File Reference	1490
11.231TransportInterface.hh File Reference	1490
11.232TransportTypes.hh File Reference	1493
11.232.1Detailed Description	1494
11.233UniversalJoint.hh File Reference	1494
11.234UpdateInfo.hh File Reference	1495
11.235UserCamera.hh File Reference	1496
11.236UserCameraPrivate.hh File Reference	1496
11.237UtilTypes.hh File Reference	1497
11.237.1Detailed Description	1498
11.238Vector2d.hh File Reference	1498
11.239Vector2i.hh File Reference	1499
11.240Vector3.hh File Reference	1500
11.241Vector4.hh File Reference	1500
11.242Video.hh File Reference	1502
11.243VideoVisual.hh File Reference	1503
11.244VideoVisualPrivate.hh File Reference	1503
11.245ViewController.hh File Reference	1504
11.246Visual.hh File Reference	1505
11.247VisualPrivate.hh File Reference	1506
11.248WindowManager.hh File Reference	1507
11.249WireBox.hh File Reference	1508
11.250WireBoxPrivate.hh File Reference	1509
11.251WirelessReceiver.hh File Reference	1509
11.252WirelessTransceiver.hh File Reference	1510
11.253WirelessTransmitter.hh File Reference	1511
11.254World.hh File Reference	1512
11.255WorldState.hh File Reference	1513

11.256	WrenchVisual.hh File Reference	1515
11.257	WrenchVisualPrivate.hh File Reference	1515

Chapter 1

Gazebo API Reference

This documentation provides useful information about the Gazebo API. The code reference is divided into the groups below. Should you find problems with this documentation - typos, unclear phrases, or insufficient detail - please create a new `bitbucket issue`. Include sufficient detail to quickly locate the problematic documentation, and set the issue's fields accordingly: Assignee - blank; Kind - bug; Priority - minor; Version - blank.

Class List - Index of all classes in Gazebo, organized alphabetically

Hierarchy - Index of classes, organized hierarchically according to their inheritance

Modules Common: Classes and files used ubiquitously across Gazebo

Events: For creating and destroying Gazebo events

Math: A set of classes that encapsulate math related properties and functions.

Messages: All messages and helper functions.

Physics: Classes for physics and dynamics

Rendering: A set of rendering related class, functions, and definitions.

Sensors: A set of sensor classes, functions, and definitions.

Transport: Handles transportation of messages.

Links Website: The main gazebo website, which contains news, downloads, and contact information.

Wiki: A collection of user supported documentation.

Tutorials: Tutorials that describe how to use Gazebo and implement your own simulations.

Download: How to download and install Gazebo

Chapter 2

Todo List

Member `gazebo::physics::Joint::GetForce` (p. 552) (unsigned int `_index`)

: not yet implemented. Get external forces applied at this Joint. Note that the unit of force should be consistent with the rest of the simulation scales.

Member `gazebo::physics::World::RunBlocking` (p. 1249) (unsigned int `_iterations=0`)

In gazebo 3.0 this should be move to the proper section.

Member `gazebo::sensors::CameraSensor::GetTopic` (p. 230) () const

to be implemented

Class `gazebo::SystemPlugin` (p. 1098)

how to make doxygen reference to the file `gazebo.cc::g_plugins?`

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Common	35
Events	47
Math	50
Messages	57
Classes for physics and dynamics	68
DART Physics	76
Bullet Physics	79
Simbody Physics	81
Rendering	83
Sensors	87
Transport	92
Utility	99

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

boost	101
gazebo	
Forward declarations for the common classes	101
gazebo::common	
Common namespace	106
gazebo::event	
Event (p. 416) namespace	110
gazebo::math	
Math namespace	111
gazebo::msgs	
Messages namespace	113
gazebo::physics	
Namespace for physics	115
gazebo::rendering	
Rendering namespace	123
gazebo::sensors	
Sensors namespace	129
gazebo::transport	133
gazebo::util	136
google	137
google::protobuf	137
google::protobuf::compiler	137
google::protobuf::compiler::cpp	137
Ogre	137
ogre	137
SimTK	137
SkyX	137

Chapter 5

Class Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

gazebo::math::Angle	145
gazebo::common::Animation	153
gazebo::common::NumericAnimation	752
gazebo::common::PoseAnimation	805
gazebo::common::AudioDecoder	161
gazebo::physics::BallJoint< T >	167
gazebo::physics::Base	168
gazebo::physics::Entity	404
gazebo::physics::Collision	235
gazebo::physics::DARTCollision	309
gazebo::physics::SimbodyCollision	943
gazebo::physics::Link	595
gazebo::physics::DARTLink	338
gazebo::physics::SimbodyLink	972
gazebo::physics::Model	678
gazebo::physics::Actor	139
gazebo::physics::DARTModel	350
gazebo::physics::SimbodyModel	982
gazebo::physics::Joint	541
gazebo::physics::DARTJoint	327
gazebo::physics::BallJoint< DARTJoint >	167
gazebo::physics::DARTBallJoint	301
gazebo::physics::Hinge2Joint< DARTJoint >	512
gazebo::physics::DARTHinge2Joint	317
gazebo::physics::HingeJoint< DARTJoint >	513
gazebo::physics::DARTHingeJoint	322
gazebo::physics::ScrewJoint< DARTJoint >	896
gazebo::physics::DARTScrewJoint	364
gazebo::physics::SliderJoint< DARTJoint >	1044
gazebo::physics::DARTSliderJoint	371
gazebo::physics::UniversalJoint< DARTJoint >	1135
gazebo::physics::DARTUniversalJoint	378

gazebo::physics::SimbodyJoint	960
gazebo::physics::BallJoint< SimbodyJoint >	167
gazebo::physics::SimbodyBallJoint	936
gazebo::physics::Hinge2Joint< SimbodyJoint >	512
gazebo::physics::SimbodyHinge2Joint	950
gazebo::physics::HingeJoint< SimbodyJoint >	513
gazebo::physics::SimbodyHingeJoint	955
gazebo::physics::ScrewJoint< SimbodyJoint >	896
gazebo::physics::SimbodyScrewJoint	1002
gazebo::physics::SliderJoint< SimbodyJoint >	1044
gazebo::physics::SimbodySliderJoint	1010
gazebo::physics::UniversalJoint< SimbodyJoint >	1135
gazebo::physics::SimbodyUniversalJoint	1016
gazebo::physics::Road	869
gazebo::physics::Shape	932
gazebo::physics::BoxShape	185
gazebo::physics::DARTBoxShape	308
gazebo::physics::SimbodyBoxShape	941
gazebo::physics::CylinderShape	298
gazebo::physics::DARTCylinderShape	313
gazebo::physics::SimbodyCylinderShape	946
gazebo::physics::HeightmapShape	506
gazebo::physics::DARTHeightmapShape	315
gazebo::physics::SimbodyHeightmapShape	948
gazebo::physics::MeshShape	675
gazebo::physics::DARTMeshShape	348
gazebo::physics::SimbodyMeshShape	981
gazebo::physics::MultiRayShape	723
gazebo::physics::DARTMultiRayShape	352
gazebo::physics::SimbodyMultiRayShape	984
gazebo::physics::PlaneShape	791
gazebo::physics::DARTPlaneShape	361
gazebo::physics::SimbodyPlaneShape	997
gazebo::physics::RayShape	849
gazebo::physics::DARTRayShape	363
gazebo::physics::SimbodyRayShape	999
gazebo::physics::SphereShape	1054
gazebo::physics::DARTSphereShape	376
gazebo::physics::SimbodySphereShape	1014
gazebo::math::Box	181
gazebo::common::Logger::Buffer	188
gazebo::common::FileLogger::Buffer	189
gazebo::common::BVHLoader	191
gazebo::transport::CallbackHelper	192
gazebo::transport::CallbackHelperT< M >	195
gazebo::transport::RawCallbackHelper	840
gazebo::transport::SubscriptionTransport	1088
gazebo::rendering::Camera	197
gazebo::rendering::DepthCamera	383
gazebo::rendering::GpuLaser	467
gazebo::rendering::UserCamera	1137
gazebo::rendering::CameraPrivate	225

gazebo::common::Color	249
gazebo::event::Connection	263
gazebo::transport::Connection	264
gazebo::event::ConnectionPrivate	275
gazebo::transport::ConnectionReadTask	276
gazebo::common::Console	277
gazebo::physics::Contact	279
gazebo::physics::ContactManager	282
gazebo::rendering::ContactVisualPrivate::ContactPoint	286
gazebo::physics::ContactPublisher	287
gazebo::rendering::Conversions	296
gazebo::physics::DARTTypes	377
gazebo::rendering::DummyPageProvider	395
gazebo::rendering::DynamicRenderable	400
gazebo::rendering::DynamicLines	397
gazebo::event::Event	416
gazebo::event::EventT< T >	434
gazebo::event::EventPrivate	419
gazebo::event::EventTPrivate< T >	443
gazebo::event::Events	421
gazebo::rendering::Events	432
gazebo::common::Exception	444
gazebo::common::InternalError	538
gazebo::common::AssertionInternalError	160
gazebo::common::FileLogger	447
gazebo::physics::FrictionPyramid	455
google::protobuf::compiler::cpp::GazeboGenerator	461
gazebo::physics::GearboxJoint< T >	462
gazebo::rendering::Grid	488
gazebo::physics::Gripper	492
gazebo::rendering::GUIOverlay	494
gazebo::rendering::GUIOverlayPrivate	498
gazebo::rendering::GzTerrainMatGen	499
gazebo::rendering::Heightmap	499
gazebo::common::HeightmapData	504
gazebo::common::ImageHeightmap	523
gazebo::physics::Hinge2Joint< T >	512
gazebo::physics::HingeJoint< T >	513
gazebo::common::Image	515
gazebo::physics::Inertial	529
gazebo::transport::IOManager	540
gazebo::physics::JointController	568
gazebo::physics::JointControllerPrivate	572
gazebo::physics::JointWrench	581
gazebo::common::KeyEvent	583
gazebo::common::KeyFrame	584
gazebo::common::NumericKeyFrame	754
gazebo::common::PoseKeyFrame	808
gazebo::rendering::Light	589
gazebo::common::Logger	625
Logplay	631
gazebo::Master	637

gazebo::common::Material	639
gazebo::math::Matrix3	648
gazebo::math::Matrix4	653
gazebo::common::Mesh	660
gazebo::common::MeshCSG	667
gazebo::common::MeshLoader	669
gazebo::common::ColladaLoader	234
gazebo::common::STLLoader	1072
gazebo::common::ModelDatabasePrivate	694
gazebo::common::MouseEvent	706
gazebo::rendering::MovableText	709
gazebo::common::MovingWindowFilter< T >	715
gazebo::common::MovingWindowFilterPrivate< T >	716
gazebo::msgs::MsgFactory	717
gazebo::transport::Node	731
gazebo::common::NodeAnimation	738
gazebo::common::NodeAssignment	741
gazebo::common::NodeTransform	742
gazebo::sensors::Noise	748
gazebo::sensors::GaussianNoiseModel	457
gazebo::sensors::ImageGaussianNoiseModel	520
gazebo::sensors::NoiseFactory	751
gazebo::util::OpenALSink	757
gazebo::util::OpenALSource	758
gazebo::physics::PhysicsEngine	766
gazebo::physics::DARTPhysics	354
gazebo::physics::SimbodyPhysics	986
gazebo::physics::PhysicsFactory	781
gazebo::common::PID	782
gazebo::math::Plane	788
gazebo::PluginT< T >	794
gazebo::PluginT< ModelPlugin >	794
gazebo::ModelPlugin	696
gazebo::PluginT< SensorPlugin >	794
gazebo::SensorPlugin	924
gazebo::PluginT< SystemPlugin >	794
gazebo::SystemPlugin	1098
gazebo::PluginT< VisualPlugin >	794
gazebo::VisualPlugin	1217
gazebo::PluginT< WorldPlugin >	794
gazebo::WorldPlugin	1251
gazebo::math::Pose	797
gazebo::rendering::Projector	810
gazebo::transport::Publication	812
gazebo::transport::PublicationTransport	818
gazebo::transport::Publisher	820
gazebo::transport::PublishTask	823
gazebo::math::Quaternion	824
gazebo::math::Rand	838
Road	870
gazebo::rendering::Road2d	871
gazebo::math::RotationSpline	871

gazebo::rendering::Scene	879
gazebo::physics::ScrewJoint< T >	896
gazebo::sensors::Sensor	907
gazebo::sensors::CameraSensor	227
gazebo::sensors::ContactSensor	287
gazebo::sensors::DepthCameraSensor	387
gazebo::sensors::ForceTorqueSensor	449
gazebo::sensors::GpsSensor	464
gazebo::sensors::GpuRaySensor	477
gazebo::sensors::ImuSensor	525
gazebo::sensors::MultiCameraSensor	718
gazebo::sensors::RaySensor	842
gazebo::sensors::RFIDSensor	859
gazebo::sensors::RFIDTag	861
gazebo::sensors::SonarSensor	1047
gazebo::sensors::WirelessTransceiver	1232
gazebo::sensors::WirelessReceiver	1230
gazebo::sensors::WirelessTransmitter	1235
SensorFactor	919
gazebo::sensors::SensorFactory	919
gazebo::Server	926
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg	928
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL	930
gazebo::sensors::SimTimeEvent	1020
gazebo::sensors::SimTimeEventHandler	1021
SingletonT< T >	1022
gazebo::common::MeshManager	670
gazebo::common::ModelDatabase	693
gazebo::common::SystemPaths	1092
gazebo::rendering::RenderEngine	855
gazebo::rendering::RTShaderSystem	875
gazebo::sensors::SensorManager	920
gazebo::transport::ConnectionManager	271
gazebo::transport::TopicManager	1122
gazebo::util::DiagnosticManager	390
gazebo::util::LogPlay	628
gazebo::util::LogRecord	631
gazebo::util::OpenAL	755
SingletonT< ConnectionManager >	1022
SingletonT< DiagnosticManager >	1022
SingletonT< LogPlay >	1022
SingletonT< LogRecord >	1022
SingletonT< MeshManager >	1022
SingletonT< ModelDatabase >	1022
SingletonT< OpenAL >	1022
SingletonT< RenderEngine >	1022
SingletonT< RTShaderSystem >	1022
SingletonT< SensorManager >	1022
SingletonT< SystemPaths >	1022
SingletonT< TopicManager >	1022
gazebo::common::Skeleton	1024
gazebo::common::SkeletonAnimation	1031
gazebo::common::SkeletonNode	1035
gazebo::physics::SliderJoint< T >	1044

gazebo::rendering::GzTerrainMatGen::SM2Profile	1045
gazebo::common::SphericalCoordinates	1057
gazebo::common::SphericalCoordinatesPrivate	1062
gazebo::math::Spline	1063
gazebo::physics::State	1068
gazebo::physics::CollisionState	245
gazebo::physics::JointState	574
gazebo::physics::LinkState	618
gazebo::physics::ModelState	698
gazebo::physics::WorldState	1253
gazebo::common::SubMesh	1074
gazebo::transport::SubscribeOptions	1084
gazebo::transport::Subscriber	1086
gazebo::physics::SurfaceParams	1090
gazebo::common::Time	1099
gazebo::common::Timer	1120
gazebo::util::DiagnosticTimer	394
gazebo::physics::TrajectoryInfo	1129
gazebo::physics::UniversalJoint< T >	1135
gazebo::common::UpdateInfo	1136
gazebo::rendering::UserCameraPrivate	1146
gazebo::math::Vector2d	1147
gazebo::math::Vector2i	1156
gazebo::math::Vector3	1165
gazebo::math::Vector4	1178
gazebo::common::Video	1188
gazebo::rendering::ViewController	1192
gazebo::rendering::FPSViewController	453
gazebo::rendering::OrbitViewController	763
gazebo::rendering::Visual	1196
gazebo::rendering::ArrowVisual	157
gazebo::rendering::AxisVisual	163
gazebo::rendering::CameraVisual	231
gazebo::rendering::COMVisual	260
gazebo::rendering::ContactVisual	292
gazebo::rendering::JointVisual	579
gazebo::rendering::LaserVisual	586
gazebo::rendering::RFIDTagVisual	864
gazebo::rendering::RFIDVisual	867
gazebo::rendering::SelectionObj	899
gazebo::rendering::SonarVisual	1051
gazebo::rendering::TransmitterVisual	1131
gazebo::rendering::VideoVisual	1189
gazebo::rendering::WrenchVisual	1259
gazebo::rendering::VisualPrivate	1219
gazebo::rendering::ArrowVisualPrivate	159
gazebo::rendering::AxisVisualPrivate	165
gazebo::rendering::CameraVisualPrivate	233
gazebo::rendering::COMVisualPrivate	262
gazebo::rendering::ContactVisualPrivate	293
gazebo::rendering::JointVisualPrivate	580
gazebo::rendering::LaserVisualPrivate	587
gazebo::rendering::RFIDTagVisualPrivate	865

gazebo::rendering::RFIDVisualPrivate	868
gazebo::rendering::SelectionObjPrivate	903
gazebo::rendering::SonarVisualPrivate	1052
gazebo::rendering::TransmitterVisualPrivate	1132
gazebo::rendering::VideoVisualPrivate	1191
gazebo::rendering::WrenchVisualPrivate	1261
gazebo::rendering::WindowManager	1225
gazebo::rendering::WireBox	1227
gazebo::rendering::WireBoxPrivate	1229
gazebo::physics::World	1239

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

gazebo::physics::Actor	
Actor (p. 139) class enables GPU based mesh model / skeleton scriptable animation	139
gazebo::math::Angle	
An angle and related functions	145
gazebo::common::Animation	
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes	153
gazebo::rendering::ArrowVisual	
Basic arrow visualization	157
gazebo::rendering::ArrowVisualPrivate	
Private data for the Arrow Visual (p. 1196) class	159
gazebo::common::AssertionInternalError	
Class for generating Exceptions which come from gazebo assertions	160
gazebo::common::AudioDecoder	
An audio decoder based on FFMPEG	161
gazebo::rendering::AxisVisual	
Basic axis visualization	163
gazebo::rendering::AxisVisualPrivate	
Private data for the Axis Visual (p. 1196) class	165
gazebo::physics::BallJoint< T >	
Base (p. 168) class for a ball joint	167
gazebo::physics::Base	
Base (p. 168) class for most physics classes	168
gazebo::math::Box	
Mathematical representation of a box and related functions	181
gazebo::physics::BoxShape	
Box geometry primitive	185
gazebo::common::Logger::Buffer	
String buffer for the base logger	188
gazebo::common::FileLogger::Buffer	
String buffer for the file logger	189
gazebo::common::BVHLoader	
Handles loading BVH animation files	191

gazebo::transport::CallbackHelper	A helper class to handle callbacks when messages arrive	192
gazebo::transport::CallbackHelperT < M >	Callback helper Template	195
gazebo::rendering::Camera	Basic camera sensor	197
gazebo::rendering::CameraPrivate	Private data for the Camera (p. 197) class	225
gazebo::sensors::CameraSensor	Basic camera sensor	227
gazebo::rendering::CameraVisual	Basic camera visualization	231
gazebo::rendering::CameraVisualPrivate		233
gazebo::common::ColladaLoader	Class used to load Collada mesh files	234
gazebo::physics::Collision	Base (p. 168) class for all collision entities	235
gazebo::physics::CollisionState	Store state information of a physics::Collision (p. 235) object	245
gazebo::common::Color	Defines a color	249
gazebo::rendering::COMVisual	Basic Center of Mass visualization	260
gazebo::rendering::COMVisualPrivate	Private data for the COM Visual (p. 1196) class	262
gazebo::event::Connection	A class that encapsulates a connection	263
gazebo::transport::Connection	Single TCP/IP connection manager	264
gazebo::transport::ConnectionManager	Manager of connections	271
gazebo::event::ConnectionPrivate		275
gazebo::transport::ConnectionReadTask	276	
gazebo::common::Console	Container for loggers, and global logging options (such as verbose vs	277
gazebo::physics::Contact	A contact between two collisions	279
gazebo::physics::ContactManager	Aggregates all the contact information generated by the collision detection engine	282
gazebo::rendering::ContactVisualPrivate::ContactPoint	A contact point visualization	286
gazebo::physics::ContactPublisher	A custom contact publisher created for each contact filter in the Contact (p. 279) Manager	287
gazebo::sensors::ContactSensor	Contact sensor	287
gazebo::rendering::ContactVisual	Contact visualization	292
gazebo::rendering::ContactVisualPrivate	Private data for the Arrow Visual (p. 1196) class	293
gazebo::rendering::Conversions	Conversions (p. 296) Conversions.hh (p. 1306) rendering/Conversions.hh (p. 1306)	296
gazebo::physics::CylinderShape	Cylinder collision	298

gazebo::physics::DARTBallJoint	
An DARTBallJoint (p. 301)	301
gazebo::physics::DARTBoxShape	
DART Box shape	308
gazebo::physics::DARTCollision	
Base (p. 168) class for all DART collisions	309
gazebo::physics::DARTCylinderShape	
DART cylinder shape	313
gazebo::physics::DARTHeightmapShape	
DART Height map collision	315
gazebo::physics::DARTHinge2Joint	
A two axis hinge joint	317
gazebo::physics::DARTHingeJoint	
A single axis hinge joint	322
gazebo::physics::DARTJoint	
DART joint interface	327
gazebo::physics::DARTLink	
DART Link (p. 595) class	338
gazebo::physics::DARTMeshShape	
Triangle mesh collision	348
gazebo::physics::DARTModel	
DART model class	350
gazebo::physics::DARTMultiRayShape	
DART specific version of MultiRayShape (p. 723)	352
gazebo::physics::DARTPhysics	
DART physics engine	354
gazebo::physics::DARTPlaneShape	
An DART Plane shape	361
gazebo::physics::DARTRayShape	
Ray collision	363
gazebo::physics::DARTScrewJoint	
A screw joint	364
gazebo::physics::DARTSliderJoint	
A slider joint	371
gazebo::physics::DARTSphereShape	
A DART sphere shape	376
gazebo::physics::DARTTypes	
A set of functions for converting between the math types used by gazebo and dart	377
gazebo::physics::DARTUniversalJoint	
A universal joint	378
gazebo::rendering::DepthCamera	
Depth camera used to render depth data into an image buffer	383
gazebo::sensors::DepthCameraSensor	
.	387
gazebo::util::DiagnosticManager	
A diagnostic manager class	390
gazebo::util::DiagnosticTimer	
A timer designed for diagnostics	394
gazebo::rendering::DummyPageProvider	
Pretends to provide procedural page content to avoid page loading	395
gazebo::rendering::DynamicLines	
Class for drawing lines that can change	397
gazebo::rendering::DynamicRenderable	
Abstract base class providing mechanisms for dynamically growing hardware buffers	400

gazebo::physics::Entity	
Base (p. 168) class for all physics objects in Gazebo	404
gazebo::event::Event	
Base class for all events	416
gazebo::event::EventPrivate	419
gazebo::event::Events	
An Event (p. 416) class to get notifications for simulator events	421
gazebo::rendering::Events	
Base class for rendering events	432
gazebo::event::EventT< T >	
A class for event processing	434
gazebo::event::EventTPrivate< T >	443
gazebo::common::Exception	
Class for generating exceptions	444
gazebo::common::FileLogger	
A logger that outputs messages to a file	447
gazebo::sensors::ForceTorqueSensor	
Sensor (p. 907) for measure force and torque on a joint	449
gazebo::rendering::FPSViewController	
First Person Shooter style view controller	453
gazebo::physics::FrictionPyramid	
Parameters used for friction pyramid model	455
gazebo::sensors::GaussianNoiseModel	
Gaussian noise class	457
google::protobuf::compiler::cpp::GazeboGenerator	
Google protobuf message generator for gazebo::msgs (p. 113)	461
gazebo::physics::GearboxJoint< T >	
A double axis gearbox joint	462
gazebo::sensors::GpsSensor	
GpsSensor (p. 464) to provide position measurement	464
gazebo::rendering::GpuLaser	
GPU based laser distance sensor	467
gazebo::sensors::GpuRaySensor	477
gazebo::rendering::Grid	
Displays a grid of cells, drawn with lines	488
gazebo::physics::Gripper	
A gripper abstraction	492
gazebo::rendering::GUIOverlay	
A class that creates a CEGUI overlay on a render window	494
gazebo::rendering::GUIOverlayPrivate	
Private data for the GUIOverlay (p. 494) class	498
gazebo::rendering::GzTerrainMatGen	499
gazebo::rendering::Heightmap	
Rendering a terrain using heightmap information	499
gazebo::common::HeightmapData	
Encapsulates a generic heightmap data file	504
gazebo::physics::HeightmapShape	
HeightmapShape (p. 506) collision shape builds a heightmap from an image	506
gazebo::physics::Hinge2Joint< T >	
A two axis hinge joint	512
gazebo::physics::HingeJoint< T >	
A single axis hinge joint	513
gazebo::common::Image	
Encapsulates an image	515

gazebo::sensors::ImageGaussianNoiseModel	520
gazebo::common::ImageHeightmap	
Encapsulates an image that will be interpreted as a heightmap	523
gazebo::sensors::ImuSensor	
An IMU sensor	525
gazebo::physics::Inertial	
A class for inertial information about a link	529
gazebo::common::InternalError	
Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs	538
gazebo::transport::IOManager	
Manages boost::asio IO	540
gazebo::physics::Joint	
Base (p. 168) class for all joints	541
gazebo::physics::JointController	
A class for manipulating physics::Joint (p. 541)	568
gazebo::physics::JointControllerPrivate	572
gazebo::physics::JointState	
Keeps track of state of a physics::Joint (p. 541)	574
gazebo::rendering::JointVisual	
Visualization for joints	579
gazebo::rendering::JointVisualPrivate	
Private data for the Joint Visual (p. 1196) class	580
gazebo::physics::JointWrench	
Wrench information from a joint	581
gazebo::common::KeyEvent	
Generic description of a keyboard event	583
gazebo::common::KeyFrame	
A key frame in an animation	584
gazebo::rendering::LaserVisual	
Visualization for laser data	586
gazebo::rendering::LaserVisualPrivate	
Private data for the Laser Visual (p. 1196) class	587
gazebo::rendering::Light	
A light source	589
gazebo::physics::Link	
Link (p. 595) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body	595
gazebo::physics::LinkState	
Store state information of a physics::Link (p. 595) object	618
gazebo::common::Logger	
Terminal logger	625
gazebo::util::LogPlay	628
Logplay	
Open and playback log files that were recorded using LogRecord	631
gazebo::util::LogRecord	
Addtogroup gazebo_util	631
gazebo::Master	
A manager that directs topic connections, enables each gazebo network client to locate one another for peer-to-peer communication	637
gazebo::common::Material	
Encapsulates description of a material	639
gazebo::math::Matrix3	
A 3x3 matrix class	648

gazebo::math::Matrix4	A 3x3 matrix class	653
gazebo::common::Mesh	A 3D mesh	660
gazebo::common::MeshCSG	Creates CSG meshes	667
gazebo::common::MeshLoader	Base class for loading meshes	669
gazebo::common::MeshManager	Maintains and manages all meshes	670
gazebo::physics::MeshShape	Triangle mesh collision shape	675
gazebo::physics::Model	A model is a collection of links, joints, and plugins	678
gazebo::common::ModelDatabase	Connects to model database, and has utility functions to find models	693
gazebo::common::ModelDatabasePrivate	Private class attributes for ModelDatabase (p. 693)	694
gazebo::ModelPlugin	A plugin with access to physics::Model (p. 678)	696
gazebo::physics::ModelState	Store state information of a physics::Model (p. 678) object	698
gazebo::common::MouseEvent	Generic description of a mouse event	706
gazebo::rendering::MovableText	Movable text	709
gazebo::common::MovingWindowFilter< T >	Base class for MovingWindowFilter (p. 715)	715
gazebo::common::MovingWindowFilterPrivate< T >		716
gazebo::msgs::MsgFactory	A factory that generates protobuf message based on a string type	717
gazebo::sensors::MultiCameraSensor	Multiple camera sensor	718
gazebo::physics::MultiRayShape	Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner	723
gazebo::transport::Node	A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics	731
gazebo::common::NodeAnimation	Node animation	738
gazebo::common::NodeAssignment	Vertex to node weighted assignment for skeleton animation visualization	741
gazebo::common::NodeTransform	NodeTransform (p. 742) Skeleton.hh (p. 1465) common/common.hh	742
gazebo::sensors::Noise	Noise (p. 748) models for sensor output signals	748
gazebo::sensors::NoiseFactory	Use this noise manager for creating and loading noise models	751
gazebo::common::NumericAnimation	A numeric animation	752
gazebo::common::NumericKeyFrame	A keyframe for a NumericAnimation (p. 752)	754
gazebo::util::OpenAL	3D audio setup and playback	755

gazebo::util::OpenALSink	
OpenAL (p. 755) Listener	757
gazebo::util::OpenALSource	
OpenAL (p. 755) Source	758
gazebo::rendering::OrbitViewController	
Orbit view controller	763
gazebo::physics::PhysicsEngine	
Base (p. 168) class for a physics engine	766
gazebo::physics::PhysicsFactory	
The physics factory instantiates different physics engines	781
gazebo::common::PID	
Generic PID (p. 782) controller class	782
gazebo::math::Plane	
A plane and related functions	788
gazebo::physics::PlaneShape	
Collision (p. 235) for an infinite plane	791
gazebo::PluginT < T >	
A class which all plugins must inherit from	794
gazebo::math::Pose	
Encapsulates a position and rotation in three space	797
gazebo::common::PoseAnimation	
A pose animation	805
gazebo::common::PoseKeyFrame	
A keyframe for a PoseAnimation (p. 805)	808
gazebo::rendering::Projector	
Projects a material onto surface, light a light projector	810
gazebo::transport::Publication	
A publication for a topic	812
gazebo::transport::PublicationTransport	
Transport/transport.hh	818
gazebo::transport::Publisher	
A publisher of messages on a topic	820
gazebo::transport::PublishTask	
823	
gazebo::math::Quaternion	
A quaternion class	824
gazebo::math::Rand	
Random number generator class	838
gazebo::transport::RawCallbackHelper	
Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher	840
gazebo::sensors::RaySensor	
Sensor (p. 907) with one or more rays	842
gazebo::physics::RayShape	
Base (p. 168) class for Ray collision geometry	849
gazebo::rendering::RenderEngine	
Adaptor to Ogre3d	855
gazebo::sensors::RFIDSensor	
Sensor (p. 907) class for RFID type of sensor	859
gazebo::sensors::RFIDTag	
RFIDTag (p. 861) to interact with RFIDTagSensors	861
gazebo::rendering::RFIDTagVisual	
Visualization for RFID tags sensor	864

gazebo::rendering::RFIDTagVisualPrivate	Private data for the RFID Tag Visual (p. 1196) class	865
gazebo::rendering::RFIDVisual	Visualization for RFID sensor	867
gazebo::rendering::RFIDVisualPrivate	Private data for the RFID Visual (p. 1196) class	868
gazebo::physics::Road	For building a Road (p. 869) from SDF	869
Road	Used to render a strip of road	870
gazebo::rendering::Road2d	871
gazebo::math::RotationSpline	Spline (p. 1063) for rotations	871
gazebo::rendering::RTShaderSystem	Implements Ogre (p. 137)'s Run-Time Shader system	875
gazebo::rendering::Scene	Representation of an entire scene graph	879
gazebo::physics::ScrewJoint< T >	A screw joint, which has both prismatic and rotational DOFs	896
gazebo::rendering::SelectionObj	Interactive selection object for models and links	899
gazebo::rendering::SelectionObjPrivate	Private data for the Selection Obj class	903
gazebo::sensors::Sensor	Base class for sensors	907
SensorFactor	The sensor factory; the class is just for namespacing purposes	919
gazebo::sensors::SensorFactory	919
gazebo::sensors::SensorManager	Class to manage and update all sensors	920
gazebo::SensorPlugin	A plugin with access to physics::Sensor	924
gazebo::Server	926
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg	Keeping the CG shader for reference	928
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL	Utility class to help with generating shaders for GLSL	930
gazebo::physics::Shape	Base (p. 168) class for all shapes	932
gazebo::physics::SimbodyBallJoint	SimbodyBallJoint (p. 936) class models a ball joint in Simbody	936
gazebo::physics::SimbodyBoxShape	Simbody box collision	941
gazebo::physics::SimbodyCollision	Simbody collisions	943
gazebo::physics::SimbodyCylinderShape	Cylinder collision	946
gazebo::physics::SimbodyHeightmapShape	Height map collision	948
gazebo::physics::SimbodyHinge2Joint	A two axis hinge joint	950
gazebo::physics::SimbodyHingeJoint	A single axis hinge joint	955

gazebo::physics::SimbodyJoint	
Base (p. 168) class for all joints	960
gazebo::physics::SimbodyLink	
Simbody Link (p. 595) class	972
gazebo::physics::SimbodyMeshShape	
Triangle mesh collision	981
gazebo::physics::SimbodyModel	
A model is a collection of links, joints, and plugins	982
gazebo::physics::SimbodyMultiRayShape	
Simbody specific version of MultiRayShape (p. 723)	984
gazebo::physics::SimbodyPhysics	
Simbody physics engine	986
gazebo::physics::SimbodyPlaneShape	
Simbody collision for an infinite plane	997
gazebo::physics::SimbodyRayShape	
Ray shape for simbody	999
gazebo::physics::SimbodyScrewJoint	
A screw joint	1002
gazebo::physics::SimbodySliderJoint	
A slider joint	1010
gazebo::physics::SimbodySphereShape	
Simbody sphere collision	1014
gazebo::physics::SimbodyUniversalJoint	
A simbody universal joint class	1016
gazebo::sensors::SimTimeEvent	
1020	
gazebo::sensors::SimTimeEventHandler	
Monitors simulation time, and notifies conditions when a specified time has been reached	1021
SingletonT< T >	
Singleton template class	1022
gazebo::common::Skeleton	
A skeleton	1024
gazebo::common::SkeletonAnimation	
Skeleton (p. 1024) animation	1031
gazebo::common::SkeletonNode	
A skeleton node	1035
gazebo::physics::SliderJoint< T >	
A slider joint	1044
gazebo::rendering::GzTerrainMatGen::SM2Profile	
Shader model 2 profile target	1045
gazebo::sensors::SonarSensor	
Sensor (p. 907) with sonar cone	1047
gazebo::rendering::SonarVisual	
Visualization for sonar data	1051
gazebo::rendering::SonarVisualPrivate	
Private data for the Sonar Visual (p. 1196) class	1052
gazebo::physics::SphereShape	
Sphere collision shape	1054
gazebo::common::SphericalCoordinates	
Convert spherical coordinates for planetary surfaces	1057
gazebo::common::SphericalCoordinatesPrivate	
Common/common.hh	1062
gazebo::math::Spline	
Splines	1063

gazebo::physics::State	
State (p. 1068) of an entity	1068
gazebo::common::STLLoader	
Class used to load STL mesh files	1072
gazebo::common::SubMesh	
A child mesh	1074
gazebo::transport::SubscribeOptions	
Options for a subscription	1084
gazebo::transport::Subscriber	
A subscriber to a topic	1086
gazebo::transport::SubscriptionTransport	
Transport/transport.hh	1088
gazebo::physics::SurfaceParams	
SurfaceParams (p. 1090) defines various Surface contact parameters	1090
gazebo::common::SystemPaths	
Functions to handle getting system paths, keeps track of:	1092
gazebo::SystemPlugin	
A plugin loaded within the gzserver on startup	1098
gazebo::common::Time	
A Time (p. 1099) class, can be used to hold wall- or sim-time	1099
gazebo::common::Timer	
A timer class, used to time things in real world walltime	1120
gazebo::transport::TopicManager	
Manages topics and their subscriptions	1122
gazebo::physics::TrajectoryInfo	
Information about a trajectory for an Actor (p. 139)	1129
gazebo::rendering::TransmitterVisual	
Visualization for the wireless propagation data	1131
gazebo::rendering::TransmitterVisualPrivate	
Private data for the Transmitter Visual (p. 1196) class	1132
gazebo::physics::UniversalJoint< T >	
A universal joint	1135
gazebo::common::UpdateInfo	
Information for use in an update event	1136
gazebo::rendering::UserCamera	
A camera used for user visualization of a scene	1137
gazebo::rendering::UserCameraPrivate	
Private data for the UserCamera (p. 1137) class	1146
gazebo::math::Vector2d	
Generic double x, y vector	1147
gazebo::math::Vector2i	
Generic integer x, y vector	1156
gazebo::math::Vector3	
Generic vector containing 3 elements	1165
gazebo::math::Vector4	
Double Generic x, y, z, w vector	1178
gazebo::common::Video	
Handle video encoding and decoding using libavcodec	1188
gazebo::rendering::VideoVisual	
A visual element that displays a video as a texture	1189
gazebo::rendering::VideoVisualPrivate	
Private data for the Video Visual (p. 1196) class	1191
gazebo::rendering::ViewController	
Base class for view controllers	1192

gazebo::rendering::Visual	
A renderable object	1196
gazebo::VisualPlugin	
A plugin loaded within the gzserver on startup	1217
gazebo::rendering::VisualPrivate	
Private data for the Visual (p. 1196) class	1219
gazebo::rendering::WindowManager	
Class to manage render windows	1225
gazebo::rendering::WireBox	
Draws a wireframe box	1227
gazebo::rendering::WireBoxPrivate	
Private data for the WireBox (p. 1227) class	1229
gazebo::sensors::WirelessReceiver	
Sensor (p. 907) class for receiving wireless signals	1230
gazebo::sensors::WirelessTransceiver	
Sensor (p. 907) class for receiving wireless signals	1232
gazebo::sensors::WirelessTransmitter	
Transmitter to send wireless signals	1235
gazebo::physics::World	
The world provides access to all other object within a simulated environment	1239
gazebo::WorldPlugin	
A plugin with access to physics::World (p. 1239)	1251
gazebo::physics::WorldState	
Store state information of a physics::World (p. 1239) object	1253
gazebo::rendering::WrenchVisual	
Visualization for sonar data	1259
gazebo::rendering::WrenchVisualPrivate	
Private data for the Wrench Visual (p. 1196) class	1261

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Actor.hh	1265
Angle.hh	1266
Animation.hh	1269
ArrowVisual.hh	1270
ArrowVisualPrivate.hh	1271
Assert.hh	1271
AudioDecoder.hh	1272
AxisVisual.hh	1274
AxisVisualPrivate.hh	1274
BallJoint.hh	1275
Base.hh	1276
Base64.hh	1277
Box.hh	1278
BoxShape.hh	1279
BVHLoader.hh	1280
CallbackHelper.hh	1282
Camera.hh	1283
CameraPrivate.hh	1285
CameraSensor.hh	1285
CameraVisual.hh	1286
CameraVisualPrivate.hh	1287
cegui.h	1287
ColladaLoader.hh	1288
Collision.hh	1289
CollisionState.hh	1290
Color.hh	1291
Commonface.hh	1292
CommonTypes.hh	1294
COMVisual.hh	1296
COMVisualPrivate.hh	1296
Connection.hh	1297
ConnectionManager.hh	1299
Console.hh	1300
Contact.hh	1301

ContactManager.hh	1303
ContactSensor.hh	1304
ContactVisual.hh	1304
ContactVisualPrivate.hh	1305
Conversions.hh	1306
CylinderShape.hh	1307
dart_inc.h	1308
DARTBallJoint.hh	1308
DARTBoxShape.hh	1309
DARTCollision.hh	1310
DARTCylinderShape.hh	1311
DARTHeightmapShape.hh	1311
DARTHinge2Joint.hh	1312
DARTHingeJoint.hh	1312
DARTJoint.hh	1313
DARTLink.hh	1313
DARTMeshShape.hh	1314
DARTModel.hh	1315
DARTMultiRayShape.hh	1315
DARTPhysics.hh	1316
DARTPlaneShape.hh	1316
DARTRayShape.hh	1317
DARTScrewJoint.hh	1318
DARTSliderJoint.hh	1318
DARTSphereShape.hh	1319
DARTTypes.hh	
DART wrapper forward declarations and typedefs	1319
DARTUniversalJoint.hh	1320
Dem.hh	1321
DemPrivate.hh	1322
DepthCamera.hh	1323
DepthCameraSensor.hh	1324
Diagnostics.hh	1324
DynamicLines.hh	1325
DynamicRenderable.hh	1326
Entity.hh	1327
Event.hh	1328
Events.hh	1329
Exception.hh	1330
ffmpeg_inc.h	1332
ForceTorqueSensor.hh	1332
FPSViewController.hh	1333
GaussianNoiseModel.hh	1334
gazebo.hh	1334
gazebo_core.hh	1336
GazeboGenerator.hh	1337
GearboxJoint.hh	1338
GpsSensor.hh	1339
GpuLaser.hh	1339
GpuRaySensor.hh	1340
Grid.hh	1341
Gripper.hh	1341
GUIOverlay.hh	1342
GUIOverlayPrivate.hh	1343

Heightmap.hh	1344
HeightmapData.hh	1345
HeightmapShape.hh	1346
Helpers.hh	1347
Hinge2Joint.hh	1350
HingeJoint.hh	1351
Image.hh	1352
ImageHeightmap.hh	1352
ImuSensor.hh	1354
Inertial.hh	1354
IOManager.hh	1355
Joint.hh	1356
JointController.hh	1358
JointControllerPrivate.hh	1359
JointState.hh	1359
JointVisual.hh	1360
JointVisualPrivate.hh	1361
JointWrench.hh	1362
KeyEvent.hh	1363
KeyFrame.hh	1364
LaserVisual.hh	1365
LaserVisualPrivate.hh	1366
Light.hh	1367
Link.hh	1367
LinkState.hh	1369
LogPlay.hh	1370
LogRecord.hh	1370
mainpage.html	1371
MapShape.hh	1371
Master.hh	1372
common/Material.hh	1373
rendering/Material.hh	1375
MathTypes.hh	
Forward declarations for the math classes	1375
Matrix3.hh	1376
Matrix4.hh	1376
Mesh.hh	1377
MeshCSG.hh	1378
MeshLoader.hh	1380
MeshManager.hh	1381
MeshShape.hh	1383
Model.hh	1384
ModelDatabase.hh	1385
ModelDatabasePrivate.hh	1387
ModelState.hh	1388
MouseEvent.hh	1389
MovableText.hh	1390
MovingWindowFilter.hh	1391
MsgFactory.hh	1392
msgs.hh	1393
MultiCameraSensor.hh	1396
MultiRayShape.hh	1396
Node.hh	1397
Noise.hh	1398

ogre_gazebo.h	1399
OpenAL.hh	1401
OrbitViewController.hh	1401
PhysicsEngine.hh	1402
PhysicsFactory.hh	1403
PhysicsIface.hh	1404
PhysicsTypes.hh	
Default namespace for gazebo	1406
PID.hh	1409
Plane.hh	1410
PlaneShape.hh	1411
Plugin.hh	1412
Pose.hh	1416
Projector.hh	1417
Publication.hh	1417
PublicationTransport.hh	1419
Publisher.hh	1420
Quaternion.hh	1421
Rand.hh	1422
RaySensor.hh	1424
RayShape.hh	1425
RenderEngine.hh	1426
RenderEvents.hh	1426
RenderingIface.hh	1427
RenderTypes.hh	1428
RFIDSensor.hh	1430
RFIDTag.hh	1431
RFIDTagVisual.hh	1432
RFIDTagVisualPrivate.hh	1432
RFIDVisual.hh	1433
RFIDVisualPrivate.hh	1433
Road.hh	1434
Road2d.hh	1435
RotationSpline.hh	1436
RTShaderSystem.hh	1437
Scene.hh	1438
ScrewJoint.hh	1439
SelectionObj.hh	1440
SelectionObjPrivate.hh	1442
Sensor.hh	1442
SensorFactory.hh	1443
SensorManager.hh	1444
SensorsIface.hh	1445
SensorTypes.hh	
Forward declarations and typedefs for sensors	1447
Server.hh	1449
Shape.hh	1450
simbody_inc.h	1451
SimbodyBallJoint.hh	1451
SimbodyBoxShape.hh	1452
SimbodyCollision.hh	1452
SimbodyCylinderShape.hh	1453
SimbodyHeightmapShape.hh	1453
SimbodyHinge2Joint.hh	1454

SimbodyHingeJoint.hh	1455
SimbodyJoint.hh	1455
SimbodyLink.hh	1456
SimbodyMeshShape.hh	1457
SimbodyModel.hh	1457
SimbodyMultiRayShape.hh	1458
SimbodyPhysics.hh	1458
SimbodyPlaneShape.hh	1459
SimbodyRayShape.hh	1460
SimbodyScrewJoint.hh	1460
SimbodySliderJoint.hh	1461
SimbodySphereShape.hh	1462
SimbodyTypes.hh	
Simbody wrapper forward declarations and typedefs	1462
SimbodyUniversalJoint.hh	1464
SingletonT.hh	1464
Skeleton.hh	1465
SkeletonAnimation.hh	1467
SliderJoint.hh	1468
SonarSensor.hh	1469
SonarVisual.hh	1470
SonarVisualPrivate.hh	1471
SphereShape.hh	1471
SphericalCoordinates.hh	1472
SphericalCoordinatesPrivate.hh	1473
Spline.hh	1475
State.hh	1476
STLLoader.hh	1477
SubscribeOptions.hh	1479
Subscriber.hh	1480
SubscriptionTransport.hh	1481
SurfaceParams.hh	1482
system.hh	1484
SystemPaths.hh	1484
Time.hh	1486
Timer.hh	1487
TopicManager.hh	1488
TransmitterVisual.hh	1489
TransmitterVisualPrivate.hh	1490
TransportIface.hh	1490
TransportTypes.hh	
Forward declarations for transport	1493
UniversalJoint.hh	1494
UpdateInfo.hh	1495
UserCamera.hh	1496
UserCameraPrivate.hh	1496
UtilTypes.hh	1497
Vector2d.hh	1498
Vector2i.hh	1499
Vector3.hh	1500
Vector4.hh	1500
Video.hh	1502
VideoVisual.hh	1503
VideoVisualPrivate.hh	1503

ViewController.hh	1504
Visual.hh	1505
VisualPrivate.hh	1506
WindowManager.hh	1507
WireBox.hh	1508
WireBoxPrivate.hh	1509
WirelessReceiver.hh	1509
WirelessTransceiver.hh	1510
WirelessTransmitter.hh	1511
World.hh	1512
WorldState.hh	1513
WrenchVisual.hh	1515
WrenchVisualPrivate.hh	1515

Chapter 8

Module Documentation

8.1 Common

Output a message.

Files

- file **CommonTypes.hh**

Namespaces

- namespace **gazebo::common**
Common namespace.

Classes

- class **gazebo::common::Animation**
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.
- class **gazebo::common::AssertionInternalError**
Class for generating Exceptions which come from gazebo assertions.
- class **gazebo::common::AudioDecoder**
An audio decoder based on FFmpeg.
- class **gazebo::common::BVHLoader**
Handles loading BVH animation files.
- class **gazebo::common::ColladaLoader**
Class used to load Collada mesh files.
- class **gazebo::common::Color**
Defines a color.
- class **gazebo::common::Console**
Container for loggers, and global logging options (such as verbose vs.
- class **gazebo::common::Exception**
Class for generating exceptions.
- class **gazebo::common::FileLogger**

- A logger that outputs messages to a file.*

 - class **gazebo::common::HeightmapData**
Encapsulates a generic heightmap data file.
 - class **gazebo::common::Image**
Encapsulates an image.
 - class **gazebo::common::ImageHeightmap**
Encapsulates an image that will be interpreted as a heightmap.
 - class **gazebo::common::InternalError**
Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs.
 - class **gazebo::common::KeyEvent**
Generic description of a keyboard event.
 - class **gazebo::common::KeyFrame**
A key frame in an animation.
 - class **gazebo::common::Logger**
Terminal logger.
 - class **gazebo::common::Material**
Encapsulates description of a material.
 - class **gazebo::common::Mesh**
A 3D mesh.
 - class **gazebo::common::MeshCSG**
Creates CSG meshes.
 - class **gazebo::common::MeshLoader**
Base class for loading meshes.
 - class **gazebo::common::MeshManager**
Maintains and manages all meshes.
 - class **gazebo::common::ModelDatabase**
Connects to model database, and has utility functions to find models.
 - class **gazebo::ModelPlugin**
*A plugin with access to **physics::Model** (p. 678).*
 - class **gazebo::common::MouseEvent**
Generic description of a mouse event.
 - class **gazebo::common::MovingWindowFilter**< T >
*Base class for **MovingWindowFilter** (p. 715).*
 - class **gazebo::common::MovingWindowFilterPrivate**< T >
 - class **gazebo::common::NodeAnimation**
Node animation.
 - class **gazebo::common::NodeAssignment**
Vertex to node weighted assignement for skeleton animation visualization.
 - class **gazebo::common::NodeTransform**
***NodeTransform** (p. 742) **Skeleton.hh** (p. 1465) *common/common.hh**
 - class **gazebo::common::NumericAnimation**
A numeric animation.
 - class **gazebo::common::NumericKeyFrame**
*A keyframe for a **NumericAnimation** (p. 752).*
 - class **gazebo::common::PID**
*Generic **PID** (p. 782) controller class.*
 - class **gazebo::PluginT**< T >

- A class which all plugins must inherit from.*
- class **gazebo::common::PoseAnimation**
 - A pose animation.*
- class **gazebo::common::PoseKeyFrame**
 - A keyframe for a **PoseAnimation** (p. 805).*
- class **gazebo::SensorPlugin**
 - A plugin with access to **physics::Sensor**.*
- class **SingletonT < T >**
 - Singleton template class.*
- class **gazebo::common::Skeleton**
 - A skeleton.*
- class **gazebo::common::SkeletonAnimation**
 - Skeleton** (p. 1024) animation.*
- class **gazebo::common::SkeletonNode**
 - A skeleton node.*
- class **gazebo::common::SphericalCoordinates**
 - Convert spherical coordinates for planetary surfaces.*
- class **gazebo::common::SphericalCoordinatesPrivate**
 - common/common.hh*
- class **gazebo::common::STLLoader**
 - Class used to load STL mesh files.*
- class **gazebo::common::SubMesh**
 - A child mesh.*
- class **gazebo::common::SystemPaths**
 - Functions to handle getting system paths, keeps track of:*
- class **gazebo::SystemPlugin**
 - A plugin loaded within the gzserver on startup.*
- class **gazebo::common::Time**
 - A **Time** (p. 1099) class, can be used to hold wall- or sim-time.*
- class **gazebo::common::Timer**
 - A timer class, used to time things in real world walltime.*
- class **gazebo::common::Video**
 - Handle video encoding and decoding using libavcodec.*
- class **gazebo::VisualPlugin**
 - A plugin loaded within the gzserver on startup.*
- class **gazebo::WorldPlugin**
 - A plugin with access to **physics::World** (p. 1239).*

Macros

- #define **gzdbg** (**gazebo::common::Console::dbg**(__FILE__, __LINE__))
 - Output a debug message.*
- #define **gzerr** (**gazebo::common::Console::err**(__FILE__, __LINE__))
 - Output an error message.*
- #define **gzlog** (**gazebo::common::Console::log**())
 - Output a message to a log file.*
- #define **gzLogInit**(_str) (**gazebo::common::Console::log**.Init(_str))

Initialize log file with filename given by `_str`.

- #define **gzmsg** (`gazebo::common::Console::msg()`)
- #define **gzthrow**(`msg`)

This macro logs an error to the throw stream and throws an exception that contains the file name and line number.

- #define **gzwarn** (`gazebo::common::Console::warn(__FILE__, __LINE__)`)
- Output a warning message.

Enumerations

- enum **gazebo::PluginType** {
gazebo::WORLD_PLUGIN, **gazebo::MODEL_PLUGIN**, **gazebo::SENSOR_PLUGIN**, **gazebo::SYSTEM_PLUGIN**,
gazebo::VISUAL_PLUGIN }

Used to specify the type of plugin.

Functions

- **gazebo::common::MovingWindowFilter**< `T` >::**MovingWindowFilter** ()
 Constructor.
- **gazebo::common::MovingWindowFilterPrivate**< `T` >::**MovingWindowFilterPrivate** ()
- virtual **gazebo::common::MovingWindowFilter**< `T` >::~~**MovingWindowFilter** ()
 Destructor.
- **GAZEBO_VISIBLE** void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)
 add path suffix to **common::SystemPaths** (p. 1092)
- void **gazebo::common::ModelDatabase::DownloadDependencies** (const std::string &_path)
 Download all dependencies for a give model path.
- **GAZEBO_VISIBLE** std::string **gazebo::common::find_file** (const std::string &_file)
 search for file in **common::SystemPaths** (p. 1092)
- **GAZEBO_VISIBLE** std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath)
 search for file in **common::SystemPaths** (p. 1092)
- **GAZEBO_VISIBLE** std::string **gazebo::common::find_file_path** (const std::string &_file)
 search for a file in **common::SystemPaths** (p. 1092)
- void **gazebo::common::ModelDatabase::Fini** ()
 Finalize the model database.
- `T` **gazebo::common::MovingWindowFilter**< `T` >::**Get** ()
 Get filtered result.
- template<typename `T` >
GAZEBO_VISIBLE std::string **gazebo::common::get_sha1** (const `T` &_buffer)
 Compute the SHA1 hash of an array of bytes.
- std::string **gazebo::common::ModelDatabase::GetDBConfig** (const std::string &_uri)
 Return the database.config file as a string.
- std::string **gazebo::common::ModelDatabase::GetModelConfig** (const std::string &_uri)
 Return the model.config file as a string.
- std::string **gazebo::common::ModelDatabase::GetModelFile** (const std::string &_uri)
 Get a model's SDF file based on a URI.
- std::string **gazebo::common::ModelDatabase::GetModelName** (const std::string &_uri)
 Get the name of a model based on a URI.

- `std::string gazebo::common::ModelDatabase::GetModelPath` (const std::string &_uri, bool _forceDownload=false)

Get the local path to a model.
- `std::map< std::string, std::string > gazebo::common::ModelDatabase::GetModels` ()

Returns the dictionary of all the model names.
- `event::ConnectionPtr gazebo::common::ModelDatabase::GetModels` (boost::function< void(const std::map< std::string, std::string > &)> _func)

Get the dictionary of all model names via a callback.
- `std::string gazebo::common::ModelDatabase::GetURI` ()

Returns the the global model database URI.
- `bool gazebo::common::MovingWindowFilter< T >::GetWindowFilled` () const

Get whether the window has been filled.
- `unsigned int gazebo::common::MovingWindowFilter< T >::GetWindowSize` () const

Get the window size.
- `bool gazebo::common::ModelDatabase::HasModel` (const std::string &_modelName)

Returns true if the model exists on the database.
- `GAZEBO_VISIBLE` void `gazebo::common::load` ()

Load the common library.
- `void gazebo::common::MovingWindowFilter< T >::SetWindowSize` (unsigned int _n)

Set window size.
- `void gazebo::common::ModelDatabase::Start` (bool _fetchImmediately=false)

Start the model database.
- `void gazebo::common::MovingWindowFilter< T >::Update` (T _val)

Update value of filter.

Variables

- static `std::string gazebo::common::PixelFormatNames` []

String names for the pixel formats.

8.1.1 Detailed Description

Output a message.

8.1.2 Macro Definition Documentation

8.1.2.1 #define gzdbg (gazebo::common::Console::dbg(_FILE_, _LINE_))

Output a debug message.

8.1.2.2 #define gzerr (gazebo::common::Console::err(_FILE_, _LINE_))

Output an error message.

Referenced by gazebo::transport::Connection::AsyncRead(), gazebo::rendering::GUIOverlay::ButtonCallback(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::physics::GearboxJoint< T >::Load(), gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::SimbodySphereShape::SetRadius(), gazebo::physics::SimbodyBoxShape::SetSize(), gazebo::physics::DARTCylinderShape::SetSize(), gazebo::physics::SimbodyCylinderShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

8.1.2.3 #define gzlog (gazebo::common::Console::log())

Output a message to a log file.

8.1.2.4 #define gzLogInit(_str) (gazebo::common::Console::log.Init(_str))

Initialize log file with filename given by _str.

If called twice, it will close currently in use and open a new log file.

Parameters

in	_str	Name of log file for gzlog messages.
----	------	--------------------------------------

8.1.2.5 #define gzmsg (gazebo::common::Console::msg())

8.1.2.6 #define gzthrow(msg)

Value:

```
{std::ostringstream throwStream;\
  throwStream << msg << std::endl << std::flush;\
  throw gazebo::common::Exception(__FILE__, __LINE__, throwStream.str()); }
```

This macro logs an error to the throw stream and throws an exception that contains the file name and line number.

Referenced by gazebo::transport::TopicManager::Advertise(), gazebo::transport::CallbackHelperT< M >::GetMsgType(), and gazebo::transport::SubscribeOptions::Init().

8.1.2.7 #define gzwarn (gazebo::common::Console::warn(__FILE__, __LINE__))

Output a warning message.

Referenced by gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::SimbodySphereShape::SetRadius(), gazebo::physics::SimbodyBoxShape::SetSize(), gazebo::physics::DARTCylinderShape::SetSize(), gazebo::physics::SimbodyCylinderShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

8.1.3 Enumeration Type Documentation

8.1.3.1 enum gazebo::PluginType

Used to specify the type of plugin.

Enumerator:

WORLD_PLUGIN A World plugin.

MODEL_PLUGIN A Model plugin.

SENSOR_PLUGIN A Sensor plugin.

SYSTEM_PLUGIN A System plugin.

VISUAL_PLUGIN A Visual plugin.

8.1.4 Function Documentation

8.1.4.1 `template<typename T > gazebo::common::MovingWindowFilter< T >::MovingWindowFilter ()`

Constructor.

8.1.4.2 `template<typename T > gazebo::common::MovingWindowFilterPrivate< T >::MovingWindowFilterPrivate ()`

FIXME hardcoded initial value for now

8.1.4.3 `template<typename T > gazebo::common::MovingWindowFilter< T >::~~MovingWindowFilter ()`
`[virtual]`

Destructor.

References NULL.

8.1.4.4 **GAZEBO_VISIBLE** `void gazebo::common::add_search_path_suffix (const std::string & _suffix)`

add path suffix to **common::SystemPaths** (p. 1092)

Parameters

<code>in</code>	<code>_suffix</code>	The suffix to add.
-----------------	----------------------	--------------------

8.1.4.5 `void gazebo::common::ModelDatabase::DownloadDependencies (const std::string & _path)`

Download all dependencies for a give model path.

Look's in the model's manifest file (`_path/model.config`) for all models listed in the `<depend>` block, and downloads the models if necessary.

Parameters

<code>in</code>	<code>_path</code>	Path to a model.
-----------------	--------------------	------------------

8.1.4.6 **GAZEBO_VISIBLE** `std::string gazebo::common::find_file (const std::string & _file)`

search for file in **common::SystemPaths** (p. 1092)

Parameters

<code>in</code>	<code>_file</code>	Name of the file to find.
-----------------	--------------------	---------------------------

Returns

The path containing the file.

8.1.4.7 GAZEBO_VISIBLE std::string gazebo::common::find_file (const std::string & *_file*, bool *_searchLocalPath*)

search for file in **common::SystemPaths** (p. 1092)

Parameters

in	<i>_file</i>	Name of the file to find.
in	<i>_searchLocalPath</i>	True to search in the current working directory.

Returns

The path containing the file.

8.1.4.8 GAZEBO_VISIBLE std::string gazebo::common::find_file_path (const std::string & *_file*)

search for a file in **common::SystemPaths** (p. 1092)

Parameters

in	<i>_file</i>	the file name to look for.
----	--------------	----------------------------

Returns

The path containing the file.

8.1.4.9 void gazebo::common::ModelDatabase::Fini ()

Finalize the model database.

8.1.4.10 template<typename T> T gazebo::common::MovingWindowFilter< T >::Get ()

Get filtered result.

Returns

latest filtered value

8.1.4.11 template<typename T> GAZEBO_VISIBLE std::string gazebo::common::get_sha1 (const T & *_buffer*)

Compute the SHA1 hash of an array of bytes.

Parameters

in	<i>_buffer</i>	Input sequence. The permitted data types for this function are std::string and any STL container.
----	----------------	---

Returns

The string representation (40 character) of the SHA1 hash.

References NULL.

8.1.4.12 `std::string gazebo::common::ModelDatabase::GetDBConfig (const std::string & _uri)`

Return the database.config file as a string.

Returns

The database config file from the model database.

8.1.4.13 `std::string gazebo::common::ModelDatabase::GetModelConfig (const std::string & _uri)`

Return the model.config file as a string.

Returns

The model config file from the model database.

8.1.4.14 `std::string gazebo::common::ModelDatabase::GetModelFile (const std::string & _uri)`

Get a model's SDF file based on a URI.

Get a model file based on a URI. If the model is on a remote server, then the model fetched and installed locally.

Parameters

<code>in</code>	<code>_uri</code>	The URI of the model
-----------------	-------------------	----------------------

Returns

The full path and filename to the SDF file

8.1.4.15 `std::string gazebo::common::ModelDatabase::GetModelName (const std::string & _uri)`

Get the name of a model based on a URI.

The URI must be fully qualified: `http://gazebosim.org/gazebo_models/ground_plane` or `model-://gazebo_models`

Parameters

<code>in</code>	<code>_uri</code>	the model uri
-----------------	-------------------	---------------

Returns

the model's name.

8.1.4.16 `std::string gazebo::common::ModelDatabase::GetModelPath (const std::string & _uri, bool _forceDownload = false)`

Get the local path to a model.

Get the path to a model based on a URI. If the model is on a remote server, then the model fetched and installed locally.

Parameters

in	<code>_uri</code>	the model uri
in	<code>_forceDownload</code>	True to skip searching local paths.

Returns

path to a model directory

8.1.4.17 `std::map<std::string, std::string> gazebo::common::ModelDatabase::GetModels ()`

Returns the dictionary of all the model names.

This is a blocking call. Which means it will wait for the **ModelDatabase** (p. 693) to download the model list.

Returns

a map of model names, indexed by their full URI.

8.1.4.18 `event::ConnectionPtr gazebo::common::ModelDatabase::GetModels (boost::function< void(const std::map< std::string, std::string > &)> _func)`

Get the dictionary of all model names via a callback.

This is the non-blocking version of **ModelDatabase::GetModels** (p. 44)

Parameters

in	<code>_func</code>	Callback function that receives the list of models.
----	--------------------	---

Returns

A boost shared pointer. This pointer must remain valid in order to receive the callback.

8.1.4.19 `std::string gazebo::common::ModelDatabase::GetURI ()`

Returns the the global model database URI.

Returns

the URI.

8.1.4.20 `template<typename T> bool gazebo::common::MovingWindowFilter< T >::GetWindowFilled () const`

Get whether the window has been filled.

Returns

True if the window has been filled.

8.1.4.21 `template<typename T> unsigned int gazebo::common::MovingWindowFilter< T >::GetWindowSize () const`

Get the window size.

Returns

The size of the moving window.

8.1.4.22 `bool gazebo::common::ModelDatabase::HasModel (const std::string & _modelName)`

Returns true if the model exists on the database.

Parameters

<code>in</code>	<code>_modelName</code>	URI of the model (eg: model://my_model_name).
-----------------	-------------------------	---

Returns

True if the model was found.

8.1.4.23 `GAZEBO_VISIBLE void gazebo::common::load ()`

Load the common library.

8.1.4.24 `template<typename T> void gazebo::common::MovingWindowFilter< T >::SetWindowSize (unsigned int _n)`

Set window size.

Parameters

<code>in</code>	<code>_n</code>	new desired window size
-----------------	-----------------	-------------------------

8.1.4.25 `void gazebo::common::ModelDatabase::Start (bool _fetchImmediately = false)`

Start the model database.

Parameters

<code>in</code>	<code>_fetchImmediately</code>	True to fetch the models without waiting.
-----------------	--------------------------------	---

8.1.4.26 `template<typename T> void gazebo::common::MovingWindowFilter< T >::Update (T _val)`

Update value of filter.

Parameters

in	<i>_val</i>	new raw value
----	-------------	---------------

8.1.5 Variable Documentation

8.1.5.1 `std::string gazebo::common::PixelFormatNames[]` [static]

Initial value:

```
{
  "UNKNOWN_PIXEL_FORMAT",
  "L_INT8",
  "L_INT16",
  "RGB_INT8",
  "RGBA_INT8",
  "BGRA_INT8",
  "RGB_INT16",
  "RGB_INT32",
  "BGR_INT8",
  "BGR_INT16",
  "BGR_INT32",
  "R_FLOAT16",
  "RGB_FLOAT16",
  "R_FLOAT32",
  "RGB_FLOAT32",
  "BAYER_RGGB8",
  "BAYER_RGGR8",
  "BAYER_GBRG8",
  "BAYER_GRBG8"
}
```

String names for the pixel formats.

See Also

Image::PixelFormat (p. 516).

8.2 Events

Namespaces

- namespace **gazebo::event**
Event (p. 416) namespace.

Classes

- class **gazebo::event::Connection**
A class that encapsulates a connection.
- class **gazebo::event::ConnectionPrivate**
- class **gazebo::event::Event**
Base class for all events.
- class **gazebo::event::EventPrivate**
- class **gazebo::event::Events**
*An **Event** (p. 416) class to get notifications for simulator events.*
- class **gazebo::event::EventT< T >**
A class for event processing.
- class **gazebo::event::EventTPrivate< T >**

Functions

- **gazebo::event::EventT< T >::EventT ()**
Constructor.
- virtual **gazebo::event::EventT< T >::~~EventT ()**
Destructor.
- ConnectionPtr **gazebo::event::EventT< T >::Connect** (const boost::function< T > &_subscriber)
Connect a callback to this event.
- unsigned int **gazebo::event::EventT< T >::ConnectionCount ()** const
Get the number of connections.
- virtual void **gazebo::event::EventT< T >::Disconnect** (ConnectionPtr _c)
Disconnect a callback to this event.
- virtual void **gazebo::event::EventT< T >::Disconnect** (int _id)
Disconnect a callback to this event.

8.2.1 Detailed Description

8.2.2 Function Documentation

8.2.2.1 `template<typename T > gazebo::event::EventT< T >::EventT ()`

Constructor.

References `gazebo::event::Event::dataPtr`.

8.2.2.2 `template<typename T> gazebo::event::EventT< T >::~~EventT () [virtual]`

Destructor.

Destructor. Deletes all the associated connections.

8.2.2.3 `template<typename T> ConnectionPtr gazebo::event::EventT< T >::Connect (const boost::function< T > & _subscriber)`

Connect a callback to this event.

Adds a connection.

Parameters

<code>in</code>	<code>_subscriber</code>	Pointer to a callback function.
-----------------	--------------------------	---------------------------------

Returns

A **Connection** (p.263) object, which will automatically call Disconnect when it goes out of scope.

Parameters

<code>in</code>	<code>_subscriber</code>	the subscriber to connect.
-----------------	--------------------------	----------------------------

8.2.2.4 `template<typename T> unsigned int gazebo::event::EventT< T >::ConnectionCount () const`

Get the number of connections.

Returns

Number of connection to this **Event** (p.416).

Number of connections.

8.2.2.5 `template<typename T> void gazebo::event::EventT< T >::Disconnect (ConnectionPtr _c) [virtual]`

Disconnect a callback to this event.

Removes a connection.

Parameters

<code>in</code>	<code>_c</code>	The connection to disconnect.
<code>in</code>	<code>_c</code>	the connection.

Implements **gazebo::event::Event** (p.418).

References NULL.

8.2.2.6 `template<typename T> void gazebo::event::EventT< T >::Disconnect (int _id) [virtual]`

Disconnect a callback to this event.

Removes a connection.

Parameters

<code>in</code>	<code>_id</code>	The id of the connection to disconnect.
<code>in</code>	<code>_id</code>	the connection index.

Implements **gazebo::event::Event** (p. 418).

8.3 Math

A set of classes that encapsulate math related properties and functions.

Files

- file **MathTypes.hh**
Forward declarations for the math classes.

Namespaces

- namespace **gazebo::math**
Math namespace.

Classes

- class **gazebo::math::Angle**
An angle and related functions.
- class **gazebo::math::Box**
Mathematical representation of a box and related functions.
- class **gazebo::math::Matrix3**
A 3x3 matrix class.
- class **gazebo::math::Matrix4**
A 3x3 matrix class.
- class **gazebo::math::Plane**
A plane and related functions.
- class **gazebo::math::Pose**
Encapsulates a position and rotation in three space.
- class **gazebo::math::Quaternion**
A quaternion class.
- class **gazebo::math::Rand**
Random number generator class.
- class **gazebo::math::RotationSpline**
***Spline** (p. 1063) for rotations.*
- class **gazebo::math::Spline**
Splines.
- class **gazebo::math::Vector2d**
Generic double x , y vector.
- class **gazebo::math::Vector2i**
Generic integer x , y vector.
- class **gazebo::math::Vector3**
*The **Vector3** (p. 1165) class represents the generic vector containing 3 elements.*
- class **gazebo::math::Vector4**
double Generic x , y , z , w vector

Functions

- `template<typename T >`
T gazebo::math::clamp (T _v, T _min, T _max)
Simple clamping function.
- `template<typename T >`
bool gazebo::math::equal (const T &_a, const T &_b, const T &_epsilon=1e-6)
check if two values are equal, within a tolerance
- `float gazebo::math::fixnan` (float _v)
Fix a nan value.
- `double gazebo::math::fixnan` (double _v)
Fix a nan value.
- `bool gazebo::math::isnan` (float _v)
check if a float is NaN
- `bool gazebo::math::isnan` (double _v)
check if a double is NaN
- `bool gazebo::math::isPowerOfTwo` (unsigned int _x)
is this a power of 2?
- `template<typename T >`
T gazebo::math::max (const std::vector< T > &_values)
get the maximum value of vector of values
- `template<typename T >`
T gazebo::math::mean (const std::vector< T > &_values)
get mean of vector of values
- `template<typename T >`
T gazebo::math::min (const std::vector< T > &_values)
get the minimum value of vector of values
- `double gazebo::math::parseFloat` (const std::string &_input)
parse string into float
- `int gazebo::math::parseInt` (const std::string &_input)
parse string into an integer
- `template<typename T >`
T gazebo::math::precision (const T &_a, const unsigned int &_precision)
get value at a specified precision
- `unsigned int gazebo::math::roundUpPowerOfTwo` (unsigned int _x)
Get the smallest power of two that is greater or equal to a given value.
- `template<typename T >`
T gazebo::math::variance (const std::vector< T > &_values)
get variance of vector of values

Variables

- `static const double gazebo::math::NAN_D` = `std::numeric_limits<double>::quiet_NaN()`
Returns the representation of a quiet not a number (NaN)
- `static const int gazebo::math::NAN_I` = `std::numeric_limits<int>::quiet_NaN()`
Returns the representation of a quiet not a number (NaN)

8.3.1 Detailed Description

A set of classes that encapsulate math related properties and functions.

8.3.2 Function Documentation

8.3.2.1 `template<typename T > T gazebo::math::clamp (T _v, T _min, T _max) [inline]`

Simple clamping function.

Parameters

in	<code>_v</code>	value
in	<code>_min</code>	minimum
in	<code>_max</code>	maximum

References `gazebo::math::max()`, and `gazebo::math::min()`.

8.3.2.2 `template<typename T > bool gazebo::math::equal (const T & _a, const T & _b, const T & _epsilon = 1e-6) [inline]`

check if two values are equal, within a tolerance

Parameters

in	<code>_a</code>	the first value
in	<code>_b</code>	the second value
in	<code>_epsilon</code>	the tolerance

Referenced by `gazebo::math::Quaternion::Correct()`, `gazebo::math::Quaternion::GetInverse()`, `gazebo::physics::DARTSphereShape::SetRadius()`, `gazebo::physics::SimbodySphereShape::SetRadius()`, `gazebo::physics::SimbodyBoxShape::SetSize()`, `gazebo::physics::DARTCylinderShape::SetSize()`, `gazebo::physics::SimbodyCylinderShape::SetSize()`, and `gazebo::physics::DARTBoxShape::SetSize()`.

8.3.2.3 `float gazebo::math::fixnan (float _v) [inline]`

Fix a nan value.

Parameters

in	<code>_v</code>	Value to correct.
----	-----------------	-------------------

Returns

0 if `_v` is NaN, `_v` otherwise.

References `gazebo::math::isnan()`.

8.3.2.4 `double gazebo::math::fixnan (double _v) [inline]`

Fix a nan value.

Parameters

in	_v	Value to correct.
----	----	-------------------

Returns

0 if _v is NaN, _v otherwise.

References gazebo::math::isnan().

8.3.2.5 bool gazebo::math::isnan (float _v) [inline]

check if a float is NaN

Parameters

in	_v	the value
----	----	-----------

Returns

true if _v is not a number, false otherwise

Referenced by gazebo::math::fixnan(), and gazebo::math::isnan().

8.3.2.6 bool gazebo::math::isnan (double _v) [inline]

check if a double is NaN

Parameters

in	_v	the value
----	----	-----------

Returns

true if _v is not a number, false otherwise

References gazebo::math::isnan().

8.3.2.7 bool gazebo::math::isPowerOfTwo (unsigned int _x) [inline]

is this a power of 2?

Parameters

in	_x	the number
----	----	------------

Returns

true if _x is a power of 2, false otherwise

Referenced by gazebo::math::roundUpPowerOfTwo().

8.3.2.8 `template<typename T> T gazebo::math::max (const std::vector< T > & _values) [inline]`

get the maximum value of vector of values

Parameters

<code>in</code>	<code>_values</code>	the vector of values
-----------------	----------------------	----------------------

Returns

maximum

References `gazebo::math::min()`.

Referenced by `gazebo::math::clamp()`, and `gazebo::math::min()`.

8.3.2.9 `template<typename T> T gazebo::math::mean (const std::vector< T > & _values) [inline]`

get mean of vector of values

Parameters

<code>in</code>	<code>_values</code>	the vector of values
-----------------	----------------------	----------------------

Returns

the mean

8.3.2.10 `template<typename T> T gazebo::math::min (const std::vector< T > & _values) [inline]`

get the minimum value of vector of values

Parameters

<code>in</code>	<code>_values</code>	the vector of values
-----------------	----------------------	----------------------

Returns

minimum

References `gazebo::math::max()`.

Referenced by `gazebo::math::clamp()`, and `gazebo::math::max()`.

8.3.2.11 `double gazebo::math::parseFloat (const std::string & _input) [inline]`

parse string into float

Parameters

<code>_input</code>	the string
---------------------	------------

Returns

a floating point number (can be NaN) or 0 with a message in the error stream

References gazebo::math::NAN_D.

8.3.2.12 `int gazebo::math::parseInt (const std::string & _input) [inline]`

parse string into an integer

Parameters

<code>in</code>	<code><i>_input</i></code>	the string
-----------------	----------------------------	------------

Returns

an integer, 0 or 0 and a message in the error stream

References gazebo::math::NAN_I.

8.3.2.13 `template<typename T> T gazebo::math::precision (const T & _a, const unsigned int & _precision) [inline]`

get value at a specified precision

Parameters

<code>in</code>	<code><i>_a</i></code>	the number
<code>in</code>	<code><i>_precision</i></code>	the precision

Returns

the value for the specified precision

8.3.2.14 `unsigned int gazebo::math::roundUpPowerOfTwo (unsigned int _x) [inline]`

Get the smallest power of two that is greater or equal to a given value.

Parameters

<code>in</code>	<code><i>_x</i></code>	the number
-----------------	------------------------	------------

Returns

the same value if `_x` is already a power of two. Otherwise, it returns the smallest power of two that is greater than `_x`

References gazebo::math::isPowerOfTwo().

8.3.2.15 `template<typename T> T gazebo::math::variance (const std::vector< T > & _values) [inline]`

get variance of vector of values

Parameters

<code>in</code>	<code>_values</code>	the vector of values
-----------------	----------------------	----------------------

Returns

the squared deviation

8.3.3 Variable Documentation

8.3.3.1 `const double gazebo::math::NAN_D = std::numeric_limits<double>::quiet_NaN()` [static]

Returns the representation of a quiet not a number (NaN)

Referenced by `gazebo::math::parseFloat()`.

8.3.3.2 `const int gazebo::math::NAN_I = std::numeric_limits<int>::quiet_NaN()` [static]

Returns the representation of a quiet not a number (NaN)

Referenced by `gazebo::math::parseInt()`.

8.4 Messages

All messages and helper functions.

Namespaces

- namespace **gazebo::msgs**
Messages namespace.

Classes

- class **google::protobuf::compiler::cpp::GazeboGenerator**
*Google protobuf message generator for **gazebo::msgs** (p. 113).*
- class **gazebo::msgs::MsgFactory**
A factory that generates protobuf message based on a string type.

Macros

- #define **GZ_REGISTER_STATIC_MSG**(_msgtype, _classname)
Static message registration macro.

Functions

- **GAZEBO_VISIBLE** msgs::Vector3d **gazebo::msgs::Convert** (const math::Vector3 &_v)
*Convert a **math::Vector3** (p. 1165) to a msgs::Vector3d.*
- **GAZEBO_VISIBLE** msgs::Quaternion **gazebo::msgs::Convert** (const math::Quaternion &_q)
*Convert a **math::Quaternion** (p. 824) to a msgs::Quaternion.*
- **GAZEBO_VISIBLE** msgs::Pose **gazebo::msgs::Convert** (const math::Pose &_p)
*Convert a **math::Pose** (p. 797) to a msgs::Pose.*
- **GAZEBO_VISIBLE** msgs::Color **gazebo::msgs::Convert** (const common::Color &_c)
*Convert a **common::Color** (p. 249) to a msgs::Color.*
- **GAZEBO_VISIBLE** msgs::Time **gazebo::msgs::Convert** (const common::Time &_t)
*Convert a **common::Time** (p. 1099) to a msgs::Time.*
- **GAZEBO_VISIBLE** msgs::PlaneGeom **gazebo::msgs::Convert** (const math::Plane &_p)
*Convert a **math::Plane** (p. 788) to a msgs::PlaneGeom.*
- **GAZEBO_VISIBLE** math::Vector3 **gazebo::msgs::Convert** (const msgs::Vector3d &_v)
Convert a msgs::Vector3d to a math::Vector.
- **GAZEBO_VISIBLE** math::Quaternion **gazebo::msgs::Convert** (const msgs::Quaternion &_q)
*Convert a msgs::Quaternion to a **math::Quaternion** (p. 824).*
- **GAZEBO_VISIBLE** math::Pose **gazebo::msgs::Convert** (const msgs::Pose &_p)
*Convert a msgs::Pose to a **math::Pose** (p. 797).*
- **GAZEBO_VISIBLE** common::Color **gazebo::msgs::Convert** (const msgs::Color &_c)
*Convert a msgs::Color to a **common::Color** (p. 249).*
- **GAZEBO_VISIBLE** common::Time **gazebo::msgs::Convert** (const msgs::Time &_t)
*Convert a msgs::Time to a **common::Time** (p. 1099).*
- **GAZEBO_VISIBLE** math::Plane **gazebo::msgs::Convert** (const msgs::PlaneGeom &_p)

Convert a `msgs::PlaneGeom` to a `common::Plane`.

- **GAZEBO_VISIBLE** `msgs::Request * gazebo::msgs::CreateRequest` (`const std::string &_request, const std::string &_data=""`)
Create a request message.
- **GAZEBO_VISIBLE** `msgs::Fog gazebo::msgs::FogFromSDF` (`sdf::ElementPtr _sdf`)
Create a `msgs::Fog` from a fog SDF element.
- **GAZEBO_VISIBLE** `msgs::Geometry gazebo::msgs::GeometryFromSDF` (`sdf::ElementPtr _sdf`)
Create a `msgs::Geometry` from a geometry SDF element.
- **GAZEBO_VISIBLE** `msgs::Header * gazebo::msgs::GetHeader` (`google::protobuf::Message &_message`)
Get the header from a protobuf message.
- **GAZEBO_VISIBLE** `msgs::GUI gazebo::msgs::GUIFromSDF` (`sdf::ElementPtr _sdf`)
Create a `msgs::GUI` from a GUI SDF element.
- **GAZEBO_VISIBLE** `void gazebo::msgs::Init` (`google::protobuf::Message &_message, const std::string &_id=""`)
Initialize a message.
- **GAZEBO_VISIBLE** `msgs::Light gazebo::msgs::LightFromSDF` (`sdf::ElementPtr _sdf`)
Create a `msgs::Light` from a light SDF element.
- **GAZEBO_VISIBLE** `msgs::MeshGeom gazebo::msgs::MeshFromSDF` (`sdf::ElementPtr _sdf`)
Create a `msgs::MeshGeom` from a mesh SDF element.
- **GAZEBO_VISIBLE** `msgs::Scene gazebo::msgs::SceneFromSDF` (`sdf::ElementPtr _sdf`)
Create a `msgs::Scene` from a scene SDF element.
- **GAZEBO_VISIBLE** `void gazebo::msgs::Set` (`common::Image &_img, const msgs::Image &_msg`)
Convert a `msgs::Image` to a `common::Image` (p. 515).
- **GAZEBO_VISIBLE** `void gazebo::msgs::Set` (`msgs::Image * _msg, const common::Image &_i`)
Set a `msgs::Image` from a `common::Image` (p. 515).
- **GAZEBO_VISIBLE** `void gazebo::msgs::Set` (`msgs::Vector3d * _pt, const math::Vector3 &_v`)
Set a `msgs::Vector3d` from a `math::Vector3` (p. 1165).
- **GAZEBO_VISIBLE** `void gazebo::msgs::Set` (`msgs::Vector2d * _pt, const math::Vector2d &_v`)
Set a `msgs::Vector2d` from a `math::Vector3` (p. 1165).
- **GAZEBO_VISIBLE** `void gazebo::msgs::Set` (`msgs::Quaternion * _q, const math::Quaternion &_v`)
Set a `msgs::Quaternion` from a `math::Quaternion` (p. 824).
- **GAZEBO_VISIBLE** `void gazebo::msgs::Set` (`msgs::Pose * _p, const math::Pose &_v`)
Set a `msgs::Pose` from a `math::Pose` (p. 797).
- **GAZEBO_VISIBLE** `void gazebo::msgs::Set` (`msgs::Color * _c, const common::Color &_v`)
Set a `msgs::Color` from a `common::Color` (p. 249).
- **GAZEBO_VISIBLE** `void gazebo::msgs::Set` (`msgs::Time * _t, const common::Time &_v`)
Set a `msgs::Time` from a `common::Time` (p. 1099).
- `void gazebo::msgs::Set` (`msgs::SphericalCoordinates * _s, const common::SphericalCoordinates &_v`)
Set a `msgs::SphericalCoordinates` from a `common::SphericalCoordinates` (p. 1057) object.
- **GAZEBO_VISIBLE** `void gazebo::msgs::Set` (`msgs::PlaneGeom * _p, const math::Plane &_v`)
Set a `msgs::Plane` from a `math::Plane` (p. 788).
- **GAZEBO_VISIBLE** `void gazebo::msgs::Stamp` (`msgs::Header * _header`)
Time stamp a header.
- **GAZEBO_VISIBLE** `void gazebo::msgs::Stamp` (`msgs::Time * _time`)
Set the time in a time message.
- **GAZEBO_VISIBLE** `msgs::TrackVisual gazebo::msgs::TrackVisualFromSDF` (`sdf::ElementPtr _sdf`)
Create a `msgs::TrackVisual` from a track visual SDF element.
- **GAZEBO_VISIBLE** `msgs::Visual gazebo::msgs::VisualFromSDF` (`sdf::ElementPtr _sdf`)
Create a `msgs::Visual` from a visual SDF element.

8.4.1 Detailed Description

All messages and helper functions.

8.4.2 Macro Definition Documentation

8.4.2.1 #define GZ_REGISTER_STATIC_MSG(*_msgtype*, *_classname*)

Value:

```
GAZEBO_VISIBLE \
  boost::shared_ptr<google::protobuf::Message> New##_classname() \
  { \
    return boost::shared_ptr<gazebo::msgs::_classname>(\
      new gazebo::msgs::_classname); \
  } \
  class GAZEBO_VISIBLE Msg##_classname \
  { \
  public: Msg##_classname() \
    { gazebo::msgs::MsgFactory::RegisterMsg(_msgtype, New##_classname); } \
  }; \
  static Msg##_classname GzMsgInitializer;
```

Static message registration macro.

Use this macro to register messages.

Parameters

in	<i>_msgtype</i>	Message type name.
in	<i>_classname</i>	Class name for message.

8.4.3 Function Documentation

8.4.3.1 GAZEBO_VISIBLE msgs::Vector3d gazebo::msgs::Convert (const math::Vector3 & *_v*)

Convert a **math::Vector3** (p. 1165) to a msgs::Vector3d.

Parameters

in	<i>_v</i>	The vector to convert
----	-----------	-----------------------

Returns

A msgs::Vector3d object

8.4.3.2 GAZEBO_VISIBLE msgs::Quaternion gazebo::msgs::Convert (const math::Quaternion & *_q*)

Convert a **math::Quaternion** (p. 824) to a msgs::Quaternion.

Parameters

in	<i>_q</i>	The quaternion to convert
----	-----------	---------------------------

Returns

A `msgs::Quaternion` object

8.4.3.3 GAZEBO_VISIBLE `msgs::Pose` `gazebo::msgs::Convert` (`const math::Pose & _p`)

Convert a `math::Pose` (p. 797) to a `msgs::Pose`.

Parameters

<code>in</code>	<code>_p</code>	The pose to convert
-----------------	-----------------	---------------------

Returns

A `msgs::Pose` object

8.4.3.4 GAZEBO_VISIBLE `msgs::Color` `gazebo::msgs::Convert` (`const common::Color & _c`)

Convert a `common::Color` (p. 249) to a `msgs::Color`.

Parameters

<code>in</code>	<code>_c</code>	The color to convert
-----------------	-----------------	----------------------

Returns

A `msgs::Color` object

8.4.3.5 GAZEBO_VISIBLE `msgs::Time` `gazebo::msgs::Convert` (`const common::Time & _t`)

Convert a `common::Time` (p. 1099) to a `msgs::Time`.

Parameters

<code>in</code>	<code>_t</code>	The time to convert
-----------------	-----------------	---------------------

Returns

A `msgs::Time` object

8.4.3.6 GAZEBO_VISIBLE `msgs::PlaneGeom` `gazebo::msgs::Convert` (`const math::Plane & _p`)

Convert a `math::Plane` (p. 788) to a `msgs::PlaneGeom`.

Parameters

<code>in</code>	<code>_p</code>	The plane to convert
-----------------	-----------------	----------------------

Returns

A `msgs::PlaneGeom` object

8.4.3.7 GAZEBO_VISIBLE `math::Vector3 gazebo::msgs::Convert (const msgs::Vector3d & _v)`

Convert a `msgs::Vector3d` to a `math::Vector`.

Parameters

<code>in</code>	<code>_v</code>	The plane to convert
-----------------	-----------------	----------------------

Returns

A `math::Vector3` (p. 1165) object

8.4.3.8 GAZEBO_VISIBLE `math::Quaternion gazebo::msgs::Convert (const msgs::Quaternion & _q)`

Convert a `msgs::Quaternion` to a `math::Quaternion` (p. 824).

Parameters

<code>in</code>	<code>_q</code>	The quaternion to convert
-----------------	-----------------	---------------------------

Returns

A `math::Quaternion` (p. 824) object

8.4.3.9 GAZEBO_VISIBLE `math::Pose gazebo::msgs::Convert (const msgs::Pose & _p)`

Convert a `msgs::Pose` to a `math::Pose` (p. 797).

Parameters

<code>in</code>	<code>_p</code>	The pose to convert
-----------------	-----------------	---------------------

Returns

A `math::Pose` (p. 797) object

8.4.3.10 GAZEBO_VISIBLE `common::Color gazebo::msgs::Convert (const msgs::Color & _c)`

Convert a `msgs::Color` to a `common::Color` (p. 249).

Parameters

<code>in</code>	<code>_c</code>	The color to convert
-----------------	-----------------	----------------------

Returns

A **common::Color** (p. 249) object

8.4.3.11 GAZEBO_VISIBLE common::Time gazebo::msgs::Convert (const msgs::Time & _t)

Convert a msgs::Time to a **common::Time** (p. 1099).

Parameters

<i>in</i>	<i>_t</i>	The time to convert
-----------	-----------	---------------------

Returns

A **common::Time** (p. 1099) object

8.4.3.12 GAZEBO_VISIBLE math::Plane gazebo::msgs::Convert (const msgs::PlaneGeom & _p)

Convert a msgs::PlaneGeom to a common::Plane.

Parameters

<i>in</i>	<i>_p</i>	The plane to convert
-----------	-----------	----------------------

Returns

A common::Plane object

8.4.3.13 GAZEBO_VISIBLE msgs::Request* gazebo::msgs::CreateRequest (const std::string & _request, const std::string & _data = " ")

Create a request message.

Parameters

<i>in</i>	<i>_request</i>	Request string
<i>in</i>	<i>_data</i>	Optional data string

Returns

A Request message

8.4.3.14 GAZEBO_VISIBLE msgs::Fog gazebo::msgs::FogFromSDF (sdf::ElementPtr _sdf)

Create a msgs::Fog from a fog SDF element.

Parameters

<i>in</i>	<i>_sdf</i>	The sdf element
-----------	-------------	-----------------

Returns

The new msgs::Fog object

8.4.3.15 GAZEBO_VISIBLE msgs::Geometry gazebo::msgs::GeometryFromSDF (sdf::ElementPtr _sdf)

Create a msgs::Geometry from a geometry SDF element.

Parameters

in	<i>_sdf</i>	The sdf element
----	-------------	-----------------

Returns

The new msgs::Geometry object

8.4.3.16 GAZEBO_VISIBLE msgs::Header* gazebo::msgs::GetHeader (google::protobuf::Message & _message)

Get the header from a protobuf message.

Parameters

in	<i>_message</i>	A google protobuf message
----	-----------------	---------------------------

Returns

A pointer to the message's header

8.4.3.17 GAZEBO_VISIBLE msgs::GUI gazebo::msgs::GUIFromSDF (sdf::ElementPtr _sdf)

Create a msgs::GUI from a GUI SDF element.

Parameters

in	<i>_sdf</i>	The sdf element
----	-------------	-----------------

Returns

The new msgs::GUI object

8.4.3.18 GAZEBO_VISIBLE void gazebo::msgs::Init (google::protobuf::Message & _message, const std::string & _id = " ")

Initialize a message.

Parameters

in	<i>_message</i>	Message to initialize
in	<i>_id</i>	Optional string id

Referenced by gazebo::physics::HingeJoint< SimbodyJoint >::Init(), gazebo::physics::BallJoint< SimbodyJoint >::Init(), gazebo::physics::UniversalJoint< SimbodyJoint >::Init(), gazebo::physics::GearboxJoint< T >::Init(), and gazebo::physics::ScrewJoint< SimbodyJoint >::Init().

8.4.3.19 GAZEBO_VISIBLE msgs::Light gazebo::msgs::LightFromSDF (sdf::ElementPtr _sdf)

Create a msgs::Light from a light SDF element.

Parameters

in	<i>_sdf</i>	The sdf element
----	-------------	-----------------

Returns

The new msgs::Light object

8.4.3.20 GAZEBO_VISIBLE msgs::MeshGeom gazebo::msgs::MeshFromSDF (sdf::ElementPtr _sdf)

Create a msgs::MeshGeom from a mesh SDF element.

Parameters

in	<i>_sdf</i>	The sdf element
----	-------------	-----------------

Returns

The new msgs::MeshGeom object

8.4.3.21 GAZEBO_VISIBLE msgs::Scene gazebo::msgs::SceneFromSDF (sdf::ElementPtr _sdf)

Create a msgs::Scene from a scene SDF element.

Parameters

in	<i>_sdf</i>	The sdf element
----	-------------	-----------------

Returns

The new msgs::Scene object

8.4.3.22 GAZEBO_VISIBLE void gazebo::msgs::Set (common::Image & _img, const msgs::Image & _msg)

Convert a msgs::Image to a **common::Image** (p. 515).

Parameters

out	<i>_img</i>	The common::Image (p. 515) container
in	<i>_msg</i>	The Image message to convert

8.4.3.23 **GAZEBO_VISIBLE** void gazebo::msgs::Set (msgs::Image * *_msg*, const common::Image & *_i*)

Set a msgs::Image from a **common::Image** (p. 515).

Parameters

out	<i>_msg</i>	A msgs::Image pointer
in	<i>_i</i>	A common::Image (p. 515) reference

8.4.3.24 **GAZEBO_VISIBLE** void gazebo::msgs::Set (msgs::Vector3d * *_pt*, const math::Vector3 & *_v*)

Set a msgs::Vector3d from a **math::Vector3** (p. 1165).

Parameters

out	<i>_pt</i>	A msgs::Vector3d pointer
in	<i>_v</i>	A math::Vector3 (p. 1165) reference

8.4.3.25 **GAZEBO_VISIBLE** void gazebo::msgs::Set (msgs::Vector2d * *_pt*, const math::Vector2d & *_v*)

Set a msgs::Vector2d from a **math::Vector3** (p. 1165).

Parameters

out	<i>_pt</i>	A msgs::Vector2d pointer
in	<i>_v</i>	A math::Vector2d (p. 1147) reference

8.4.3.26 **GAZEBO_VISIBLE** void gazebo::msgs::Set (msgs::Quaternion * *_q*, const math::Quaternion & *_v*)

Set a msgs::Quaternion from a **math::Quaternion** (p. 824).

Parameters

out	<i>_q</i>	A msgs::Quaternion pointer
in	<i>_v</i>	A math::Quaternion (p. 824) reference

8.4.3.27 **GAZEBO_VISIBLE** void gazebo::msgs::Set (msgs::Pose * *_p*, const math::Pose & *_v*)

Set a msgs::Pose from a **math::Pose** (p. 797).

Parameters

out	<i>_p</i>	A msgs::Pose pointer
in	<i>_v</i>	A math::Pose (p. 797) reference

8.4.3.28 **GAZEBO_VISIBLE** void gazebo::msgs::Set (msgs::Color * *_c*, const common::Color & *_v*)

Set a msgs::Color from a **common::Color** (p. 249).

Parameters

out	<code>_p</code>	A <code>msgs::Color</code> pointer
in	<code>_v</code>	A common::Color (p. 249) reference

8.4.3.29 **GAZEBO_VISIBLE** void gazebo::msgs::Set (msgs::Time * *_t*, const common::Time & *_v*)

Set a `msgs::Time` from a **common::Time** (p. 1099).

Parameters

out	<code>_p</code>	A <code>msgs::Time</code> pointer
in	<code>_v</code>	A common::Time (p. 1099) reference

8.4.3.30 void gazebo::msgs::Set (msgs::SphericalCoordinates * *_s*, const common::SphericalCoordinates & *_v*)

Set a `msgs::SphericalCoordinates` from a **common::SphericalCoordinates** (p. 1057) object.

Parameters

out	<code>_p</code>	A <code>msgs::SphericalCoordinates</code> pointer.
in	<code>_v</code>	A common::SphericalCoordinates (p. 1057) reference

8.4.3.31 **GAZEBO_VISIBLE** void gazebo::msgs::Set (msgs::PlaneGeom * *_p*, const math::Plane & *_v*)

Set a `msgs::Plane` from a **math::Plane** (p. 788).

Parameters

out	<code>_p</code>	A <code>msgs::Plane</code> pointer
in	<code>_v</code>	A math::Plane (p. 788) reference

8.4.3.32 **GAZEBO_VISIBLE** void gazebo::msgs::Stamp (msgs::Header * *_header*)

Time stamp a header.

Parameters

in	<code>_header</code>	Header to stamp
----	----------------------	-----------------

8.4.3.33 **GAZEBO_VISIBLE** void gazebo::msgs::Stamp (msgs::Time * *_time*)

Set the time in a time message.

Parameters

in	<code>_time</code>	A Time message
----	--------------------	----------------

8.4.3.34 GAZEBO_VISIBLE msgs::TrackVisual gazebo::msgs::TrackVisualFromSDF (sdf::ElementPtr *_sdf*)

Create a msgs::TrackVisual from a track visual SDF element.

Parameters

<i>in</i>	<i>_sdf</i>	The sdf element
-----------	-------------	-----------------

Returns

The new msgs::TrackVisual object

8.4.3.35 GAZEBO_VISIBLE msgs::Visual gazebo::msgs::VisualFromSDF (sdf::ElementPtr *_sdf*)

Create a msgs::Visual from a visual SDF element.

Parameters

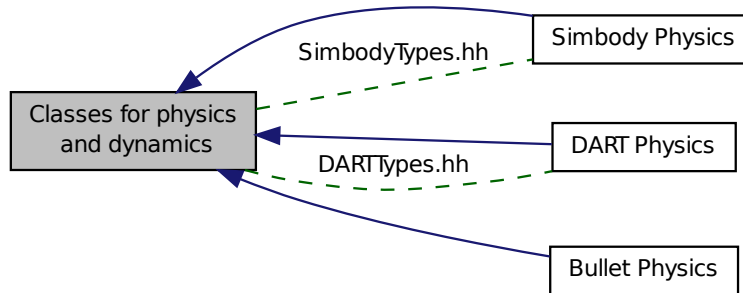
<i>in</i>	<i>_sdf</i>	The sdf element
-----------	-------------	-----------------

Returns

The new msgs::Visual object

8.5 Classes for physics and dynamics

Collaboration diagram for Classes for physics and dynamics:



Modules

- **DART Physics**
dart physics engine wrapper
- **Bullet Physics**
- **Simbody Physics**
simbody physics engine wrapper

Files

- file **DARTTypes.hh**
DART wrapper forward declarations and typedefs.
- file **PhysicsTypes.hh**
default namespace for gazebo
- file **SimbodyTypes.hh**
Simbody wrapper forward declarations and typedefs.

Namespaces

- namespace **gazebo::physics**
namespace for physics

Classes

- class **gazebo::physics::Actor**
Actor (p. 139) class enables GPU based mesh model / skeleton scriptable animation.
- class **gazebo::physics::BallJoint< T >**

- Base* (p. 168) class for a ball joint.
- class **gazebo::physics::Base**
 - Base* (p. 168) class for most physics classes.
- class **gazebo::physics::BoxShape**
 - Box geometry primitive.
- class **gazebo::physics::Collision**
 - Base* (p. 168) class for all collision entities.
- class **gazebo::physics::CollisionState**
 - Store state information of a *physics::Collision* (p. 235) object.
- class **gazebo::physics::Contact**
 - A contact between two collisions.
- class **gazebo::physics::ContactManager**
 - Aggregates all the contact information generated by the collision detection engine.
- class **gazebo::physics::CylinderShape**
 - Cylinder collision.
- class **gazebo::physics::Entity**
 - Base* (p. 168) class for all physics objects in Gazebo.
- class **gazebo::physics::FrictionPyramid**
 - Parameters used for friction pyramid model.
- class **gazebo::physics::GearboxJoint< T >**
 - A double axis gearbox joint.
- class **gazebo::physics::Gripper**
 - A gripper abstraction.
- class **gazebo::physics::HeightmapShape**
 - HeightmapShape* (p. 506) collision shape builds a heightmap from an image.
- class **gazebo::physics::Hinge2Joint< T >**
 - A two axis hinge joint.
- class **gazebo::physics::HingeJoint< T >**
 - A single axis hinge joint.
- class **gazebo::physics::Inertial**
 - A class for inertial information about a link.
- class **gazebo::physics::Joint**
 - Base* (p. 168) class for all joints.
- class **gazebo::physics::JointController**
 - A class for manipulating *physics::Joint* (p. 541).
- class **gazebo::physics::JointState**
 - keeps track of state of a *physics::Joint* (p. 541)
- class **gazebo::physics::JointWrench**
 - Wrench information from a joint.
- class **gazebo::physics::Link**
 - Link* (p. 595) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.
- class **gazebo::physics::LinkState**
 - Store state information of a *physics::Link* (p. 595) object.
- class **Logplay**
 - Open and playback log files that were recorded using *LogRecord*.
- class **gazebo::util::LogPlay**

- class **gazebo::physics::MeshShape**
Triangle mesh collision shape.
- class **gazebo::physics::Model**
A model is a collection of links, joints, and plugins.
- class **gazebo::physics::ModelState**
*Store state information of a **physics::Model** (p. 678) object.*
- class **gazebo::physics::MultiRayShape**
Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.
- class **gazebo::physics::PhysicsEngine**
***Base** (p. 168) class for a physics engine.*
- class **gazebo::physics::PhysicsFactory**
The physics factory instantiates different physics engines.
- class **gazebo::physics::PlaneShape**
***Collision** (p. 235) for an infinite plane.*
- class **gazebo::physics::RayShape**
***Base** (p. 168) class for Ray collision geometry.*
- class **gazebo::physics::Road**
*for building a **Road** (p. 869) from SDF*
- class **gazebo::physics::ScrewJoint**< T >
A screw joint, which has both prismatic and rotational DOFs.
- class **gazebo::physics::Shape**
***Base** (p. 168) class for all shapes.*
- class **gazebo::physics::SimbodyModel**
A model is a collection of links, joints, and plugins.
- class **gazebo::physics::SliderJoint**< T >
A slider joint.
- class **gazebo::physics::SphereShape**
Sphere collision shape.
- class **gazebo::physics::State**
***State** (p. 1068) of an entity.*
- class **gazebo::physics::SurfaceParams**
***SurfaceParams** (p. 1090) defines various Surface contact parameters.*
- class **gazebo::physics::UniversalJoint**< T >
A universal joint.
- class **gazebo::physics::World**
The world provides access to all other object within a simulated environment.
- class **gazebo::physics::WorldState**
*Store state information of a **physics::World** (p. 1239) object.*

Macros

- #define **GZ_REGISTER_PHYSICS_ENGINE**(name, classname)
Static physics registration macro.

Typedefs

- typedef PhysicsEnginePtr(* **gazebo::physics::PhysicsFactoryFn**)(WorldPtr world)

Functions

- **GAZEBO_VISIBLE** WorldPtr **gazebo::physics::create_world** (const std::string &_name="")
Create a world given a name.
- **GAZEBO_VISIBLE** bool **gazebo::physics::fini** ()
*Finalize transport by calling **gazebo::transport::fini** (p. 94).*
- **GAZEBO_VISIBLE** WorldPtr **gazebo::physics::get_world** (const std::string &_name="")
Returns a pointer to a world by name.
- **GAZEBO_VISIBLE** uint32_t **gazebo::physics::getUniqueId** ()
Get a unique ID.
- **GAZEBO_VISIBLE** void **gazebo::physics::init_world** (WorldPtr _world)
Init world given a pointer to it.
- **GAZEBO_VISIBLE** void **gazebo::physics::init_worlds** ()
*initialize multiple worlds stored in static variable **gazebo::g_worlds***
- **GAZEBO_VISIBLE** bool **gazebo::physics::load** ()
*Setup **gazebo::SystemPlugin** (p. 1098)'s and call **gazebo::transport::init** (p. 96).*
- **GAZEBO_VISIBLE** void **gazebo::physics::load_world** (WorldPtr _world, sdf::ElementPtr _sdf)
Load world from sdf::Element pointer.
- **GAZEBO_VISIBLE** void **gazebo::physics::load_worlds** (sdf::ElementPtr _sdf)
load multiple worlds from single sdf::Element pointer
- **GAZEBO_VISIBLE** void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)
*Pause world by calling **World::SetPaused** (p. 1250).*
- **GAZEBO_VISIBLE** void **gazebo::physics::pause_worlds** (bool pause)
*pause multiple worlds stored in static variable **gazebo::g_worlds***
- **GAZEBO_VISIBLE** void **gazebo::physics::remove_worlds** ()
*remove multiple worlds stored in static variable **gazebo::g_worlds***
- **GAZEBO_VISIBLE** void **gazebo::physics::run_world** (WorldPtr _world, unsigned int _iterations=0)
*Run world by calling **World::Run()** (p. 1249) given a pointer to it.*
- **GAZEBO_VISIBLE** void **gazebo::physics::run_worlds** (unsigned int _iterations=0)
*Run multiple worlds stored in static variable **gazebo::g_worlds**.*
- **GAZEBO_VISIBLE** void **gazebo::physics::stop_world** (WorldPtr _world)
*Stop world by calling **World::Stop()** (p. 1251) given a pointer to it.*
- **GAZEBO_VISIBLE** void **gazebo::physics::stop_worlds** ()
*stop multiple worlds stored in static variable **gazebo::g_worlds***
- **GAZEBO_VISIBLE** bool **gazebo::physics::worlds_running** ()
Return true if any world is running.

Variables

- static std::string **gazebo::physics::EntityTypename** []
String names for the different entity types.

8.5.1 Detailed Description

8.5.2 Macro Definition Documentation

8.5.2.1 #define GZ_REGISTER_PHYSICS_ENGINE(*name*, *classname*)

Value:

```
GAZEBO_VISIBLE PhysicsEnginePtr New##classname(WorldPtr _world) \
{ \
    return PhysicsEnginePtr(new gazebo::physics::classname(_world)); \
} \
GAZEBO_VISIBLE \
void Register##classname() \
{ \
    PhysicsFactory::RegisterPhysicsEngine(name, New##classname);\
}
```

Static physics registration macro.

Use this macro to register physics engine with the server.

Parameters

<i>in</i>	<i>name</i>	Physics type name, as it appears in the world file.
<i>in</i>	<i>classname</i>	C++ class name for the physics engine.

8.5.3 Typedef Documentation

8.5.3.1 typedef PhysicsEnginePtr(* gazebo::physics::PhysicsFactoryFn)(WorldPtr world)

8.5.4 Function Documentation

8.5.4.1 GAZEBO_VISIBLE WorldPtr gazebo::physics::create_world (const std::string & *_name* = " ")

Create a world given a name.

Parameters

<i>in</i>	<i>_name</i>	Name of the world to create.
-----------	--------------	------------------------------

Returns

Pointer to the new world.

8.5.4.2 GAZEBO_VISIBLE bool gazebo::physics::fini ()

Finalize transport by calling **gazebo::transport::fini** (p. 94).

8.5.4.3 GAZEBO_VISIBLE WorldPtr gazebo::physics::get_world (const std::string & *_name* = " ")

Returns a pointer to a world by name.

Parameters

in	<i>_name</i>	Name of the world to get.
----	--------------	---------------------------

Returns

Pointer to the world.

8.5.4.4 GAZEBO_VISIBLE uint32_t gazebo::physics::getUniqueld ()

Get a unique ID.

Returns

A unique integer

8.5.4.5 GAZEBO_VISIBLE void gazebo::physics::init_world (WorldPtr *_world*)

Init world given a pointer to it.

Parameters

in	<i>_world</i>	World (p. 1239) to initialize.
----	---------------	---------------------------------------

8.5.4.6 GAZEBO_VISIBLE void gazebo::physics::init_worlds ()

initialize multiple worlds stored in static variable gazebo::g_worlds

8.5.4.7 GAZEBO_VISIBLE bool gazebo::physics::load ()

Setup **gazebo::SystemPlugin** (p. 1098)'s and call **gazebo::transport::init** (p. 96).

8.5.4.8 GAZEBO_VISIBLE void gazebo::physics::load_world (WorldPtr *_world*, sdf::ElementPtr *_sdf*)

Load world from sdf::Element pointer.

Parameters

in	<i>_world</i>	Pointer to a world.
in	<i>_sdf</i>	SDF values to load from.

8.5.4.9 GAZEBO_VISIBLE void gazebo::physics::load_worlds (sdf::ElementPtr *_sdf*)

load multiple worlds from single sdf::Element pointer

Parameters

in	<i>_sdf</i>	SDF values used to create worlds.
----	-------------	-----------------------------------

8.5.4.10 GAZEBO_VISIBLE void gazebo::physics::pause_world (WorldPtr *_world*, bool *_pause*)

Pause world by calling **World::SetPaused** (p. 1250).

Parameters

in	<i>_world</i>	World (p. 1239) to pause or unpause.
in	<i>_pause</i>	True to pause, False to unpause.

8.5.4.11 GAZEBO_VISIBLE void gazebo::physics::pause_worlds (bool *pause*)

pause multiple worlds stored in static variable gazebo::g_worlds

Parameters

in	<i>_pause</i>	True to pause, False to unpause.
----	---------------	----------------------------------

8.5.4.12 GAZEBO_VISIBLE void gazebo::physics::remove_worlds ()

remove multiple worlds stored in static variable gazebo::g_worlds

8.5.4.13 GAZEBO_VISIBLE void gazebo::physics::run_world (WorldPtr *_world*, unsigned int *_iterations* = 0)

Run world by calling **World::Run()** (p. 1249) given a pointer to it.

Parameters

in	<i>_world</i>	World (p. 1239) to run.
in	<i>_iterations</i>	Number of iterations for each world to take. Zero indicates that each world should continue forever.

8.5.4.14 GAZEBO_VISIBLE void gazebo::physics::run_worlds (unsigned int *_iterations* = 0)

Run multiple worlds stored in static variable gazebo::g_worlds.

Parameters

in	<i>_iterations</i>	Number of iterations for each world to take. Zero indicates that each world should continue forever.
----	--------------------	--

8.5.4.15 GAZEBO_VISIBLE void gazebo::physics::stop_world (WorldPtr *_world*)

Stop world by calling **World::Stop()** (p. 1251) given a pointer to it.

Parameters

in	<i>_world</i>	World (p. 1239) to stop.
----	---------------	---------------------------------

8.5.4.16 GAZEBO_VISIBLE void gazebo::physics::stop_worlds ()

stop multiple worlds stored in static variable gazebo::g_worlds

8.5.4.17 GAZEBO_VISIBLE bool gazebo::physics::worlds_running ()

Return true if any world is running.

Returns

True if any world is running.

8.5.5 Variable Documentation**8.5.5.1** std::string gazebo::physics::EntityTypename[] [static]**Initial value:**

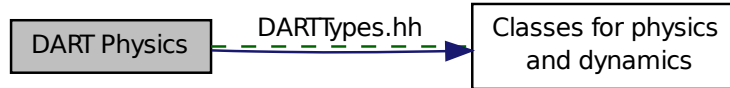
```
{
    "common",
    "entity",
    "model",
    "actor",
    "link",
    "collision",
    "light",
    "visual",
    "joint",
    "ball",
    "hinge2",
    "hinge",
    "slider",
    "universal",
    "shape",
    "box",
    "cylinder",
    "heightmap",
    "map",
    "multiray",
    "ray",
    "plane",
    "sphere",
    "trimesh"
}
```

String names for the different entity types.

8.6 DART Physics

dart physics engine wrapper

Collaboration diagram for DART Physics:



Files

- file **DARTTypes.hh**
DART wrapper forward declarations and typedefs.

Classes

- class **gazebo::physics::DARTLink**
DART Link (p. 595) class.
- class **gazebo::physics::DARTModel**
DART model class.
- class **gazebo::physics::DARTPhysics**
DART physics engine.
- class **gazebo::physics::DARTRayShape**
Ray collision.
- class **gazebo::physics::DARTTypes**
A set of functions for converting between the math types used by gazebo and dart.

Functions

- **gazebo::physics::DARTRayShape::DARTRayShape** (PhysicsEnginePtr _physicsEngine)
Constructor for a global ray.
- **gazebo::physics::DARTRayShape::DARTRayShape** (CollisionPtr _collision)
Constructor.
- virtual **gazebo::physics::DARTRayShape::~~DARTRayShape** ()
Destructor.
- static Eigen::Isometry3d **gazebo::physics::DARTTypes::ConvPose** (const math::Pose &_pose)
- static math::Pose **gazebo::physics::DARTTypes::ConvPose** (const Eigen::Isometry3d &_T)
- static Eigen::Quaterniond **gazebo::physics::DARTTypes::ConvQuat** (const math::Quaternion &_quat)
- static math::Quaternion **gazebo::physics::DARTTypes::ConvQuat** (const Eigen::Quaterniond &_quat)
- static Eigen::Vector3d **gazebo::physics::DARTTypes::ConvVec3** (const math::Vector3 &_vec3)
- static math::Vector3 **gazebo::physics::DARTTypes::ConvVec3** (const Eigen::Vector3d &_vec3)

- virtual void **gazebo::physics::DARTRayShape::GetIntersection** (double &_dist, std::string &_entity)
Get the nearest intersection.
- virtual void **gazebo::physics::DARTRayShape::SetPoints** (const math::Vector3 &_posStart, const math::Vector3 &_posEnd)
Set the ray based on starting and ending points relative to the body.
- virtual void **gazebo::physics::DARTRayShape::Update** ()
Update the ray collision.

8.6.1 Detailed Description

dart physics engine wrapper

8.6.2 Function Documentation

8.6.2.1 gazebo::physics::DARTRayShape::DARTRayShape (PhysicsEnginePtr _physicsEngine) [explicit]

Constructor for a global ray.

Parameters

in	<i>_physicsEngine</i>	Pointer to the physics engine.
----	-----------------------	--------------------------------

8.6.2.2 gazebo::physics::DARTRayShape::DARTRayShape (CollisionPtr _collision) [explicit]

Constructor.

Parameters

in	<i>_collision</i>	Collision (p. 235) object this ray is attached to.
----	-------------------	---

8.6.2.3 virtual gazebo::physics::DARTRayShape::~~DARTRayShape () [virtual]

Destructor.

8.6.2.4 static Eigen::Isometry3d gazebo::physics::DARTTypes::ConvPose (const math::Pose & _pose) [inline], [static]

References gazebo::math::Pose::pos, and gazebo::math::Pose::rot.

8.6.2.5 static math::Pose gazebo::physics::DARTTypes::ConvPose (const Eigen::Isometry3d & .T) [inline], [static]

References gazebo::math::Pose::pos, and gazebo::math::Pose::rot.

8.6.2.6 `static Eigen::Quaterniond gazebo::physics::DARTTypes::ConvQuat (const math::Quaternion & _quat) [inline], [static]`

References `gazebo::math::Quaternion::w`, `gazebo::math::Quaternion::x`, `gazebo::math::Quaternion::y`, and `gazebo::math::Quaternion::z`.

8.6.2.7 `static math::Quaternion gazebo::physics::DARTTypes::ConvQuat (const Eigen::Quaterniond & _quat) [inline], [static]`

8.6.2.8 `static Eigen::Vector3d gazebo::physics::DARTTypes::ConvVec3 (const math::Vector3 & _vec3) [inline], [static]`

References `gazebo::math::Vector3::x`, `gazebo::math::Vector3::y`, and `gazebo::math::Vector3::z`.

Referenced by `gazebo::physics::DARTBoxShape::SetSize()`.

8.6.2.9 `static math::Vector3 gazebo::physics::DARTTypes::ConvVec3 (const Eigen::Vector3d & _vec3) [inline], [static]`

8.6.2.10 `virtual void gazebo::physics::DARTRayShape::GetIntersection (double & _dist, std::string & _entity) [virtual]`

Get the nearest intersection.

Parameters

out	<code>_dist</code>	Distance to the intersection.
out	<code>_entity</code>	Name of the entity that was hit.

Implements `gazebo::physics::RayShape` (p. 852).

8.6.2.11 `virtual void gazebo::physics::DARTRayShape::SetPoints (const math::Vector3 & _posStart, const math::Vector3 & _posEnd) [virtual]`

Set the ray based on starting and ending points relative to the body.

Parameters

in	<code>_posStart</code>	Start position, relative the body
in	<code>_posEnd</code>	End position, relative to the body

Reimplemented from `gazebo::physics::RayShape` (p. 853).

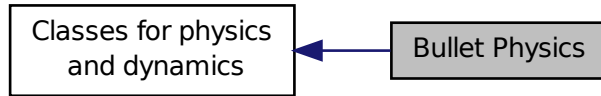
8.6.2.12 `virtual void gazebo::physics::DARTRayShape::Update () [virtual]`

Update the ray collision.

Implements `gazebo::physics::RayShape` (p. 854).

8.7 Bullet Physics

Collaboration diagram for Bullet Physics:



Classes

- class `gazebo::physics::DARTMultiRayShape`
DART specific version of `MultiRayShape` (p. 723).

Functions

- `gazebo::physics::DARTMultiRayShape::DARTMultiRayShape` (`CollisionPtr _parent`)
Constructor.
- virtual `gazebo::physics::DARTMultiRayShape::~~DARTMultiRayShape` ()
Destructor.
- void `gazebo::physics::DARTMultiRayShape::AddRay` (`const math::Vector3 &_start`, `const math::Vector3 &_end`)
Add a ray to the collision.
- virtual void `gazebo::physics::DARTMultiRayShape::UpdateRays` ()
Physics engine specific method for updating the rays.

8.7.1 Detailed Description

8.7.2 Function Documentation

8.7.2.1 `gazebo::physics::DARTMultiRayShape::DARTMultiRayShape (CollisionPtr _parent) [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Parent Collision (p. 235).
----	----------------------	-----------------------------------

8.7.2.2 `virtual gazebo::physics::DARTMultiRayShape::~~DARTMultiRayShape () [virtual]`

Destructor.

8.7.2.3 `void gazebo::physics::DARTMultiRayShape::AddRay (const math::Vector3 & _start, const math::Vector3 & _end)`
[protected], [virtual]

Add a ray to the collision.

Parameters

in	<code>_start</code>	Start location of the ray.
in	<code>_end</code>	End location of the ray.

Reimplemented from **`gazebo::physics::MultiRayShape`** (p. 726).

8.7.2.4 `virtual void gazebo::physics::DARTMultiRayShape::UpdateRays ()` [virtual]

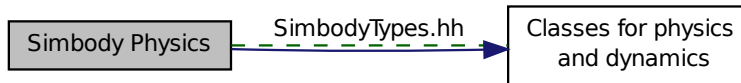
Physics engine specific method for updating the rays.

Implements **`gazebo::physics::MultiRayShape`** (p. 730).

8.8 Simbody Physics

simbody physics engine wrapper

Collaboration diagram for Simbody Physics:



Files

- file **SimbodyTypes.hh**
Simbody wrapper forward declarations and typedefs.

Classes

- class **gazebo::physics::SimbodyBallJoint**
***SimbodyBallJoint** (p. 936) class models a ball joint in Simbody.*
- class **gazebo::physics::SimbodyBoxShape**
Simbody box collision.
- class **gazebo::physics::SimbodyCollision**
Simbody collisions.
- class **gazebo::physics::SimbodyCylinderShape**
Cylinder collision.
- class **gazebo::physics::SimbodyHeightmapShape**
Height map collision.
- class **gazebo::physics::SimbodyHinge2Joint**
A two axis hinge joint.
- class **gazebo::physics::SimbodyHingeJoint**
A single axis hinge joint.
- class **gazebo::physics::SimbodyJoint**
***Base** (p. 168) class for all joints.*
- class **gazebo::physics::SimbodyLink**
*Simbody **Link** (p. 595) class.*
- class **gazebo::physics::SimbodyMeshShape**
Triangle mesh collision.
- class **gazebo::physics::SimbodyMultiRayShape**
*Simbody specific version of **MultiRayShape** (p. 723).*
- class **gazebo::physics::SimbodyPhysics**
Simbody physics engine.
- class **gazebo::physics::SimbodyPlaneShape**

Simbody collision for an infinite plane.

- class **gazebo::physics::SimbodyRayShape**

Ray shape for simbody.

- class **gazebo::physics::SimbodyScrewJoint**

A screw joint.

- class **gazebo::physics::SimbodySliderJoint**

A slider joint.

- class **gazebo::physics::SimbodySphereShape**

Simbody sphere collision.

- class **gazebo::physics::SimbodyUniversalJoint**

A simbody universal joint class.

8.8.1 Detailed Description

simbody physics engine wrapper

8.9 Rendering

A set of rendering related class, functions, and definitions.

Namespaces

- namespace **gazebo::rendering**
Rendering namespace.

Classes

- class **gazebo::rendering::ArrowVisual**
Basic arrow visualization.
- class **gazebo::rendering::AxisVisual**
Basic axis visualization.
- class **gazebo::rendering::Camera**
Basic camera sensor.
- class **gazebo::rendering::CameraVisual**
Basic camera visualization.
- class **gazebo::rendering::COMVisual**
Basic Center of Mass visualization.
- class **gazebo::rendering::ContactVisual**
Contact visualization.
- class **gazebo::rendering::Conversions**
Conversions (p. 296) Conversions.hh (p. 1306) rendering/Conversions.hh (p. 1306).
- class **gazebo::rendering::DepthCamera**
Depth camera used to render depth data into an image buffer.
- class **gazebo::rendering::DynamicLines**
Class for drawing lines that can change.
- class **gazebo::rendering::DynamicRenderable**
Abstract base class providing mechanisms for dynamically growing hardware buffers.
- class **gazebo::rendering::Events**
Base class for rendering events.
- class **gazebo::rendering::FPSViewController**
First Person Shooter style view controller.
- class **gazebo::rendering::GpuLaser**
GPU based laser distance sensor.
- class **gazebo::rendering::Grid**
Displays a grid of cells, drawn with lines.
- class **gazebo::rendering::GUIOverlay**
A class that creates a CEGUI overlay on a render window.
- class **gazebo::rendering::Heightmap**
Rendering a terrain using heightmap information.
- class **gazebo::rendering::JointVisual**
Visualization for joints.
- class **gazebo::rendering::LaserVisual**

Visualization for laser data.

- class **gazebo::rendering::Light**
A light source.
- class **gazebo::rendering::MovableText**
Movable text.
- class **gazebo::rendering::OrbitViewController**
Orbit view controller.
- class **gazebo::rendering::Projector**
Projects a material onto surface, light a light projector.
- class **gazebo::rendering::RenderEngine**
Adaptor to Ogre3d.
- class **gazebo::rendering::RFIDTagVisual**
Visualization for RFID tags sensor.
- class **gazebo::rendering::RFIDVisual**
Visualization for RFID sensor.
- class **Road**
Used to render a strip of road.
- class **gazebo::rendering::Road2d**
- class **gazebo::rendering::RTShaderSystem**
*Implements **Ogre** (p. 137)'s Run-Time Shader system.*
- class **gazebo::rendering::Scene**
Representation of an entire scene graph.
- class **gazebo::rendering::SelectionObj**
Interactive selection object for models and links.
- class **gazebo::rendering::SonarVisual**
Visualization for sonar data.
- class **gazebo::rendering::TransmitterVisual**
Visualization for the wireless propagation data.
- class **gazebo::rendering::UserCamera**
A camera used for user visualization of a scene.
- class **gazebo::rendering::VideoVisual**
A visual element that displays a video as a texture.
- class **gazebo::rendering::ViewController**
Base class for view controllers.
- class **gazebo::rendering::Visual**
A renderable object.
- class **gazebo::rendering::WindowManager**
Class to manage render windows.
- class **gazebo::rendering::WireBox**
Draws a wireframe box.
- class **gazebo::rendering::WrenchVisual**
Visualization for sonar data.

Functions

- **GAZEBO_VISIBLE** rendering::ScenePtr **gazebo::rendering::create_scene** (const std::string &_name, bool _enableVisualizations, bool _isServer=false)
create **rendering::Scene** (p. 879) by name.
- **GAZEBO_VISIBLE** bool **gazebo::rendering::fini** ()
teardown rendering engine.
- **GAZEBO_VISIBLE** rendering::ScenePtr **gazebo::rendering::get_scene** (const std::string &_name="")
get pointer to **rendering::Scene** (p. 879) by name.
- **GAZEBO_VISIBLE** bool **gazebo::rendering::init** ()
init rendering engine.
- **GAZEBO_VISIBLE** bool **gazebo::rendering::load** ()
load rendering engine.
- **GAZEBO_VISIBLE** void **gazebo::rendering::remove_scene** (const std::string &_name)
remove a **rendering::Scene** (p. 879) by name

8.9.1 Detailed Description

A set of rendering related class, functions, and definitions.

8.9.2 Function Documentation

8.9.2.1 **GAZEBO_VISIBLE** rendering::ScenePtr **gazebo::rendering::create_scene** (const std::string & *_name*, bool *_enableVisualizations*, bool *_isServer = false*)

create **rendering::Scene** (p. 879) by name.

Parameters

in	<i>_name</i>	Name of the scene to create.
in	<i>_enable-Visualizations</i>	True enables visualization elements such as laser lines.

8.9.2.2 **GAZEBO_VISIBLE** bool **gazebo::rendering::fini** ()

teardown rendering engine.

8.9.2.3 **GAZEBO_VISIBLE** rendering::ScenePtr **gazebo::rendering::get_scene** (const std::string & *_name* = " ")

get pointer to **rendering::Scene** (p. 879) by name.

Parameters

in	<i>_name</i>	Name of the scene to retrieve.
----	--------------	--------------------------------

8.9.2.4 GAZEBO_VISIBLE bool gazebo::rendering::init ()

init rendering engine.

8.9.2.5 GAZEBO_VISIBLE bool gazebo::rendering::load ()

load rendering engine.

8.9.2.6 GAZEBO_VISIBLE void gazebo::rendering::remove_scene (const std::string & *_name*)

remove a **rendering::Scene** (p. 879) by name

Parameters

<i>in</i>	<i>_name</i>	The name of the scene to remove.
-----------	--------------	----------------------------------

8.10 Sensors

A set of sensor classes, functions, and definitions.

Files

- file **SensorTypes.hh**
Forward declarations and typedefs for sensors.

Namespaces

- namespace **gazebo::sensors**
Sensors namespace.

Classes

- class **gazebo::sensors::CameraSensor**
Basic camera sensor.
- class **gazebo::sensors::ContactSensor**
Contact sensor.
- class **gazebo::sensors::DepthCameraSensor**
- class **gazebo::sensors::ForceTorqueSensor**
Sensor (p. 907) for measure force and torque on a joint.
- class **gazebo::sensors::GpsSensor**
GpsSensor (p. 464) to provide position measurement.
- class **gazebo::sensors::GpuRaySensor**
- class **gazebo::sensors::ImuSensor**
An IMU sensor.
- class **gazebo::sensors::MultiCameraSensor**
Multiple camera sensor.
- class **gazebo::sensors::Noise**
Noise (p. 748) models for sensor output signals.
- class **gazebo::sensors::NoiseFactory**
Use this noise manager for creating and loading noise models.
- class **gazebo::sensors::RaySensor**
Sensor (p. 907) with one or more rays.
- class **gazebo::sensors::RFIDSensor**
Sensor (p. 907) class for RFID type of sensor.
- class **gazebo::sensors::RFIDTag**
RFIDTag (p. 861) to interact with RFIDTagSensors.
- class **gazebo::sensors::Sensor**
Base class for sensors.
- class **SensorFactor**
The sensor factory; the class is just for namespacing purposes.
- class **gazebo::sensors::SensorFactory**
- class **gazebo::sensors::SensorManager**

Class to manage and update all sensors.

- class **gazebo::sensors::SonarSensor**
Sensor (p. 907) with sonar cone.
- class **gazebo::sensors::WirelessReceiver**
Sensor (p. 907) class for receiving wireless signals.
- class **gazebo::sensors::WirelessTransceiver**
Sensor (p. 907) class for receiving wireless signals.
- class **gazebo::sensors::WirelessTransmitter**
Transmitter to send wireless signals.

Macros

- #define **GZ_REGISTER_STATIC_SENSOR**(name, classname)
Static sensor registration macro.

Functions

- **GAZEBO_VISIBLE** std::string **gazebo::sensors::create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName) **GAZEBO_DEPRECATED**(2.0)
Deprecated.
- **GAZEBO_VISIBLE** std::string **gazebo::sensors::create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName, uint32_t _parentId)
Create a sensor using SDF.
- **GAZEBO_VISIBLE** void **gazebo::sensors::disable** ()
Disable sensors.
- **GAZEBO_VISIBLE** void **gazebo::sensors::enable** ()
Enable sensors.
- **GAZEBO_VISIBLE** bool **gazebo::sensors::fini** ()
shutdown the sensor generation loop.
- **GAZEBO_VISIBLE** SensorPtr **gazebo::sensors::get_sensor** (const std::string &_name)
Get a sensor using by name.
- **GAZEBO_VISIBLE** bool **gazebo::sensors::init** ()
initialize the sensor generation loop.
- **GAZEBO_VISIBLE** bool **gazebo::sensors::load** ()
Load the sensor library.
- **GAZEBO_VISIBLE** void **gazebo::sensors::remove_sensor** (const std::string &_sensorName)
Remove a sensor by name.
- **GAZEBO_VISIBLE** bool **gazebo::sensors::remove_sensors** ()
Remove all sensors.
- **GAZEBO_VISIBLE** void **gazebo::sensors::run_once** (bool _force=false)
Run the sensor generation one step.
- **GAZEBO_VISIBLE** void **gazebo::sensors::run_threads** ()
Run sensors in a threads. This is a non-blocking call.
- **GAZEBO_VISIBLE** void **gazebo::sensors::stop** ()
Stop the sensor generation loop.

8.10.1 Detailed Description

A set of sensor classes, functions, and definitions. GPU based laser sensor.

Depth camera sensor This sensor is used for simulating standard monocular cameras

This sensor cast rays into the world, tests for intersections, and reports the range to the nearest object. It is used by ranging sensor models (e.g., sonars and scanning laser range finders).

8.10.2 Macro Definition Documentation

8.10.2.1 #define GZ_REGISTER_STATIC_SENSOR(*name*, *classname*)

Value:

```
GAZEBO_VISIBLE Sensor *New##classname() \
{ \
    return new gazebo::sensors::classname(); \
} \
GAZEBO_VISIBLE \
void Register##classname() \
{ \
    SensorFactory::RegisterSensor(name, New##classname);\
}
```

Static sensor registration macro.

Use this macro to register sensors with the server.

Parameters

<i>name</i>	Sensor type name, as it appears in the world file.
<i>classname</i>	C++ class name for the sensor.

8.10.3 Function Documentation

8.10.3.1 GAZEBO_VISIBLE std::string gazebo::sensors::create_sensor (sdf::ElementPtr *_elem*, const std::string & *_worldName*, const std::string & *_parentName*)

Deprecated.

8.10.3.2 GAZEBO_VISIBLE std::string gazebo::sensors::create_sensor (sdf::ElementPtr *_elem*, const std::string & *_worldName*, const std::string & *_parentName*, uint32_t *_parentId*)

Create a sensor using SDF.

Parameters

in	<i>_elem</i>	The SDF element that describes the sensor.
in	<i>_worldName</i>	Name of the world in which to create the sensor.
in	<i>_parentName</i>	The fully scoped parent name (model::link).

Returns

The name of the new sensor.

8.10.3.3 GAZEBO_VISIBLE void gazebo::sensors::disable ()

Disable sensors.

8.10.3.4 GAZEBO_VISIBLE void gazebo::sensors::enable ()

Enable sensors.

8.10.3.5 GAZEBO_VISIBLE bool gazebo::sensors::fini ()

shutdown the sensor generation loop.

Returns

True if successfully finalized, false if not

8.10.3.6 GAZEBO_VISIBLE SensorPtr gazebo::sensors::get_sensor (const std::string & *_name*)

Get a sensor using by name.

The given name should have: world_name::model_name::link_name::sensor_name

Parameters

<i>in</i>	<i>_name</i>	Name of the sensor. This name should be fully scoped. This means <i>_name</i> = world_name::model_name::link_name::sensor_name. You may use the unscoped sensor name if that name is unique within the entire simulation. If the name is not unique a NULL pointer is returned.
-----------	--------------	---

Returns

Pointer to the sensor, NULL if the sensor could not be found.

8.10.3.7 GAZEBO_VISIBLE bool gazebo::sensors::init ()

initialize the sensor generation loop.

Returns

True if successfully initialized, false if not

8.10.3.8 GAZEBO_VISIBLE bool gazebo::sensors::load ()

Load the sensor library.

Returns

True if successfully loaded, false if not.

8.10.3.9 **GAZEBO_VISIBLE** void gazebo::sensors::remove_sensor (const std::string & *_sensorName*)

Remove a sensor by name.

Parameters

<i>in</i>	<i>_sensorName</i>	Name of sensor to remove
-----------	--------------------	--------------------------

8.10.3.10 **GAZEBO_VISIBLE** bool gazebo::sensors::remove_sensors ()

Remove all sensors.

Returns

True if all successfully removed, false if not

8.10.3.11 **GAZEBO_VISIBLE** void gazebo::sensors::run_once (bool *_force* = false)

Run the sensor generation one step.

Parameters

<i>_force,:</i>	If true, all sensors are forced to update. Otherwise a sensor will update based on it's Hz rate.
-----------------	--

8.10.3.12 **GAZEBO_VISIBLE** void gazebo::sensors::run_threads ()

Run sensors in a threads. This is a non-blocking call.

8.10.3.13 **GAZEBO_VISIBLE** void gazebo::sensors::stop ()

Stop the sensor generation loop.

8.11 Transport

Handles transportation of messages.

Files

- file **TransportTypes.hh**
Forward declarations for transport.

Classes

- class **gazebo::transport::CallbackHelper**
A helper class to handle callbacks when messages arrive.
- class **gazebo::transport::CallbackHelperT< M >**
Callback helper Template.
- class **gazebo::transport::Connection**
Single TCP/IP connection manager.
- class **gazebo::transport::ConnectionManager**
Manager of connections.
- class **gazebo::transport::IOManager**
Manages boost::asio IO.
- class **gazebo::transport::Node**
A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.
- class **gazebo::transport::Publication**
A publication for a topic.
- class **gazebo::transport::PublicationTransport**
transport/transport.hh
- class **gazebo::transport::Publisher**
A publisher of messages on a topic.
- class **gazebo::transport::RawCallbackHelper**
Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.
- class **gazebo::transport::SubscribeOptions**
Options for a subscription.
- class **gazebo::transport::Subscriber**
A subscriber to a topic.
- class **gazebo::transport::SubscriptionTransport**
transport/transport.hh
- class **gazebo::transport::TopicManager**
Manages topics and their subscriptions.

Typedefs

- typedef boost::shared_ptr
< CallbackHelper > **gazebo::transport::CallbackHelperPtr**
*boost shared pointer to **transport::CallbackHelper** (p. 192)*

Functions

- **GAZEBO_VISIBLE** void **gazebo::transport::clear_buffers** ()
Clear any remaining communication buffers.
- **GAZEBO_VISIBLE**
transport::ConnectionPtr **gazebo::transport::connectToMaster** ()
Create a connection to master.
- **GAZEBO_VISIBLE** void **gazebo::transport::fini** ()
Cleanup the transport component.
- **GAZEBO_VISIBLE** bool **gazebo::transport::get_master_uri** (std::string &_master_host, unsigned int &_master_port)
Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.
- **GAZEBO_VISIBLE** void **gazebo::transport::get_topic_namespaces** (std::list< std::string > &_namespaces)
Return all the namespace (world names) on the master.
- **GAZEBO_VISIBLE** std::map
< std::string, std::list
< std::string > > **gazebo::transport::getAdvertisedTopics** ()
Get a list of all the topics and their message types.
- **GAZEBO_VISIBLE** std::list
< std::string > **gazebo::transport::getAdvertisedTopics** (const std::string &_msgType)
Get a list of all the unique advertised topic names.
- **GAZEBO_VISIBLE** bool **gazebo::transport::getMinimalComms** ()
Get whether minimal comms has been enabled.
- **GAZEBO_VISIBLE** std::string **gazebo::transport::getTopicMsgType** (const std::string &_topicName)
Get the message typename that is published on the given topic.
- **GAZEBO_VISIBLE** bool **gazebo::transport::init** (const std::string &_masterHost="", unsigned int _masterPort=0, uint32_t _timeoutIterations=30)
Initialize the transport system.
- bool **gazebo::transport::is_stopped** ()
Is the transport system stopped?
- **GAZEBO_VISIBLE** void **gazebo::transport::pause_incoming** (bool _pause)
Pause or unpaue incoming messages.
- template<typename M >
GAZEBO_VISIBLE void **gazebo::transport::publish** (const std::string &_topic, const google::protobuf::Message &_message)
A convenience function for a one-time publication of a message.
- **GAZEBO_VISIBLE**
boost::shared_ptr
< msgs::Response > **gazebo::transport::request** (const std::string &_worldName, const std::string &_request, const std::string &_data="")
Send a request and receive a response.
- **GAZEBO_VISIBLE** void **gazebo::transport::requestNoReply** (const std::string &_worldName, const std::string &_request, const std::string &_data="")
Send a request and don't wait for a response.
- **GAZEBO_VISIBLE** void **gazebo::transport::requestNoReply** (NodePtr _node, const std::string &_request, const std::string &_data="")
Send a request and don't wait for a response.
- **GAZEBO_VISIBLE** void **gazebo::transport::run** ()
Run the transport component.

- **GAZEBO_VISIBLE** void `gazebo::transport::setMinimalComms` (bool `_enabled`)
Set whether minimal comms should be used.
- **GAZEBO_VISIBLE** void `gazebo::transport::stop` ()
Stop the transport component from running.
- **GAZEBO_VISIBLE** bool `gazebo::transport::waitForNamespaces` (const `gazebo::common::Time` & `_maxWait`)
*Blocks while waiting for topic namespaces from the **Master** (p. 637).*

8.11.1 Detailed Description

Handles transportation of messages.

Remarks

Environment Variables:

- `GAZEBO_IP_WHITE_LIST`: Comma separated list of valid IPs. Leave this empty to accept connections from all addresses.
- `GAZEBO_IP`: IP address to export. This will override the default IP lookup.
- `GAZEBO_HOSTNAME`: Hostname to export. Setting this will override both `GAZEBO_IP` and the default IP lookup.

8.11.2 Typedef Documentation

8.11.2.1 `typedef boost::shared_ptr<CallbackHelper> gazebo::transport::CallbackHelperPtr`

boost shared pointer to `transport::CallbackHelper` (p. 192)

8.11.3 Function Documentation

8.11.3.1 **GAZEBO_VISIBLE** void `gazebo::transport::clear_buffers` ()

Clear any remaining communication buffers.

8.11.3.2 **GAZEBO_VISIBLE** `transport::ConnectionPtr` `gazebo::transport::connectToMaster` ()

Create a connection to master.

Returns

Connection (p. 264) to the master, NULL on error.

8.11.3.3 **GAZEBO_VISIBLE** void `gazebo::transport::fini` ()

Cleanup the transport component.

8.11.3.4 **GAZEBO_VISIBLE** bool gazebo::transport::get_master_uri (std::string & *_master_host*, unsigned int & *_master_port*)

Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.

Parameters

out	<i>_master_host</i>	The hostname of the master is set to this param
out	<i>_master_port</i>	The port of the master is set to this param

Returns

true if GAZEBO_MASTER_URI was successfully parsed; false otherwise (in which case output params are not set)

8.11.3.5 **GAZEBO_VISIBLE** void gazebo::transport::get_topic_namespaces (std::list< std::string > & *_namespaces*)

Return all the namespace (world names) on the master.

Parameters

out	<i>_namespaces</i>	The list of namespace will be written here
-----	--------------------	--

8.11.3.6 **GAZEBO_VISIBLE** std::map<std::string, std::list<std::string> > gazebo::transport::getAdvertisedTopics ()

Get a list of all the topics and their message types.

Returns

A map where keys are message types, and values are a list of topic names.

8.11.3.7 **GAZEBO_VISIBLE** std::list<std::string> gazebo::transport::getAdvertisedTopics (const std::string & *_msgType*)

Get a list of all the unique advertised topic names.

Parameters

in	<i>_msgType</i>	Type of message to filter the result on. If empty, then a list of all the topics is returned.
----	-----------------	---

Returns

A list of the advertised topics that publish messages of the type specified by *_msgType*.

8.11.3.8 **GAZEBO_VISIBLE** bool gazebo::transport::getMinimalComms ()

Get whether minimal comms has been enabled.

Returns

True if minimal comms is enabled.

8.11.3.9 GAZEBO_VISIBLE std::string gazebo::transport::getTopicMsgType (const std::string & *_topicName*)

Get the message typename that is published on the given topic.

Parameters

<i>in</i>	<i>_topicName</i>	Name of the topic to query.
-----------	-------------------	-----------------------------

Returns

The message type, or empty string if the topic is not valid.

8.11.3.10 GAZEBO_VISIBLE bool gazebo::transport::init (const std::string & *_masterHost* = " ", unsigned int *_masterPort* = 0, uint32_t *_timeoutIterations* = 30)

Initialize the transport system.

Parameters

<i>in</i>	<i>_masterHost</i>	The hostname or IP of the master. Leave empty to use pull address from the GAZEBO_MASTER_URI env var.
<i>in</i>	<i>_masterPort</i>	The port of the master. Leave empty to use pull address from the GAZEBO_MASTER_URI env var.
<i>in</i>	<i>_timeoutIterations</i>	Number of times to wait for a connection to master.

Returns

true if initialization succeeded; false otherwise

8.11.3.11 GAZEBO_VISIBLE bool gazebo::transport::is_stopped ()

Is the transport system stopped?

Returns

true if the transport system is stopped; false otherwise

8.11.3.12 GAZEBO_VISIBLE void gazebo::transport::pause_incoming (bool *_pause*)

Pause or unpaue incoming messages.

When paused, messages are queued for later delivery

Parameters

<i>in</i>	<i>_pause</i>	If true, pause; otherwise unpaue
-----------	---------------	----------------------------------

8.11.3.13 `template<typename M > GAZEBO_VISIBLE void gazebo::transport::publish (const std::string & _topic, const google::protobuf::Message & _message)`

A convenience function for a one-time publication of a message.

This is inefficient, compared to **Node::Advertise** (p. 733) followed by **Publisher::Publish** (p. 822). This function should only be used when sending a message very infrequently.

Parameters

<code>in</code>	<code><i>_topic</i></code>	The topic to advertise
<code>in</code>	<code><i>_message</i></code>	Message to be published

8.11.3.14 `GAZEBO_VISIBLE boost::shared_ptr<msgs::Response> gazebo::transport::request (const std::string & _worldName, const std::string & _request, const std::string & _data = " ")`

Send a request and receive a response.

This call will block until a response is received.

Parameters

<code>in</code>	<code><i>_worldName</i></code>	The name of the world to which the request should be sent
<code>in</code>	<code><i>_request</i></code>	The type request.
<code>in</code>	<code><i>_data</i></code>	Optional data string.

Returns

The response to the request. Can be empty.

8.11.3.15 `GAZEBO_VISIBLE void gazebo::transport::requestNoReply (const std::string & _worldName, const std::string & _request, const std::string & _data = " ")`

Send a request and don't wait for a response.

This is non-blocking.

Parameters

<code>in</code>	<code><i>_worldName</i></code>	The name of the world to which the request should be sent.
<code>in</code>	<code><i>_request</i></code>	The type request.
<code>in</code>	<code><i>_data</i></code>	Optional data string.

8.11.3.16 `GAZEBO_VISIBLE void gazebo::transport::requestNoReply (NodePtr _node, const std::string & _request, const std::string & _data = " ")`

Send a request and don't wait for a response.

This is non-blocking.

Parameters

in	<code>_node</code>	Pointer to a node that provides communication.
in	<code>_request</code>	The type request.
in	<code>_data</code>	Optional data string.

8.11.3.17 GAZEBO_VISIBLE void gazebo::transport::run ()

Run the transport component.

Creates a thread to handle message passing. This call will block until the master can be contacted or until a retry limit is reached

8.11.3.18 GAZEBO_VISIBLE void gazebo::transport::setMinimalComms (bool *_enabled*)

Set whether minimal comms should be used.

This will be used to reduce network traffic.

8.11.3.19 GAZEBO_VISIBLE void gazebo::transport::stop ()

Stop the transport component from running.

8.11.3.20 GAZEBO_VISIBLE bool gazebo::transport::waitForNamespaces (const gazebo::common::Time & *_maxWait*)

Blocks while waiting for topic namespaces from the **Master** (p. 637).

This function will wait a maximum of `_maxWait`.

Returns

True if namespaces were found before `_maxWait` time.

8.12 Utility

Files

- file **UtilTypes.hh**

Classes

- class **gazebo::util::DiagnosticManager**
A diagnostic manager class.
- class **gazebo::util::DiagnosticTimer**
A timer designed for diagnostics.
- class **gazebo::util::OpenAL**
3D audio setup and playback.
- class **gazebo::util::OpenALSink**
OpenAL (p. 755) Listener.
- class **gazebo::util::OpenALSource**
OpenAL (p. 755) Source.

Macros

- #define **DIAG_TIMER_LAP**(_name, _prefix) ((void)0)
- #define **DIAG_TIMER_START**(_name) ((void) 0)
- #define **DIAG_TIMER_STOP**(_name) ((void) 0)

8.12.1 Detailed Description

8.12.2 Macro Definition Documentation

8.12.2.1 #define **DIAG_TIMER_LAP**(*_name*, *_prefix*)((void)0)

8.12.2.2 #define **DIAG_TIMER_START**(*_name*)((void) 0)

8.12.2.3 #define **DIAG_TIMER_STOP**(*_name*)((void) 0)

Chapter 9

Namespace Documentation

9.1 boost Namespace Reference

9.2 gazebo Namespace Reference

Forward declarations for the common classes.

Namespaces

- namespace **common**
Common namespace.
- namespace **event**
***Event** (p. 416) namespace.*
- namespace **math**
Math namespace.
- namespace **msgs**
Messages namespace.
- namespace **physics**
namespace for physics
- namespace **rendering**
Rendering namespace.
- namespace **sensors**
Sensors namespace.
- namespace **transport**
- namespace **util**

Classes

- class **Master**
A manager that directs topic connections, enables each gazebo network client to locate one another for peer-to-peer communication.
- class **ModelPlugin**
*A plugin with access to **physics::Model** (p. 678).*

- class **PluginT**
A class which all plugins must inherit from.
- class **SensorPlugin**
A plugin with access to `physics::Sensor`.
- class **Server**
- class **SystemPlugin**
A plugin loaded within the gzserver on startup.
- class **VisualPlugin**
A plugin loaded within the gzserver on startup.
- class **WorldPlugin**
A plugin with access to `physics::World` (p. 1239).

Typedefs

- typedef boost::shared_ptr
< GUIPlugin > **GUIPluginPtr**
- typedef boost::shared_ptr
< ModelPlugin > **ModelPluginPtr**
- typedef boost::shared_ptr
< SensorPlugin > **SensorPluginPtr**
- typedef boost::shared_ptr
< SystemPlugin > **SystemPluginPtr**
- typedef boost::shared_ptr
< VisualPlugin > **VisualPluginPtr**
- typedef boost::shared_ptr
< WorldPlugin > **WorldPluginPtr**

Enumerations

- enum **PluginType** {
WORLD_PLUGIN, **MODEL_PLUGIN**, **SENSOR_PLUGIN**, **SYSTEM_PLUGIN**,
VISUAL_PLUGIN }
- Used to specify the type of plugin.*

Functions

- **GAZEBO_VISIBLE** void **add_plugin** (const std::string &_filename) **GAZEBO_DEPRECATED**(2.3)
Deprecated.
- **GAZEBO_VISIBLE** void **addPlugin** (const std::string &_filename)
Add a system plugin.
- **GAZEBO_VISIBLE** std::string **find_file** (const std::string &_file) **GAZEBO_DEPRECATED**(2.3)
Not implemented.
- **GAZEBO_VISIBLE** void **fini** () **GAZEBO_DEPRECATED**(2.3)
Deprecated.
- **GAZEBO_VISIBLE** bool **init** () **GAZEBO_DEPRECATED**(2.3)
Deprecated.
- **GAZEBO_VISIBLE** bool **load** (int _argc=0, char **_argv=0) **GAZEBO_DEPRECATED**(2.3)
Deprecated.

- **GAZEBO_VISIBLE**
gazebo::physics::WorldPtr loadWorld (const std::string &_worldFile)
Create and load a new world from an SDF world file.
- **GAZEBO_VISIBLE** void **print_version** () **GAZEBO_DEPRECATED**(2.3)
Deprecated.
- **GAZEBO_VISIBLE** void **printVersion** ()
Output version information to the terminal.
- **GAZEBO_VISIBLE** void **run** () **GAZEBO_DEPRECATED**(2.3)
Deprecated.
- **GAZEBO_VISIBLE** void **runWorld** (gazebo::physics::WorldPtr _world, unsigned int _iterations)
Run a world for a specific number of iterations.
- **GAZEBO_VISIBLE** bool **setupClient** (int _argc=0, char **_argv=0)
Start a gazebo client.
- **GAZEBO_VISIBLE** bool **setupServer** (int _argc=0, char **_argv=0)
Start a gazebo server.
- **GAZEBO_VISIBLE** bool **shutdown** ()
Stop and cleanup simulation.
- **GAZEBO_VISIBLE** void **stop** () **GAZEBO_DEPRECATED**(2.3)
Deprecated.

9.2.1 Detailed Description

Forward declarations for the common classes. Forward declarations for the util classes.

9.2.2 Typedef Documentation

9.2.2.1 typedef boost::shared_ptr<GUIPlugin> gazebo::GUIPluginPtr

9.2.2.2 typedef boost::shared_ptr<ModelPlugin> gazebo::ModelPluginPtr

9.2.2.3 typedef boost::shared_ptr<SensorPlugin> gazebo::SensorPluginPtr

9.2.2.4 typedef boost::shared_ptr<SystemPlugin> gazebo::SystemPluginPtr

9.2.2.5 typedef boost::shared_ptr<VisualPlugin> gazebo::VisualPluginPtr

9.2.2.6 typedef boost::shared_ptr<WorldPlugin> gazebo::WorldPluginPtr

9.2.3 Function Documentation

9.2.3.1 **GAZEBO_VISIBLE** void gazebo::add_plugin (const std::string &_filename)

Deprecated.

See Also

gazebo::addPlugin (p. 104).

9.2.3.2 GAZEBO_VISIBLE void gazebo::addPlugin (const std::string & *_filename*)

Add a system plugin.

Parameters

<i>in</i>	<i>_filename</i>	Path to the plugin.
-----------	------------------	---------------------

9.2.3.3 GAZEBO_VISIBLE std::string gazebo::find_file (const std::string & *_file*)

Not implemented.

See Also

`gazebo::common::findFile`.

9.2.3.4 GAZEBO_VISIBLE void gazebo::fini ()

Deprecated.

See Also

`gazebo::shutdown` (p. 106).

9.2.3.5 GAZEBO_VISIBLE bool gazebo::init ()

Deprecated.

See Also

`gazebo::setupClient` (p. 105).

`gazebo::setupServer` (p. 106).

9.2.3.6 GAZEBO_VISIBLE bool gazebo::load (int *_argc* = 0, char ** *_argv* = 0)

Deprecated.

See Also

`gazebo::setupClient` (p. 105).

`gazebo::setupServer` (p. 106).

9.2.3.7 GAZEBO_VISIBLE gazebo::physics::WorldPtr gazebo::loadWorld (const std::string & *_worldFile*)

Create and load a new world from an SDF world file.

Parameters

<i>in</i>	<i>_worldFile</i>	The world file to load from.
-----------	-------------------	------------------------------

Returns

Pointer to the created world. NULL on error.

9.2.3.8 GAZEBO_VISIBLE void gazebo::printVersion ()

Deprecated.

See Also

gazebo::printVersion (p. 105).

9.2.3.9 GAZEBO_VISIBLE void gazebo::printVersion ()

Output version information to the terminal.

9.2.3.10 GAZEBO_VISIBLE void gazebo::run ()

Deprecated.

See Also

gazebo::setupClient (p. 105).

gazebo::setupServer (p. 106).

9.2.3.11 GAZEBO_VISIBLE void gazebo::runWorld (gazebo::physics::WorldPtr *_world*, unsigned int *_iterations*)

Run a world for a specific number of iterations.

Parameters

<i>in</i>	<i>_world</i>	Pointer to a world.
<i>in</i>	<i>_iterations</i>	Number of iterations to execute.

9.2.3.12 GAZEBO_VISIBLE bool gazebo::setupClient (int *_argc* = 0, char ** *_argv* = 0)

Start a gazebo client.

This starts transportation, and makes it possible to connect to a running simulation

Parameters

<i>in</i>	<i>_argc</i>	Number of commandline arguments.
<i>in</i>	<i>_argv</i>	The commandline arguments.

Returns

True on success

9.2.3.13 GAZEBO_VISIBLE bool gazebo::setupServer (int *_argc* = 0, char ** *_argv* = 0)

Start a gazebo server.

This starts transportation, and makes it possible to create worlds.

Parameters

in	<i>_argc</i>	Number of commandline arguments.
in	<i>_argv</i>	The commandline arguments.

Returns

True on success

9.2.3.14 GAZEBO_VISIBLE bool gazebo::shutdown ()

Stop and cleanup simulation.

Returns

True if the simulation is shutdown; false otherwise.

Referenced by gazebo::transport::Connection::ConnectToShutdown(), and gazebo::transport::Connection::DisconnectShutdown().

9.2.3.15 GAZEBO_VISIBLE void gazebo::stop ()

Deprecated.

See Also

gazebo::shutdown (p. 106).

Referenced by gazebo::event::Events::ConnectStop(), and gazebo::event::Events::DisconnectStop().

9.3 gazebo::common Namespace Reference

Common namespace.

Classes

- class **Animation**
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.
- class **AssertionInternalError**
Class for generating Exceptions which come from gazebo assertions.
- class **AudioDecoder**
An audio decoder based on FFmpeg.
- class **BVHLoader**

- Handles loading BVH animation files.*
- class **ColladaLoader**

Class used to load Collada mesh files.
- class **Color**

Defines a color.
- class **Console**

Container for loggers, and global logging options (such as verbose vs.
- class **Exception**

Class for generating exceptions.
- class **FileLogger**

A logger that outputs messages to a file.
- class **HeightmapData**

Encapsulates a generic heightmap data file.
- class **Image**

Encapsulates an image.
- class **ImageHeightmap**

Encapsulates an image that will be interpreted as a heightmap.
- class **InternalError**

Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs.
- class **KeyEvent**

Generic description of a keyboard event.
- class **KeyFrame**

A key frame in an animation.
- class **Logger**

Terminal logger.
- class **Material**

Encapsulates description of a material.
- class **Mesh**

A 3D mesh.
- class **MeshCSG**

Creates CSG meshes.
- class **MeshLoader**

Base class for loading meshes.
- class **MeshManager**

Maintains and manages all meshes.
- class **ModelDatabase**

Connects to model database, and has utility functions to find models.
- class **ModelDatabasePrivate**

*Private class attributes for **ModelDatabase** (p. 693).*
- class **MouseEvent**

Generic description of a mouse event.
- class **MovingWindowFilter**

*Base class for **MovingWindowFilter** (p. 715).*
- class **MovingWindowFilterPrivate**
- class **NodeAnimation**

Node animation.
- class **NodeAssignment**

Vertex to node weighted assignement for skeleton animation visualization.

- class **NodeTransform**
NodeTransform (p. 742) *Skeleton.hh* (p. 1465) *common/common.hh*
- class **NumericAnimation**
A numeric animation.
- class **NumericKeyFrame**
A keyframe for a NumericAnimation (p. 752).
- class **PID**
Generic PID (p. 782) *controller class.*
- class **PoseAnimation**
A pose animation.
- class **PoseKeyFrame**
A keyframe for a PoseAnimation (p. 805).
- class **Skeleton**
A skeleton.
- class **SkeletonAnimation**
Skeleton (p. 1024) *animation.*
- class **SkeletonNode**
A skeleton node.
- class **SphericalCoordinates**
Convert spherical coordinates for planetary surfaces.
- class **SphericalCoordinatesPrivate**
commmon/common.hh
- class **STLLoader**
Class used to load STL mesh files.
- class **SubMesh**
A child mesh.
- class **SystemPaths**
Functions to handle getting system paths, keeps track of:
- class **Time**
A Time (p. 1099) *class, can be used to hold wall- or sim-time.*
- class **Timer**
A timer class, used to time things in real world walltime.
- class **UpdateInfo**
Information for use in an update event.
- class **Video**
Handle video encoding and decoding using libavcodec.

Typedefs

- typedef boost::shared_ptr
 < **Animation** > **AnimationPtr**
- typedef boost::shared_ptr
 < DiagnosticTimer > **DiagnosticTimerPtr**
- typedef std::map< unsigned int,
 SkeletonNode * > **NodeMap**
- typedef std::map< unsigned int,
 SkeletonNode * >::iterator **NodeMapIter**

- typedef boost::shared_ptr
< **NumericAnimation** > **NumericAnimationPtr**
- typedef std::vector
< common::Param * > **Param_V**
- typedef boost::shared_ptr
< **PoseAnimation** > **PoseAnimationPtr**
- typedef std::map< double,
std::vector< **NodeTransform** > > **RawNodeAnim**
- typedef std::vector
< std::vector< std::pair
< std::string, double > > > **RawNodeWeights**
- typedef std::map< std::string,
RawNodeAnim > **RawSkeletonAnim**
- typedef boost::shared_ptr
< **SphericalCoordinates** > **SphericalCoordinatesPtr**
- typedef std::map< std::string,
std::string > **StrStr_M**

Functions

- **GAZEBO_VISIBLE** void **add_search_path_suffix** (const std::string &_suffix)
*add path suffix to **common::SystemPaths** (p. 1092)*
- **GAZEBO_VISIBLE** std::string **find_file** (const std::string &_file)
*search for file in **common::SystemPaths** (p. 1092)*
- **GAZEBO_VISIBLE** std::string **find_file** (const std::string &_file, bool _searchLocalPath)
*search for file in **common::SystemPaths** (p. 1092)*
- **GAZEBO_VISIBLE** std::string **find_file_path** (const std::string &_file)
*search for a file in **common::SystemPaths** (p. 1092)*
- template<typename T >
GAZEBO_VISIBLE std::string **get_sha1** (const T &_buffer)
Compute the SHA1 hash of an array of bytes.
- **GAZEBO_VISIBLE** void **load** ()
Load the common library.

Variables

- static std::string **PixelFormatNames** []
String names for the pixel formats.
- static const double **SpeedOfLight** = 299792458
Speed of light.

9.3.1 Detailed Description

Common namespace.

9.3.2 Typedef Documentation

9.3.2.1 `typedef boost::shared_ptr<Animation> gazebo::common::AnimationPtr`

9.3.2.2 `typedef boost::shared_ptr<DiagnosticTimer> gazebo::common::DiagnosticTimerPtr`

9.3.2.3 `typedef std::map<unsigned int, SkeletonNode*> gazebo::common::NodeMap`

9.3.2.4 `typedef std::map<unsigned int, SkeletonNode*>::iterator gazebo::common::NodeMapIter`

9.3.2.5 `typedef boost::shared_ptr<NumericAnimation> gazebo::common::NumericAnimationPtr`

9.3.2.6 `typedef std::vector<common::Param*> gazebo::common::Param_V`

9.3.2.7 `typedef boost::shared_ptr<PoseAnimation> gazebo::common::PoseAnimationPtr`

9.3.2.8 `typedef std::map<double, std::vector<NodeTransform> > gazebo::common::RawNodeAnim`

9.3.2.9 `typedef std::vector<std::vector<std::pair<std::string, double> > > gazebo::common::RawNodeWeights`

9.3.2.10 `typedef std::map<std::string, RawNodeAnim> gazebo::common::RawSkeletonAnim`

9.3.2.11 `typedef boost::shared_ptr<SphericalCoordinates> gazebo::common::SphericalCoordinatesPtr`

9.3.2.12 `typedef std::map<std::string, std::string> gazebo::common::StrStr_M`

9.3.3 Variable Documentation

9.3.3.1 `const double gazebo::common::SpeedOfLight = 299792458 [static]`

Speed of light.

9.4 gazebo::event Namespace Reference

Event (p. 416) namespace.

Classes

- class **Connection**
A class that encapsulates a connection.
- class **ConnectionPrivate**
- class **Event**
Base class for all events.
- class **EventPrivate**
- class **Events**
*An **Event** (p. 416) class to get notifications for simulator events.*
- class **EventT**
A class for event processing.
- class **EventTPrivate**

Typedefs

- typedef std::vector
< **ConnectionPtr** > **Connection_V**
- typedef boost::shared_ptr
< **Connection** > **ConnectionPtr**

9.4.1 Detailed Description

Event (p. 416) namespace.

9.4.2 Typedef Documentation

9.4.2.1 typedef std::vector<ConnectionPtr> gazebo::event::Connection_V

9.4.2.2 typedef boost::shared_ptr<Connection> gazebo::event::ConnectionPtr

9.5 gazebo::math Namespace Reference

Math namespace.

Classes

- class **Angle**
An angle and related functions.
- class **Box**
Mathematical representation of a box and related functions.
- class **Matrix3**
A 3x3 matrix class.
- class **Matrix4**
A 3x3 matrix class.
- class **Plane**
A plane and related functions.
- class **Pose**
Encapsulates a position and rotation in three space.
- class **Quaternion**
A quaternion class.
- class **Rand**
Random number generator class.
- class **RotationSpline**
***Spline** (p. 1063) for rotations.*
- class **Spline**
Splines.
- class **Vector2d**
Generic double x, y vector.
- class **Vector2i**
Generic integer x, y vector.

- class **Vector3**
*The **Vector3** (p. 1165) class represents the generic vector containing 3 elements.*
- class **Vector4**
double Generic x, y, z, w vector

Typedefs

- typedef boost::mt19937 **GeneratorType**
- typedef
 boost::normal_distribution
 < double > **NormalRealDist**
- typedef
 boost::variate_generator
 < **GeneratorType**
 &, **NormalRealDist** > **NRealGen**
- typedef
 boost::variate_generator
 < **GeneratorType**
 &, **UniformIntDist** > **UIntGen**
- typedef boost::uniform_int< int > **UniformIntDist**
- typedef boost::uniform_real
 < double > **UniformRealDist**
- typedef
 boost::variate_generator
 < **GeneratorType**
 &, **UniformRealDist** > **URealGen**

Functions

- template<typename T >
T clamp (T _v, T _min, T _max)
Simple clamping function.
- template<typename T >
 bool **equal** (const T &_a, const T &_b, const T &_epsilon=1e-6)
check if two values are equal, within a tolerance
- float **fixnan** (float _v)
Fix a nan value.
- double **fixnan** (double _v)
Fix a nan value.
- bool **isnan** (float _v)
check if a float is NaN
- bool **isnan** (double _v)
check if a double is NaN
- bool **isPowerOfTwo** (unsigned int _x)
is this a power of 2?
- template<typename T >
T max (const std::vector< T > &_values)
get the maximum value of vector of values

- template<typename T >
T **mean** (const std::vector< T > &_values)
get mean of vector of values
- template<typename T >
T **min** (const std::vector< T > &_values)
get the minimum value of vector of values
- double **parseFloat** (const std::string &_input)
parse string into float
- int **parseInt** (const std::string &_input)
parse string into an integer
- template<typename T >
T **precision** (const T &_a, const unsigned int &_precision)
get value at a specified precision
- unsigned int **roundUpPowerOfTwo** (unsigned int _x)
Get the smallest power of two that is greater or equal to a given value.
- template<typename T >
T **variance** (const std::vector< T > &_values)
get variance of vector of values

Variables

- static const double **NAN_D** = std::numeric_limits<double>::quiet_NaN()
Returns the representation of a quiet not a number (NaN)
- static const int **NAN_I** = std::numeric_limits<int>::quiet_NaN()
Returns the representation of a quiet not a number (NaN)

9.5.1 Detailed Description

Math namespace.

9.5.2 Typedef Documentation

9.5.2.1 typedef boost::mt19937 gazebo::math::GeneratorType

9.5.2.2 typedef boost::normal_distribution<double> gazebo::math::NormalRealDist

9.5.2.3 typedef boost::variate_generator<GeneratorType&, NormalRealDist > gazebo::math::NRealGen

9.5.2.4 typedef boost::variate_generator<GeneratorType&, UniformIntDist > gazebo::math::UIntGen

9.5.2.5 typedef boost::uniform_int<int> gazebo::math::UniformIntDist

9.5.2.6 typedef boost::uniform_real<double> gazebo::math::UniformRealDist

9.5.2.7 typedef boost::variate_generator<GeneratorType&, UniformRealDist > gazebo::math::URealGen

9.6 gazebo::msgs Namespace Reference

Messages namespace.

Classes

- class **MsgFactory**
A factory that generates protobuf message based on a string type.

Typedefs

- typedef boost::shared_ptr
 < google::protobuf::Message >(* **MsgFactoryFn**)()

Functions

- **GAZEBO_VISIBLE** msgs::Vector3d **Convert** (const math::Vector3 &_v)
*Convert a **math::Vector3** (p. 1165) to a msgs::Vector3d.*
- **GAZEBO_VISIBLE** msgs::Quaternion **Convert** (const math::Quaternion &_q)
*Convert a **math::Quaternion** (p. 824) to a msgs::Quaternion.*
- **GAZEBO_VISIBLE** msgs::Pose **Convert** (const math::Pose &_p)
*Convert a **math::Pose** (p. 797) to a msgs::Pose.*
- **GAZEBO_VISIBLE** msgs::Color **Convert** (const common::Color &_c)
*Convert a **common::Color** (p. 249) to a msgs::Color.*
- **GAZEBO_VISIBLE** msgs::Time **Convert** (const common::Time &_t)
*Convert a **common::Time** (p. 1099) to a msgs::Time.*
- **GAZEBO_VISIBLE** msgs::PlaneGeom **Convert** (const math::Plane &_p)
*Convert a **math::Plane** (p. 788) to a msgs::PlaneGeom.*
- **GAZEBO_VISIBLE** math::Vector3 **Convert** (const msgs::Vector3d &_v)
Convert a msgs::Vector3d to a math::Vector.
- **GAZEBO_VISIBLE** math::Quaternion **Convert** (const msgs::Quaternion &_q)
*Convert a msgs::Quaternion to a **math::Quaternion** (p. 824).*
- **GAZEBO_VISIBLE** math::Pose **Convert** (const msgs::Pose &_p)
*Convert a msgs::Pose to a **math::Pose** (p. 797).*
- **GAZEBO_VISIBLE** common::Color **Convert** (const msgs::Color &_c)
*Convert a msgs::Color to a **common::Color** (p. 249).*
- **GAZEBO_VISIBLE** common::Time **Convert** (const msgs::Time &_t)
*Convert a msgs::Time to a **common::Time** (p. 1099).*
- **GAZEBO_VISIBLE** math::Plane **Convert** (const msgs::PlaneGeom &_p)
Convert a msgs::PlaneGeom to a common::Plane.
- **GAZEBO_VISIBLE** msgs::Request * **CreateRequest** (const std::string &_request, const std::string &_data="")
Create a request message.
- **GAZEBO_VISIBLE** msgs::Fog **FogFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::Fog from a fog SDF element.
- **GAZEBO_VISIBLE** msgs::Geometry **GeometryFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::Geometry from a geometry SDF element.
- **GAZEBO_VISIBLE** msgs::Header * **GetHeader** (google::protobuf::Message &_message)
Get the header from a protobuf message.
- **GAZEBO_VISIBLE** msgs::GUI **GUIFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::GUI from a GUI SDF element.
- **GAZEBO_VISIBLE** void **Init** (google::protobuf::Message &_message, const std::string &_id="")

Initialize a message.

- **GAZEBO_VISIBLE** msgs::Light **LightFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::Light from a light SDF element.
- **GAZEBO_VISIBLE** msgs::MeshGeom **MeshFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::MeshGeom from a mesh SDF element.
- **GAZEBO_VISIBLE** msgs::Scene **SceneFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::Scene from a scene SDF element.
- **GAZEBO_VISIBLE** void **Set** (common::Image &_img, const msgs::Image &_msg)
*Convert a msgs::Image to a **common::Image** (p. 515).*
- **GAZEBO_VISIBLE** void **Set** (msgs::Image *_msg, const common::Image &_i)
*Set a msgs::Image from a **common::Image** (p. 515).*
- **GAZEBO_VISIBLE** void **Set** (msgs::Vector3d *_pt, const math::Vector3 &_v)
*Set a msgs::Vector3d from a **math::Vector3** (p. 1165).*
- **GAZEBO_VISIBLE** void **Set** (msgs::Vector2d *_pt, const math::Vector2d &_v)
*Set a msgs::Vector2d from a **math::Vector3** (p. 1165).*
- **GAZEBO_VISIBLE** void **Set** (msgs::Quaternion *_q, const math::Quaternion &_v)
*Set a msgs::Quaternion from a **math::Quaternion** (p. 824).*
- **GAZEBO_VISIBLE** void **Set** (msgs::Pose *_p, const math::Pose &_v)
*Set a msgs::Pose from a **math::Pose** (p. 797).*
- **GAZEBO_VISIBLE** void **Set** (msgs::Color *_c, const common::Color &_v)
*Set a msgs::Color from a **common::Color** (p. 249).*
- **GAZEBO_VISIBLE** void **Set** (msgs::Time *_t, const common::Time &_v)
*Set a msgs::Time from a **common::Time** (p. 1099).*
- void **Set** (msgs::SphericalCoordinates *_s, const common::SphericalCoordinates &_v)
*Set a msgs::SphericalCoordinates from a **common::SphericalCoordinates** (p. 1057) object.*
- **GAZEBO_VISIBLE** void **Set** (msgs::PlaneGeom *_p, const math::Plane &_v)
*Set a msgs::Plane from a **math::Plane** (p. 788).*
- **GAZEBO_VISIBLE** void **Stamp** (msgs::Header *_header)
Time stamp a header.
- **GAZEBO_VISIBLE** void **Stamp** (msgs::Time *_time)
Set the time in a time message.
- **GAZEBO_VISIBLE** msgs::TrackVisual **TrackVisualFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::TrackVisual from a track visual SDF element.
- **GAZEBO_VISIBLE** msgs::Visual **VisualFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::Visual from a visual SDF element.

9.6.1 Detailed Description

Messages namespace.

9.6.2 Typedef Documentation

9.6.2.1 typedef boost::shared_ptr<google::protobuf::Message>(* gazebo::msgs::MsgFactoryFn)()

9.7 gazebo::physics Namespace Reference

namespace for physics

Classes

- class **Actor**
Actor (p. 139) class enables GPU based mesh model / skeleton scriptable animation.
- class **BallJoint**
Base (p. 168) class for a ball joint.
- class **Base**
Base (p. 168) class for most physics classes.
- class **BoxShape**
Box geometry primitive.
- class **Collision**
Base (p. 168) class for all collision entities.
- class **CollisionState**
Store state information of a *physics::Collision* (p. 235) object.
- class **Contact**
A contact between two collisions.
- class **ContactManager**
Aggregates all the contact information generated by the collision detection engine.
- class **ContactPublisher**
A custom contact publisher created for each contact filter in the *Contact* (p. 279) Manager.
- class **CylinderShape**
Cylinder collision.
- class **DARTBallJoint**
An *DARTBallJoint* (p. 301).
- class **DARTBoxShape**
DART Box shape.
- class **DARTCollision**
Base (p. 168) class for all DART collisions.
- class **DARTCylinderShape**
DART cylinder shape.
- class **DARTHeightmapShape**
DART Height map collision.
- class **DARTHinge2Joint**
A two axis hinge joint.
- class **DARTHingeJoint**
A single axis hinge joint.
- class **DARTJoint**
DART joint interface.
- class **DARTLink**
DART Link (p. 595) class.
- class **DARTMeshShape**
Triangle mesh collision.
- class **DARTModel**
DART model class.
- class **DARTMultiRayShape**
DART specific version of MultiRayShape (p. 723).
- class **DARTPhysics**

- DART physics engine.*
- class **DARTPlaneShape**
 - An DART Plane shape.*
- class **DARTRayShape**
 - Ray collision.*
- class **DARTScrewJoint**
 - A screw joint.*
- class **DARTSliderJoint**
 - A slider joint.*
- class **DARTSphereShape**
 - A DART sphere shape.*
- class **DARTTypes**
 - A set of functions for converting between the math types used by gazebo and dart.*
- class **DARTUniversalJoint**
 - A universal joint.*
- class **Entity**
 - Base** (p. 168) class for all physics objects in Gazebo.*
- class **FrictionPyramid**
 - Parameters used for friction pyramid model.*
- class **GearboxJoint**
 - A double axis gearbox joint.*
- class **Gripper**
 - A gripper abstraction.*
- class **HeightmapShape**
 - HeightmapShape** (p. 506) collision shape builds a heightmap from an image.*
- class **Hinge2Joint**
 - A two axis hinge joint.*
- class **HingeJoint**
 - A single axis hinge joint.*
- class **Inertial**
 - A class for inertial information about a link.*
- class **Joint**
 - Base** (p. 168) class for all joints.*
- class **JointController**
 - A class for manipulating **physics::Joint** (p. 541).*
- class **JointControllerPrivate**
- class **JointState**
 - keeps track of state of a **physics::Joint** (p. 541)*
- class **JointWrench**
 - Wrench information from a joint.*
- class **Link**
 - Link** (p. 595) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.*
- class **LinkState**
 - Store state information of a **physics::Link** (p. 595) object.*
- class **MeshShape**
 - Triangle mesh collision shape.*

- class **Model**
A model is a collection of links, joints, and plugins.
- class **ModelState**
Store state information of a `physics::Model` (p. 678) object.
- class **MultiRayShape**
Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.
- class **PhysicsEngine**
***Base** (p. 168) class for a physics engine.*
- class **PhysicsFactory**
The physics factory instantiates different physics engines.
- class **PlaneShape**
***Collision** (p. 235) for an infinite plane.*
- class **RayShape**
***Base** (p. 168) class for Ray collision geometry.*
- class **Road**
*for building a **Road** (p. 869) from SDF*
- class **ScrewJoint**
A screw joint, which has both prismatic and rotational DOFs.
- class **Shape**
***Base** (p. 168) class for all shapes.*
- class **SimbodyBallJoint**
***SimbodyBallJoint** (p. 936) class models a ball joint in Simbody.*
- class **SimbodyBoxShape**
Simbody box collision.
- class **SimbodyCollision**
Simbody collisions.
- class **SimbodyCylinderShape**
Cylinder collision.
- class **SimbodyHeightmapShape**
Height map collision.
- class **SimbodyHinge2Joint**
A two axis hinge joint.
- class **SimbodyHingeJoint**
A single axis hinge joint.
- class **SimbodyJoint**
***Base** (p. 168) class for all joints.*
- class **SimbodyLink**
*Simbody **Link** (p. 595) class.*
- class **SimbodyMeshShape**
Triangle mesh collision.
- class **SimbodyModel**
A model is a collection of links, joints, and plugins.
- class **SimbodyMultiRayShape**
*Simbody specific version of **MultiRayShape** (p. 723).*
- class **SimbodyPhysics**
Simbody physics engine.
- class **SimbodyPlaneShape**

- Simbody collision for an infinite plane.*
- class **SimbodyRayShape**
 - Ray shape for simbody.*
- class **SimbodyScrewJoint**
 - A screw joint.*
- class **SimbodySliderJoint**
 - A slider joint.*
- class **SimbodySphereShape**
 - Simbody sphere collision.*
- class **SimbodyUniversalJoint**
 - A simbody universal joint class.*
- class **SliderJoint**
 - A slider joint.*
- class **SphereShape**
 - Sphere collision shape.*
- class **State**
 - State (p. 1068) of an entity.*
- class **SurfaceParams**
 - SurfaceParams (p. 1090) defines various Surface contact parameters.*
- class **TrajectoryInfo**
 - Information about a trajectory for an Actor (p. 139).*
- class **UniversalJoint**
 - A universal joint.*
- class **World**
 - The world provides access to all other object within a simulated environment.*
- class **WorldState**
 - Store state information of a physics::World (p. 1239) object.*

Typedefs

- typedef std::vector< **ActorPtr** > **Actor_V**
- typedef boost::shared_ptr< **Actor** > **ActorPtr**
- typedef std::vector< **BasePtr** > **Base_V**
- typedef boost::shared_ptr< **Base** > **BasePtr**
- typedef boost::shared_ptr< **BoxShape** > **BoxShapePtr**
- typedef std::vector< **CollisionPtr** > **Collision_V**
- typedef boost::shared_ptr< **Collision** > **CollisionPtr**
- typedef boost::shared_ptr< **Contact** > **ContactPtr**
- typedef boost::shared_ptr< **CylinderShape** > **CylinderShapePtr**
- typedef boost::shared_ptr< **DARTCollision** > **DARTCollisionPtr**
- typedef boost::shared_ptr< **DARTJoint** > **DARTJointPtr**
- typedef boost::shared_ptr< **DARTLink** > **DARTLinkPtr**

- typedef boost::shared_ptr
 < **DARTModel** > **DARTModelPtr**
- typedef boost::shared_ptr
 < **DARTPhysics** > **DARTPhysicsPtr**
- typedef boost::shared_ptr
 < **DARTRayShape** > **DARTRayShapePtr**
- typedef boost::shared_ptr< **Entity** > **EntityPtr**
- typedef boost::shared_ptr
 < **Gripper** > **GripperPtr**
- typedef boost::shared_ptr
 < **HeightmapShape** > **HeightmapShapePtr**
- typedef boost::shared_ptr
 < **Inertial** > **InertialPtr**
- typedef std::vector< **JointPtr** > **Joint_V**
- typedef std::vector
 < **JointControllerPtr** > **JointController_V**
- typedef boost::shared_ptr
 < **JointController** > **JointControllerPtr**
- typedef boost::shared_ptr< **Joint** > **JointPtr**
- typedef std::map< std::string,
 JointState > **JointState_M**
- typedef std::vector< **LinkPtr** > **Link_V**
- typedef boost::shared_ptr< **Link** > **LinkPtr**
- typedef std::map< std::string,
 LinkState > **LinkState_M**
- typedef boost::shared_ptr
 < **MeshShape** > **MeshShapePtr**
- typedef std::vector< **ModelPtr** > **Model_V**
- typedef boost::shared_ptr< **Model** > **ModelPtr**
- typedef std::map< std::string,
 ModelState > **ModelState_M**
- typedef boost::shared_ptr
 < **MultiRayShape** > **MultiRayShapePtr**
- typedef boost::shared_ptr
 < **PhysicsEngine** > **PhysicsEnginePtr**
- typedef **PhysicsEnginePtr**(* **PhysicsFactoryFn**)(WorldPtr world)
- typedef boost::shared_ptr
 < **RayShape** > **RayShapePtr**
- typedef boost::shared_ptr< **Road** > **RoadPtr**
- typedef boost::shared_ptr< **Shape** > **ShapePtr**
- typedef boost::shared_ptr
 < **SimbodyCollision** > **SimbodyCollisionPtr**
- typedef boost::shared_ptr
 < **SimbodyLink** > **SimbodyLinkPtr**
- typedef boost::shared_ptr
 < **SimbodyModel** > **SimbodyModelPtr**
- typedef boost::shared_ptr
 < **SimbodyPhysics** > **SimbodyPhysicsPtr**
- typedef boost::shared_ptr
 < **SimbodyRayShape** > **SimbodyRayShapePtr**
- typedef boost::shared_ptr
 < **SphereShape** > **SphereShapePtr**

- typedef boost::shared_ptr
< **SurfaceParams** > **SurfaceParamsPtr**
- typedef boost::shared_ptr< **World** > **WorldPtr**

Functions

- **GAZEBO_VISIBLE WorldPtr create_world** (const std::string &_name="")
Create a world given a name.
- **GAZEBO_VISIBLE bool fini** ()
Finalize transport by calling `gazebo::transport::fini` (p. 94).
- **GAZEBO_VISIBLE WorldPtr get_world** (const std::string &_name="")
Returns a pointer to a world by name.
- **GAZEBO_VISIBLE uint32_t getUniqueId** ()
Get a unique ID.
- **GAZEBO_VISIBLE void init_world** (WorldPtr _world)
Init world given a pointer to it.
- **GAZEBO_VISIBLE void init_worlds** ()
initialize multiple worlds stored in static variable `gazebo::g_worlds`
- **GAZEBO_VISIBLE bool load** ()
Setup `gazebo::SystemPlugin` (p. 1098)'s and call `gazebo::transport::init` (p. 96).
- **GAZEBO_VISIBLE void load_world** (WorldPtr _world, sdf::ElementPtr _sdf)
Load world from `sdf::Element` pointer.
- **GAZEBO_VISIBLE void load_worlds** (sdf::ElementPtr _sdf)
load multiple worlds from single `sdf::Element` pointer
- **GAZEBO_VISIBLE void pause_world** (WorldPtr _world, bool _pause)
Pause world by calling `World::SetPaused` (p. 1250).
- **GAZEBO_VISIBLE void pause_worlds** (bool pause)
pause multiple worlds stored in static variable `gazebo::g_worlds`
- **GAZEBO_VISIBLE void remove_worlds** ()
remove multiple worlds stored in static variable `gazebo::g_worlds`
- **GAZEBO_VISIBLE void run_world** (WorldPtr _world, unsigned int _iterations=0)
Run world by calling `World::Run()` (p. 1249) given a pointer to it.
- **GAZEBO_VISIBLE void run_worlds** (unsigned int _iterations=0)
Run multiple worlds stored in static variable `gazebo::g_worlds`.
- **GAZEBO_VISIBLE void stop_world** (WorldPtr _world)
Stop world by calling `World::Stop()` (p. 1251) given a pointer to it.
- **GAZEBO_VISIBLE void stop_worlds** ()
stop multiple worlds stored in static variable `gazebo::g_worlds`
- **GAZEBO_VISIBLE bool worlds_running** ()
Return true if any world is running.

Variables

- static std::string **EntityTypename** []
String names for the different entity types.

9.7.1 Detailed Description

namespace for physics Physics forward declarations and type defines.

physics namespace

9.7.2 Typedef Documentation

9.7.2.1 typedef std::vector<ActorPtr> gazebo::physics::Actor_V

9.7.2.2 typedef boost::shared_ptr<Actor> gazebo::physics::ActorPtr

9.7.2.3 typedef std::vector<BasePtr> gazebo::physics::Base_V

9.7.2.4 typedef boost::shared_ptr<Base> gazebo::physics::BasePtr

9.7.2.5 typedef boost::shared_ptr<BoxShape> gazebo::physics::BoxShapePtr

9.7.2.6 typedef std::vector<CollisionPtr> gazebo::physics::Collision_V

9.7.2.7 typedef boost::shared_ptr<Collision> gazebo::physics::CollisionPtr

9.7.2.8 typedef boost::shared_ptr<Contact> gazebo::physics::ContactPtr

9.7.2.9 typedef boost::shared_ptr<CylinderShape> gazebo::physics::CylinderShapePtr

9.7.2.10 typedef boost::shared_ptr<DARTCollision> gazebo::physics::DARTCollisionPtr

9.7.2.11 typedef boost::shared_ptr<DARTJoint> gazebo::physics::DARTJointPtr

9.7.2.12 typedef boost::shared_ptr<DARTLink> gazebo::physics::DARTLinkPtr

9.7.2.13 typedef boost::shared_ptr<DARTModel> gazebo::physics::DARTModelPtr

9.7.2.14 typedef boost::shared_ptr<DARTPhysics> gazebo::physics::DARTPhysicsPtr

9.7.2.15 typedef boost::shared_ptr<DARTRayShape> gazebo::physics::DARTRayShapePtr

9.7.2.16 typedef boost::shared_ptr<Entity> gazebo::physics::EntityPtr

9.7.2.17 typedef boost::shared_ptr<Gripper> gazebo::physics::GripperPtr

9.7.2.18 typedef boost::shared_ptr<HeightmapShape> gazebo::physics::HeightmapShapePtr

9.7.2.19 typedef boost::shared_ptr<Inertial> gazebo::physics::InertialPtr

9.7.2.20 typedef std::vector<JointPtr> gazebo::physics::Joint_V

9.7.2.21 typedef std::vector<JointControllerPtr> gazebo::physics::JointController_V

9.7.2.22 typedef boost::shared_ptr<JointController> gazebo::physics::JointControllerPtr

- 9.7.2.23 `typedef boost::shared_ptr<Joint> gazebo::physics::JointPtr`
- 9.7.2.24 `typedef std::map<std::string, JointState> gazebo::physics::JointState_M`
- 9.7.2.25 `typedef std::vector<LinkPtr> gazebo::physics::Link_V`
- 9.7.2.26 `typedef boost::shared_ptr<Link> gazebo::physics::LinkPtr`
- 9.7.2.27 `typedef std::map<std::string, LinkState> gazebo::physics::LinkState_M`
- 9.7.2.28 `typedef boost::shared_ptr<MeshShape> gazebo::physics::MeshShapePtr`
- 9.7.2.29 `typedef std::vector<ModelPtr> gazebo::physics::Model_V`
- 9.7.2.30 `typedef boost::shared_ptr<Model> gazebo::physics::ModelPtr`
- 9.7.2.31 `typedef std::map<std::string, ModelState> gazebo::physics::ModelState_M`
- 9.7.2.32 `typedef boost::shared_ptr<MultiRayShape> gazebo::physics::MultiRayShapePtr`
- 9.7.2.33 `typedef boost::shared_ptr<PhysicsEngine> gazebo::physics::PhysicsEnginePtr`
- 9.7.2.34 `typedef boost::shared_ptr<RayShape> gazebo::physics::RayShapePtr`
- 9.7.2.35 `typedef boost::shared_ptr<Road> gazebo::physics::RoadPtr`
- 9.7.2.36 `typedef boost::shared_ptr<Shape> gazebo::physics::ShapePtr`
- 9.7.2.37 `typedef boost::shared_ptr<SimbodyCollision> gazebo::physics::SimbodyCollisionPtr`
- 9.7.2.38 `typedef boost::shared_ptr<SimbodyLink> gazebo::physics::SimbodyLinkPtr`
- 9.7.2.39 `typedef boost::shared_ptr<SimbodyModel> gazebo::physics::SimbodyModelPtr`
- 9.7.2.40 `typedef boost::shared_ptr<SimbodyPhysics> gazebo::physics::SimbodyPhysicsPtr`
- 9.7.2.41 `typedef boost::shared_ptr<SimbodyRayShape> gazebo::physics::SimbodyRayShapePtr`
- 9.7.2.42 `typedef boost::shared_ptr<SphereShape> gazebo::physics::SphereShapePtr`
- 9.7.2.43 `typedef boost::shared_ptr<SurfaceParams> gazebo::physics::SurfaceParamsPtr`
- 9.7.2.44 `typedef boost::shared_ptr<World> gazebo::physics::WorldPtr`

9.8 gazebo::rendering Namespace Reference

Rendering namespace.

Classes

- class **ArrowVisual**

- Basic arrow visualization.*

 - class **ArrowVisualPrivate**
 - Private data for the Arrow **Visual** (p. 1196) class.*
 - class **AxisVisual**
 - Basic axis visualization.*
 - class **AxisVisualPrivate**
 - Private data for the Axis **Visual** (p. 1196) class.*
 - class **Camera**
 - Basic camera sensor.*
 - class **CameraPrivate**
 - Private data for the **Camera** (p. 197) class.*
 - class **CameraVisual**
 - Basic camera visualization.*
 - class **CameraVisualPrivate**
 - class **COMVisual**
 - Basic Center of Mass visualization.*
 - class **COMVisualPrivate**
 - Private data for the COM **Visual** (p. 1196) class.*
 - class **ContactVisual**
 - Contact visualization.*
 - class **ContactVisualPrivate**
 - Private data for the Arrow **Visual** (p. 1196) class.*
 - class **Conversions**
 - Conversions** (p. 296) **Conversions.hh** (p. 1306) **rendering/Conversions.hh** (p. 1306).*
 - class **DepthCamera**
 - Depth camera used to render depth data into an image buffer.*
 - class **DummyPageProvider**
 - Pretends to provide procedural page content to avoid page loading.*
 - class **DynamicLines**
 - Class for drawing lines that can change.*
 - class **DynamicRenderable**
 - Abstract base class providing mechanisms for dynamically growing hardware buffers.*
 - class **Events**
 - Base class for rendering events.*
 - class **FPSViewController**
 - First Person Shooter style view controller.*
 - class **GpuLaser**
 - GPU based laser distance sensor.*
 - class **Grid**
 - Displays a grid of cells, drawn with lines.*
 - class **GUIOverlay**
 - A class that creates a CEGUI overlay on a render window.*
 - class **GUIOverlayPrivate**
 - Private data for the **GUIOverlay** (p. 494) class.*
 - class **GzTerrainMatGen**
 - class **Heightmap**
 - Rendering a terrain using heightmap information.*

- class **JointVisual**
Visualization for joints.
- class **JointVisualPrivate**
*Private data for the Joint **Visual** (p. 1196) class.*
- class **LaserVisual**
Visualization for laser data.
- class **LaserVisualPrivate**
*Private data for the Laser **Visual** (p. 1196) class.*
- class **Light**
A light source.
- class **MovableText**
Movable text.
- class **OrbitViewController**
Orbit view controller.
- class **Projector**
Projects a material onto surface, light a light projector.
- class **RenderEngine**
Adaptor to Ogre3d.
- class **RFIDTagVisual**
Visualization for RFID tags sensor.
- class **RFIDTagVisualPrivate**
*Private data for the RFID Tag **Visual** (p. 1196) class.*
- class **RFIDVisual**
Visualization for RFID sensor.
- class **RFIDVisualPrivate**
*Private data for the RFID **Visual** (p. 1196) class.*
- class **Road2d**
- class **RTShaderSystem**
*Implements **Ogre** (p. 137)'s Run-Time Shader system.*
- class **Scene**
Representation of an entire scene graph.
- class **SelectionObj**
Interactive selection object for models and links.
- class **SelectionObjPrivate**
Private data for the Selection Obj class.
- class **SonarVisual**
Visualization for sonar data.
- class **SonarVisualPrivate**
*Private data for the Sonar **Visual** (p. 1196) class.*
- class **TransmitterVisual**
Visualization for the wireless propagation data.
- class **TransmitterVisualPrivate**
*Private data for the Transmitter **Visual** (p. 1196) class.*
- class **UserCamera**
A camera used for user visualization of a scene.
- class **UserCameraPrivate**
*Private data for the **UserCamera** (p. 1137) class.*

- class **VideoVisual**
A visual element that displays a video as a texture.
- class **VideoVisualPrivate**
*Private data for the Video **Visual** (p. 1196) class.*
- class **ViewController**
Base class for view controllers.
- class **Visual**
A renderable object.
- class **VisualPrivate**
*Private data for the **Visual** (p. 1196) class.*
- class **WindowManager**
Class to manage render windows.
- class **WireBox**
Draws a wireframe box.
- class **WireBoxPrivate**
*Private data for the **WireBox** (p. 1227) class.*
- class **WrenchVisual**
Visualization for sonar data.
- class **WrenchVisualPrivate**
*Private data for the Wrench **Visual** (p. 1196) class.*

Typedefs

- typedef boost::shared_ptr
 < **ArrowVisual** > **ArrowVisualPtr**
- typedef boost::shared_ptr
 < **AxisVisual** > **AxisVisualPtr**
- typedef boost::shared_ptr< **Camera** > **CameraPtr**
- typedef boost::shared_ptr
 < **CameraVisual** > **CameraVisualPtr**
- typedef boost::shared_ptr
 < **COMVisual** > **COMVisualPtr**
- typedef boost::shared_ptr
 < **ContactVisual** > **ContactVisualPtr**
- typedef boost::shared_ptr
 < **DepthCamera** > **DepthCameraPtr**
- typedef boost::shared_ptr
 < **DynamicLines** > **DynamicLinesPtr**
- typedef boost::shared_ptr
 < **GpuLaser** > **GpuLaserPtr**
- typedef boost::shared_ptr
 < **JointVisual** > **JointVisualPtr**
- typedef boost::shared_ptr
 < **LaserVisual** > **LaserVisualPtr**
- typedef boost::shared_ptr< **Light** > **LightPtr**
- typedef boost::shared_ptr
 < **RFIDTagVisual** > **RFIDTagVisualPtr**
- typedef boost::shared_ptr
 < **RFIDVisual** > **RFIDVisualPtr**

- typedef boost::shared_ptr< **Scene** > **ScenePtr**
- typedef boost::shared_ptr
< **SelectionObj** > **SelectionObjPtr**
- typedef boost::shared_ptr
< **SonarVisual** > **SonarVisualPtr**
- typedef boost::shared_ptr
< **UserCamera** > **UserCameraPtr**
- typedef boost::shared_ptr< **Visual** > **VisualPtr**
- typedef boost::shared_ptr
< **WindowManager** > **WindowManagerPtr**
- typedef boost::shared_ptr
< **WrenchVisual** > **WrenchVisualPtr**

Enumerations

- enum **RenderOpType** {
RENDERING_POINT_LIST = 0, **RENDERING_LINE_LIST** = 1, **RENDERING_LINE_STRIP** = 2, **RENDERING_TRIANGLE_LIST** = 3,
RENDERING_TRIANGLE_STRIP = 4, **RENDERING_TRIANGLE_FAN** = 5, **RENDERING_MESH_RESOURCE** = 6 }

Type of render operation for a drawable.

Functions

- **GAZEBO_VISIBLE rendering::ScenePtr create_scene** (const std::string &_name, bool _enableVisualizations, bool _isServer=false)
*create **rendering::Scene** (p. 879) by name.*
- **GAZEBO_VISIBLE bool fini** ()
teardown rendering engine.
- **GAZEBO_VISIBLE rendering::ScenePtr get_scene** (const std::string &_name="")
*get pointer to **rendering::Scene** (p. 879) by name.*
- **GAZEBO_VISIBLE bool init** ()
init rendering engine.
- **GAZEBO_VISIBLE bool load** ()
load rendering engine.
- **GAZEBO_VISIBLE void remove_scene** (const std::string &_name)
*remove a **rendering::Scene** (p. 879) by name*

9.8.1 Detailed Description

Rendering namespace.

9.8.2 Typedef Documentation

9.8.2.1 typedef boost::shared_ptr< **ArrowVisual** > gazebo::rendering::**ArrowVisualPtr**

9.8.2.2 typedef boost::shared_ptr< **AxisVisual** > gazebo::rendering::**AxisVisualPtr**

- 9.8.2.3 `typedef boost::shared_ptr<Camera> gazebo::rendering::CameraPtr`
- 9.8.2.4 `typedef boost::shared_ptr<CameraVisual> gazebo::rendering::CameraVisualPtr`
- 9.8.2.5 `typedef boost::shared_ptr<COMVisual> gazebo::rendering::COMVisualPtr`
- 9.8.2.6 `typedef boost::shared_ptr<ContactVisual> gazebo::rendering::ContactVisualPtr`
- 9.8.2.7 `typedef boost::shared_ptr<DepthCamera> gazebo::rendering::DepthCameraPtr`
- 9.8.2.8 `typedef boost::shared_ptr<DynamicLines> gazebo::rendering::DynamicLinesPtr`
- 9.8.2.9 `typedef boost::shared_ptr<GpuLaser> gazebo::rendering::GpuLaserPtr`
- 9.8.2.10 `typedef boost::shared_ptr<JointVisual> gazebo::rendering::JointVisualPtr`
- 9.8.2.11 `typedef boost::shared_ptr<LaserVisual> gazebo::rendering::LaserVisualPtr`
- 9.8.2.12 `typedef boost::shared_ptr<Light> gazebo::rendering::LightPtr`
- 9.8.2.13 `typedef boost::shared_ptr<RFIDTagVisual> gazebo::rendering::RFIDTagVisualPtr`
- 9.8.2.14 `typedef boost::shared_ptr<RFIDVisual> gazebo::rendering::RFIDVisualPtr`
- 9.8.2.15 `typedef boost::shared_ptr<Scene> gazebo::rendering::ScenePtr`
- 9.8.2.16 `typedef boost::shared_ptr<SelectionObj> gazebo::rendering::SelectionObjPtr`
- 9.8.2.17 `typedef boost::shared_ptr<SonarVisual> gazebo::rendering::SonarVisualPtr`
- 9.8.2.18 `typedef boost::shared_ptr<UserCamera> gazebo::rendering::UserCameraPtr`
- 9.8.2.19 `typedef boost::shared_ptr<Visual> gazebo::rendering::VisualPtr`
- 9.8.2.20 `typedef boost::shared_ptr<WindowManager> gazebo::rendering::WindowManagerPtr`
- 9.8.2.21 `typedef boost::shared_ptr<WrenchVisual> gazebo::rendering::WrenchVisualPtr`

9.8.3 Enumeration Type Documentation

9.8.3.1 `enum gazebo::rendering::RenderOpType`

Type of render operation for a drawable.

Enumerator:

RENDERING_POINT_LIST A list of points, 1 vertex per point.

RENDERING_LINE_LIST A list of lines, 2 vertices per line.

RENDERING_LINE_STRIP A strip of connected lines, 1 vertex per line plus 1 start vertex.

RENDERING_TRIANGLE_LIST A list of triangles, 3 vertices per triangle.

RENDERING_TRIANGLE_STRIP A strip of triangles, 3 vertices for the first triangle, and 1 per triangle after that.

RENDERING_TRIANGLE_FAN A fan of triangles, 3 vertices for the first triangle, and 1 per triangle after that.

RENDERING_MESH_RESOURCE N/A.

9.9 gazebo::sensors Namespace Reference

Sensors namespace.

Classes

- class **CameraSensor**
Basic camera sensor.
- class **ContactSensor**
Contact sensor.
- class **DepthCameraSensor**
- class **ForceTorqueSensor**
Sensor (p. 907) for measure force and torque on a joint.
- class **GaussianNoiseModel**
Gaussian noise class.
- class **GpsSensor**
GpsSensor (p. 464) to provide position measurement.
- class **GpuRaySensor**
- class **ImageGaussianNoiseModel**
- class **ImuSensor**
An IMU sensor.
- class **MultiCameraSensor**
Multiple camera sensor.
- class **Noise**
Noise (p. 748) models for sensor output signals.
- class **NoiseFactory**
Use this noise manager for creating and loading noise models.
- class **RaySensor**
Sensor (p. 907) with one or more rays.
- class **RFIDSensor**
Sensor (p. 907) class for RFID type of sensor.
- class **RFIDTag**
RFIDTag (p. 861) to interact with RFIDTagSensors.
- class **Sensor**
Base class for sensors.
- class **SensorFactory**
- class **SensorManager**
Class to manage and update all sensors.
- class **SimTimeEvent**

- class **SimTimeEventHandler**
Monitors simulation time, and notifies conditions when a specified time has been reached.
- class **SonarSensor**
Sensor (p. 907) with sonar cone.

- class **WirelessReceiver**
Sensor (p. 907) class for receiving wireless signals.
- class **WirelessTransceiver**
Sensor (p. 907) class for receiving wireless signals.
- class **WirelessTransmitter**
Transmitter to send wireless signals.

Typedefs

- typedef std::vector
< **CameraSensorPtr** > **CameraSensor_V**
- typedef boost::shared_ptr
< **CameraSensor** > **CameraSensorPtr**
- typedef std::vector
< **ContactSensorPtr** > **ContactSensor_V**
- typedef boost::shared_ptr
< **ContactSensor** > **ContactSensorPtr**
- typedef std::vector
< **DepthCameraSensorPtr** > **DepthCameraSensor_V**
- typedef boost::shared_ptr
< **DepthCameraSensor** > **DepthCameraSensorPtr**
- typedef boost::shared_ptr
< **ForceTorqueSensor** > **ForceTorqueSensorPtr**
- typedef boost::shared_ptr
< **GaussianNoiseModel** > **GaussianNoiseModelPtr**
- typedef boost::shared_ptr
< **GpsSensor** > **GpsSensorPtr**
- typedef std::vector
< **GpuRaySensorPtr** > **GpuRaySensor_V**
- typedef boost::shared_ptr
< **GpuRaySensor** > **GpuRaySensorPtr**
- typedef boost::shared_ptr
< **ImageGaussianNoiseModel** > **ImageGaussianNoiseModelPtr**
*Shared pointer to **Noise** (p. 748).*
- typedef std::vector< **ImuSensorPtr** > **ImuSensor_V**
- typedef boost::shared_ptr
< **ImuSensor** > **ImuSensorPtr**
- typedef std::vector
< **MultiCameraSensorPtr** > **MultiCameraSensor_V**
- typedef boost::shared_ptr
< **MultiCameraSensor** > **MultiCameraSensorPtr**
- typedef boost::shared_ptr< **Noise** > **NoisePtr**
- typedef std::vector< **RaySensorPtr** > **RaySensor_V**
- typedef boost::shared_ptr
< **RaySensor** > **RaySensorPtr**
- typedef std::vector< **RFIDSensor** > **RFIDSensor_V**
- typedef boost::shared_ptr
< **RFIDSensor** > **RFIDSensorPtr**
- typedef std::vector< **RFIDTag** > **RFIDTag_V**
- typedef boost::shared_ptr
< **RFIDTag** > **RFIDTagPtr**

- typedef std::vector< **SensorPtr** > **Sensor_V**
- typedef **Sensor** *(* **SensorFactoryFn**)()
- typedef boost::shared_ptr< **Sensor** > **SensorPtr**
- typedef boost::shared_ptr
< **SonarSensor** > **SonarSensorPtr**
- typedef std::vector
< **WirelessReceiver** > **WirelessReceiver_V**
- typedef boost::shared_ptr
< **WirelessReceiver** > **WirelessReceiverPtr**
- typedef std::vector
< **WirelessTransceiver** > **WirelessTransceiver_V**
- typedef boost::shared_ptr
< **WirelessTransceiver** > **WirelessTransceiverPtr**
- typedef std::vector
< **WirelessTransmitter** > **WirelessTransmitter_V**
- typedef boost::shared_ptr
< **WirelessTransmitter** > **WirelessTransmitterPtr**

Enumerations

- enum **SensorCategory** { **IMAGE** = 0, **RAY** = 1, **OTHER** = 2, **CATEGORY_COUNT** = 3 }
- SensorClass is used to categorize sensors.*

Functions

- **GAZEBO_VISIBLE** std::string **create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName) **GAZEBO_DEPRECATED**(2.0)
Deprecated.
- **GAZEBO_VISIBLE** std::string **create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName, uint32_t _parentId)
Create a sensor using SDF.
- **GAZEBO_VISIBLE** void **disable** ()
Disable sensors.
- **GAZEBO_VISIBLE** void **enable** ()
Enable sensors.
- **GAZEBO_VISIBLE** bool **fini** ()
shutdown the sensor generation loop.
- **GAZEBO_VISIBLE** **SensorPtr** **get_sensor** (const std::string &_name)
Get a sensor using by name.
- **GAZEBO_VISIBLE** bool **init** ()
initialize the sensor generation loop.
- **GAZEBO_VISIBLE** bool **load** ()
Load the sensor library.
- **GAZEBO_VISIBLE** void **remove_sensor** (const std::string &_sensorName)
Remove a sensor by name.
- **GAZEBO_VISIBLE** bool **remove_sensors** ()
Remove all sensors.
- **GAZEBO_VISIBLE** void **run_once** (bool _force=false)

Run the sensor generation one step.

- **GAZEBO_VISIBLE** void **run_threads** ()

Run sensors in a threads. This is a non-blocking call.

- **GAZEBO_VISIBLE** void **stop** ()

Stop the sensor generation loop.

9.9.1 Detailed Description

Sensors namespace.

9.9.2 Typedef Documentation

9.9.2.1 typedef std::vector<CameraSensorPtr> gazebo::sensors::CameraSensor_V

9.9.2.2 typedef boost::shared_ptr<CameraSensor> gazebo::sensors::CameraSensorPtr

9.9.2.3 typedef std::vector<ContactSensorPtr> gazebo::sensors::ContactSensor_V

9.9.2.4 typedef boost::shared_ptr<ContactSensor> gazebo::sensors::ContactSensorPtr

9.9.2.5 typedef std::vector<DepthCameraSensorPtr> gazebo::sensors::DepthCameraSensor_V

9.9.2.6 typedef boost::shared_ptr<DepthCameraSensor> gazebo::sensors::DepthCameraSensorPtr

9.9.2.7 typedef boost::shared_ptr<ForceTorqueSensor> gazebo::sensors::ForceTorqueSensorPtr

9.9.2.8 typedef boost::shared_ptr<GaussianNoiseModel> gazebo::sensors::GaussianNoiseModelPtr

9.9.2.9 typedef boost::shared_ptr<GpsSensor> gazebo::sensors::GpsSensorPtr

9.9.2.10 typedef std::vector<GpuRaySensorPtr> gazebo::sensors::GpuRaySensor_V

9.9.2.11 typedef boost::shared_ptr<GpuRaySensor> gazebo::sensors::GpuRaySensorPtr

9.9.2.12 typedef boost::shared_ptr<ImageGaussianNoiseModel> gazebo::sensors::ImageGaussianNoiseModelPtr

Shared pointer to **Noise** (p. 748).

9.9.2.13 typedef std::vector<ImuSensorPtr> gazebo::sensors::ImuSensor_V

9.9.2.14 typedef boost::shared_ptr<ImuSensor> gazebo::sensors::ImuSensorPtr

9.9.2.15 typedef std::vector<MultiCameraSensorPtr> gazebo::sensors::MultiCameraSensor_V

9.9.2.16 typedef boost::shared_ptr<MultiCameraSensor> gazebo::sensors::MultiCameraSensorPtr

9.9.2.17 typedef boost::shared_ptr<Noise> gazebo::sensors::NoisePtr

9.9.2.18 typedef std::vector<RaySensorPtr> gazebo::sensors::RaySensor_V

- 9.9.2.19 `typedef boost::shared_ptr<RaySensor> gazebo::sensors::RaySensorPtr`
- 9.9.2.20 `typedef std::vector<RFIDSensor> gazebo::sensors::RFIDSensor_V`
- 9.9.2.21 `typedef boost::shared_ptr<RFIDSensor> gazebo::sensors::RFIDSensorPtr`
- 9.9.2.22 `typedef std::vector<RFIDTag> gazebo::sensors::RFIDTag_V`
- 9.9.2.23 `typedef boost::shared_ptr<RFIDTag> gazebo::sensors::RFIDTagPtr`
- 9.9.2.24 `typedef std::vector<SensorPtr> gazebo::sensors::Sensor_V`
- 9.9.2.25 `typedef Sensor*(* gazebo::sensors::SensorFactoryFn)()`
- 9.9.2.26 `typedef boost::shared_ptr<Sensor> gazebo::sensors::SensorPtr`
- 9.9.2.27 `typedef boost::shared_ptr<SonarSensor> gazebo::sensors::SonarSensorPtr`
- 9.9.2.28 `typedef std::vector<WirelessReceiver> gazebo::sensors::WirelessReceiver_V`
- 9.9.2.29 `typedef boost::shared_ptr<WirelessReceiver> gazebo::sensors::WirelessReceiverPtr`
- 9.9.2.30 `typedef std::vector<WirelessTransceiver> gazebo::sensors::WirelessTransceiver_V`
- 9.9.2.31 `typedef boost::shared_ptr<WirelessTransceiver> gazebo::sensors::WirelessTransceiverPtr`
- 9.9.2.32 `typedef std::vector<WirelessTransmitter> gazebo::sensors::WirelessTransmitter_V`
- 9.9.2.33 `typedef boost::shared_ptr<WirelessTransmitter> gazebo::sensors::WirelessTransmitterPtr`

9.9.3 Enumeration Type Documentation

9.9.3.1 enum gazebo::sensors::SensorCategory

SensorClass is used to categorize sensors.

This is used to put sensors into different threads.

Enumerator:

IMAGE Image based sensor class. This type requires the rendering engine.

RAY Ray based sensor class.

OTHER A type of sensor is not a RAY or IMAGE sensor.

CATEGORY_COUNT Number of **Sensor** (p. 907) Categories.

9.10 gazebo::transport Namespace Reference

Classes

- class **CallbackHelper**
A helper class to handle callbacks when messages arrive.

- class **CallbackHelperT**
Callback helper Template.
- class **Connection**
Single TCP/IP connection manager.
- class **ConnectionManager**
Manager of connections.
- class **ConnectionReadTask**

- class **IOManager**
Manages boost::asio IO.
- class **Node**
A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.
- class **Publication**
A publication for a topic.
- class **PublicationTransport**
transport/transport.hh
- class **Publisher**
A publisher of messages on a topic.
- class **PublishTask**

- class **RawCallbackHelper**
Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.
- class **SubscribeOptions**
Options for a subscription.
- class **Subscriber**
A subscriber to a topic.
- class **SubscriptionTransport**
transport/transport.hh
- class **TopicManager**
Manages topics and their subscriptions.

Typedefs

- typedef boost::shared_ptr
< **CallbackHelper** > **CallbackHelperPtr**
*boost shared pointer to **transport::CallbackHelper** (p. 192)*
- typedef boost::shared_ptr
< **Connection** > **ConnectionPtr**
- typedef boost::shared_ptr
< google::protobuf::Message > **MessagePtr**
- typedef boost::shared_ptr< **Node** > **NodePtr**
- typedef boost::shared_ptr
< **Publication** > **PublicationPtr**
- typedef boost::shared_ptr
< **PublicationTransport** > **PublicationTransportPtr**
- typedef boost::shared_ptr
< **Publisher** > **PublisherPtr**
- typedef boost::shared_ptr
< **Subscriber** > **SubscriberPtr**
- typedef boost::shared_ptr
< **SubscriptionTransport** > **SubscriptionTransportPtr**

Functions

- **GAZEBO_VISIBLE** void **clear_buffers** ()
Clear any remaining communication buffers.
- **GAZEBO_VISIBLE**
transport::ConnectionPtr connectToMaster ()
Create a connection to master.
- **GAZEBO_VISIBLE** void **fini** ()
Cleanup the transport component.
- **GAZEBO_VISIBLE** bool **get_master_uri** (std::string &_master_host, unsigned int &_master_port)
Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.
- **GAZEBO_VISIBLE** void **get_topic_namespaces** (std::list< std::string > &_namespaces)
Return all the namespace (world names) on the master.
- **GAZEBO_VISIBLE** std::map
< std::string, std::list
< std::string > > **getAdvertisedTopics** ()
Get a list of all the topics and their message types.
- **GAZEBO_VISIBLE** std::list
< std::string > **getAdvertisedTopics** (const std::string &_msgType)
Get a list of all the unique advertised topic names.
- **GAZEBO_VISIBLE** bool **getMinimalComms** ()
Get whether minimal comms has been enabled.
- **GAZEBO_VISIBLE** std::string **getTopicMsgType** (const std::string &_topicName)
Get the message typename that is published on the given topic.
- **GAZEBO_VISIBLE** bool **init** (const std::string &_masterHost="", unsigned int _masterPort=0, uint32_t _timeoutIterations=30)
Initialize the transport system.
- bool **is_stopped** ()
Is the transport system stopped?
- **GAZEBO_VISIBLE** void **pause_incoming** (bool _pause)
Pause or unpaue incoming messages.
- template<typename M >
GAZEBO_VISIBLE void **publish** (const std::string &_topic, const google::protobuf::Message &_message)
A convenience function for a one-time publication of a message.
- **GAZEBO_VISIBLE**
boost::shared_ptr
< msgs::Response > **request** (const std::string &_worldName, const std::string &_request, const std::string &_data="")
Send a request and receive a response.
- **GAZEBO_VISIBLE** void **requestNoReply** (const std::string &_worldName, const std::string &_request, const std::string &_data="")
Send a request and don't wait for a response.
- **GAZEBO_VISIBLE** void **requestNoReply** (NodePtr _node, const std::string &_request, const std::string &_data="")
Send a request and don't wait for a response.
- **GAZEBO_VISIBLE** void **run** ()
Run the transport component.
- **GAZEBO_VISIBLE** void **setMinimalComms** (bool _enabled)
Set whether minimal comms should be used.

- **GAZEBO_VISIBLE** void **stop** ()
Stop the transport component from running.
- **GAZEBO_VISIBLE** bool **waitForNamespaces** (const **gazebo::common::Time** &_maxWait)
*Blocks while waiting for topic namespaces from the **Master** (p. 637).*

9.10.1 Typedef Documentation

- 9.10.1.1 typedef boost::shared_ptr<**Connection**> **gazebo::transport::ConnectionPtr**
- 9.10.1.2 typedef boost::shared_ptr<google::protobuf::Message> **gazebo::transport::MessagePtr**
- 9.10.1.3 typedef boost::shared_ptr<**Node**> **gazebo::transport::NodePtr**
- 9.10.1.4 typedef boost::shared_ptr<**Publication**> **gazebo::transport::PublicationPtr**
- 9.10.1.5 typedef boost::shared_ptr<**PublicationTransport**> **gazebo::transport::PublicationTransportPtr**
- 9.10.1.6 typedef boost::shared_ptr<**Publisher**> **gazebo::transport::PublisherPtr**
- 9.10.1.7 typedef boost::shared_ptr<**Subscriber**> **gazebo::transport::SubscriberPtr**
- 9.10.1.8 typedef boost::shared_ptr<**SubscriptionTransport**> **gazebo::transport::SubscriptionTransportPtr**

9.11 gazebo::util Namespace Reference

Classes

- class **DiagnosticManager**
A diagnostic manager class.
- class **DiagnosticTimer**
A timer designed for diagnostics.
- class **LogPlay**
- class **LogRecord**
addtogroup gazebo_util
- class **OpenAL**
3D audio setup and playback.
- class **OpenALSink**
OpenAL (p. 755) Listener.
- class **OpenALSource**
OpenAL (p. 755) Source.

Typedefs

- typedef boost::shared_ptr
< **DiagnosticTimer** > **DiagnosticTimerPtr**
- typedef boost::shared_ptr
< **OpenALSink** > **OpenALSinkPtr**
- typedef boost::shared_ptr
< **OpenALSource** > **OpenALSourcePtr**

9.11.1 Typedef Documentation

9.11.1.1 typedef boost::shared_ptr<DiagnosticTimer> gazebo::util::DiagnosticTimerPtr

9.11.1.2 typedef boost::shared_ptr<OpenALSink> gazebo::util::OpenALSinkPtr

9.11.1.3 typedef boost::shared_ptr<OpenALSource> gazebo::util::OpenALSourcePtr

9.12 google Namespace Reference

Namespaces

- namespace **protobuf**

9.13 google::protobuf Namespace Reference

Namespaces

- namespace **compiler**

9.14 google::protobuf::compiler Namespace Reference

Namespaces

- namespace **cpp**

9.15 google::protobuf::compiler::cpp Namespace Reference

Classes

- class **GazeboGenerator**
Google protobuf message generator for `gazebo::msgs` (p. 113).

9.16 Ogre Namespace Reference

9.17 ogre Namespace Reference

9.18 SimTK Namespace Reference

9.19 SkyX Namespace Reference

Chapter 10

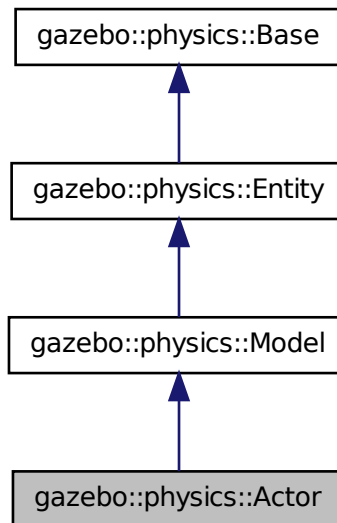
Class Documentation

10.1 gazebo::physics::Actor Class Reference

Actor (p. 139) class enables GPU based mesh model / skeleton scriptable animation.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::Actor:



Public Member Functions

- **Actor** (**BasePtr** _parent)
Constructor.

- virtual `~Actor ()`
Destructor.
- virtual void `Fini ()`
Finalize the actor.
- virtual const sdf::ElementPtr `GetSDF ()`
Get the SDF values for the actor.
- virtual void `Init ()`
Initialize the actor.
- virtual bool `IsActive ()`
Returns true when actor is playing animation.
- void `Load (sdf::ElementPtr _sdf)`
Load the actor.
- virtual void `Play ()`
Start playing the script.
- virtual void `Stop ()`
Stop playing the script.
- void `Update ()`
Update the actor.
- virtual void `UpdateParameters (sdf::ElementPtr _sdf)`
update the parameters using new sdf values.

Protected Attributes

- bool `active`
True if the actor is being updated.
- bool `autoStart`
True if the actor should start running automatically.
- `transport::PublisherPtr` `bonePosePub`
Where to send bone info.
- `std::map< std::string, bool >` `interpolateX`
True to interpolate along x direction.
- `math::Vector3` `lastPos`
Last position of the actor.
- double `lastScriptTime`
Time the script was last updated.
- unsigned int `lastTraj`
The last trajectory.
- bool `loop`
True if the animation should loop.
- `LinkPtr` `mainLink`
Base (p. 168) link.
- const `common::Mesh *` `mesh`
Pointer to the actor's mesh.
- `std::string` `oldAction`
The old action.
- double `pathLength`
Length of the actor's path.

- **common::Time playStartTime**
Time when the animation was started.
- **common::Time prevFrameTime**
Time of the previous frame.
- double **scriptLength**
Time length of a script.
- std::map< std::string,
common::SkeletonAnimation * > skelAnimation
Skeleton animations.
- **common::Skeleton * skeleton**
The actor's skeleton.
- std::map< std::string,
std::map< std::string,
std::string > > **skelNodesMap**
Skeleton to naode map.
- std::string **skinFile**
Filename for the skin.
- double **skinScale**
Scaling factor to apply to the skin.
- double **startDelay**
Amount of time to delay start by.
- std::map< unsigned int,
common::PoseAnimation * > trajectories
All the trajectories.
- std::vector< **TrajectoryInfo** > **trajInfo**
Trajectory information.
- uint32_t **visualId**
ID for this visual.
- std::string **visualName**
Name of the visual.

Additional Inherited Members

10.1.1 Detailed Description

Actor (p. 139) class enables GPU based mesh model / skeleton scriptable animation.

10.1.2 Constructor & Destructor Documentation

10.1.2.1 gazebo::physics::Actor::Actor (BasePtr *_parent*) [explicit]

Constructor.

Parameters

<code>in</code>	<code>_parent</code>	Parent object
-----------------	----------------------	---------------

10.1.2.2 `virtual gazebo::physics::Actor::~~Actor () [virtual]`

Destructor.

10.1.3 Member Function Documentation

10.1.3.1 `virtual void gazebo::physics::Actor::Fini () [virtual]`

Finalize the actor.

Reimplemented from **gazebo::physics::Model** (p. 683).

10.1.3.2 `virtual const sdf::ElementPtr gazebo::physics::Actor::GetSDF () [virtual]`

Get the SDF values for the actor.

Returns

Pointer to the SDF values.

Reimplemented from **gazebo::physics::Model** (p. 686).

10.1.3.3 `virtual void gazebo::physics::Actor::Init () [virtual]`

Initialize the actor.

Reimplemented from **gazebo::physics::Model** (p. 688).

10.1.3.4 `virtual bool gazebo::physics::Actor::IsActive () [virtual]`

Returns true when actor is playing animation.

10.1.3.5 `void gazebo::physics::Actor::Load (sdf::ElementPtr _sdf) [virtual]`

Load the actor.

Parameters

in	_sdf	SDF parameters
----	------	----------------

Reimplemented from **gazebo::physics::Model** (p. 688).

10.1.3.6 `virtual void gazebo::physics::Actor::Play () [virtual]`

Start playing the script.

10.1.3.7 `virtual void gazebo::physics::Actor::Stop () [virtual]`

Stop playing the script.

10.1.3.8 void gazebo::physics::Actor::Update () [virtual]

Update the actor.

Reimplemented from **gazebo::physics::Model** (p. 692).

10.1.3.9 virtual void gazebo::physics::Actor::UpdateParameters (sdf::ElementPtr _sdf) [virtual]

update the parameters using new sdf values.

Parameters

in	_sdf	SDF values to update from.
----	------	----------------------------

Reimplemented from **gazebo::physics::Model** (p. 692).

10.1.4 Member Data Documentation

10.1.4.1 bool gazebo::physics::Actor::active [protected]

True if the actor is being updated.

10.1.4.2 bool gazebo::physics::Actor::autoStart [protected]

True if the actor should start running automatically.

10.1.4.3 transport::PublisherPtr gazebo::physics::Actor::bonePosePub [protected]

Where to send bone info.

10.1.4.4 std::map<std::string, bool> gazebo::physics::Actor::interpolateX [protected]

True to interpolate along x direction.

10.1.4.5 math::Vector3 gazebo::physics::Actor::lastPos [protected]

Last position of the actor.

10.1.4.6 double gazebo::physics::Actor::lastScriptTime [protected]

Time the script was last updated.

10.1.4.7 unsigned int gazebo::physics::Actor::lastTraj [protected]

The last trajectory.

10.1.4.8 `bool gazebo::physics::Actor::loop` [protected]

True if the animation should loop.

10.1.4.9 `LinkPtr gazebo::physics::Actor::mainLink` [protected]

Base (p. 168) link.

10.1.4.10 `const common::Mesh* gazebo::physics::Actor::mesh` [protected]

Pointer to the actor's mesh.

10.1.4.11 `std::string gazebo::physics::Actor::oldAction` [protected]

The old action.

10.1.4.12 `double gazebo::physics::Actor::pathLength` [protected]

Length of the actor's path.

10.1.4.13 `common::Time gazebo::physics::Actor::playStartTime` [protected]

Time when the animation was started.

10.1.4.14 `common::Time gazebo::physics::Actor::prevFrameTime` [protected]

Time of the previous frame.

10.1.4.15 `double gazebo::physics::Actor::scriptLength` [protected]

Time length of a script.

10.1.4.16 `std::map<std::string, common::SkeletonAnimation*> gazebo::physics::Actor::skelAnimation` [protected]

Skeleton animations.

10.1.4.17 `common::Skeleton* gazebo::physics::Actor::skeleton` [protected]

The actor's skeleton.

10.1.4.18 `std::map<std::string, std::map<std::string, std::string>> gazebo::physics::Actor::skelNodesMap` [protected]

Skeleton to node map.

10.1.4.19 `std::string gazebo::physics::Actor::skinFile` [protected]

Filename for the skin.

10.1.4.20 `double gazebo::physics::Actor::skinScale` [protected]

Scaling factor to apply to the skin.

10.1.4.21 `double gazebo::physics::Actor::startDelay` [protected]

Amount of time to delay start by.

10.1.4.22 `std::map<unsigned int, common::PoseAnimation*> gazebo::physics::Actor::trajectories` [protected]

All the trajectories.

10.1.4.23 `std::vector<TrajectoryInfo> gazebo::physics::Actor::trajInfo` [protected]

Trajectory information.

10.1.4.24 `uint32_t gazebo::physics::Actor::visualId` [protected]

ID for this visual.

10.1.4.25 `std::string gazebo::physics::Actor::visualName` [protected]

Name of the visual.

The documentation for this class was generated from the following file:

- **Actor.hh**

10.2 gazebo::math::Angle Class Reference

An angle and related functions.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Angle** ()
Constructor.
- **Angle** (double *_radian*)
Copy Constructor.
- **Angle** (const **Angle** &*_angle*)
Copy constructor.
- virtual **~Angle** ()

Destructor.

- double **Degree** () const
Get the angle in degrees.
- void **Normalize** ()
Normalize the angle in the range $-\pi$ to π .
- bool **operator!=** (const **Angle** &_angle) const
Inequality.
- double **operator*** () const
Dereference operator.
- **Angle operator*** (const **Angle** &_angle) const
*Multiplication operator, result = this * _angle.*
- **Angle operator*=** (const **Angle** &_angle)
*Multiplication set, this = this * _angle.*
- **Angle operator+** (const **Angle** &_angle) const
Addition operator, result = this + _angle.
- **Angle operator+=** (const **Angle** &_angle)
Addition set, this = this + _angle.
- **Angle operator-** (const **Angle** &_angle) const
Subtraction, result = this - _angle.
- **Angle operator-=** (const **Angle** &_angle)
Subtraction set, this = this - _angle.
- **Angle operator/** (const **Angle** &_angle) const
Division, result = this / _angle.
- **Angle operator/=** (const **Angle** &_angle)
Division set, this = this / _angle.
- bool **operator<** (const **Angle** &_angle) const
Less than operator.
- bool **operator<=** (const **Angle** &_angle) const
Less or equal operator.
- bool **operator==** (const **Angle** &_angle) const
Equality operator, result = this == _angle.
- bool **operator>** (const **Angle** &_angle) const
Greater than operator.
- bool **operator>=** (const **Angle** &_angle) const
Greater or equal operator.
- double **Radian** () const
Get the angle in radians.
- void **SetFromDegree** (double _degree)
Set the value from an angle in degrees.
- void **SetFromRadian** (double _radian)
Set the value from an angle in radians.

Static Public Attributes

- static const **Angle HalfPi**
math::Angle (p. 145)($M_PI * 0.5$)
- static const **Angle Pi**
math::Angle(M_PI)
- static const **Angle TwoPi**
math::Angle($M_PI * 2$)
- static const **Angle Zero**
math::Angle(0)

Friends

- `std::ostream & operator<<` (`std::ostream &_out`, const **gazebo::math::Angle** &_a)
Stream insertion operator.
- `std::istream & operator>>` (`std::istream &_in`, **gazebo::math::Angle** &_a)
Stream extraction operator.

10.2.1 Detailed Description

An angle and related functions.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 gazebo::math::Angle::Angle ()

Constructor.

10.2.2.2 gazebo::math::Angle::Angle (double _radian)

Copy Constructor.

Parameters

<code>in</code>	<code>_radian</code>	Radians
-----------------	----------------------	---------

10.2.2.3 gazebo::math::Angle::Angle (const Angle & _angle)

Copy constructor.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 145) to copy
-----------------	---------------------	-------------------------------

10.2.2.4 virtual gazebo::math::Angle::~Angle () [virtual]

Destructor.

10.2.3 Member Function Documentation

10.2.3.1 `double gazebo::math::Angle::Degree () const`

Get the angle in degrees.

Returns

double containing the angle's degree value

10.2.3.2 `void gazebo::math::Angle::Normalize ()`

Normalize the angle in the range -Pi to Pi.

10.2.3.3 `bool gazebo::math::Angle::operator!=(const Angle & _angle) const`

Inequality.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 145) to check for inequality
-----------------	---------------------	---

Returns

true if this != `_angle`

10.2.3.4 `double gazebo::math::Angle::operator*() const` `[inline]`

Dereference operator.

Returns

Double containing the angle's radian value

10.2.3.5 `Angle gazebo::math::Angle::operator*(const Angle & _angle) const`

Multiplication operator, result = this * `_angle`.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 145) for multiplication
-----------------	---------------------	--

Returns

the new angle

10.2.3.6 `Angle gazebo::math::Angle::operator*=(const Angle & _angle)`

Multiplication set, this = this * `_angle`.

Parameters

in	<i>_angle</i>	Angle (p. 145) for multiplication
----	---------------	--

Returns

angle

10.2.3.7 **Angle** gazebo::math::Angle::operator+ (const **Angle** & *_angle*) const

Addition operator, result = this + *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 145) for addition
----	---------------	------------------------------------

Returns

the new angle

10.2.3.8 **Angle** gazebo::math::Angle::operator+= (const **Angle** & *_angle*)

Addition set, this = this + *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 145) for addition
----	---------------	------------------------------------

Returns

angle

10.2.3.9 **Angle** gazebo::math::Angle::operator- (const **Angle** & *_angle*) const

Substraction, result = this - *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 145) for substraction
----	---------------	--

Returns

the new angle

10.2.3.10 **Angle** gazebo::math::Angle::operator-= (const **Angle** & *_angle*)

Subtraction set, this = this - *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 145) for subtraction
----	---------------	---------------------------------------

Returns

angle

10.2.3.11 **Angle** gazebo::math::Angle::operator/ (const **Angle** & *_angle*) const

Division, result = this / *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 145) for division
----	---------------	------------------------------------

Returns

the new angle

10.2.3.12 **Angle** gazebo::math::Angle::operator/= (const **Angle** & *_angle*)

Division set, this = this / *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 145) for division
----	---------------	------------------------------------

Returns

angle

10.2.3.13 **bool** gazebo::math::Angle::operator< (const **Angle** & *_angle*) const

Less than operator.

Parameters

in	<i>_angle</i>	Angle (p. 145) to check
----	---------------	--------------------------------

Returns

true if this < *_angle*

10.2.3.14 **bool** gazebo::math::Angle::operator<= (const **Angle** & *_angle*) const

Less or equal operator.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 145) to check
-----------------	---------------------	--------------------------------

Returns

true if this \leq `_angle`

10.2.3.15 `bool gazebo::math::Angle::operator==(const Angle & _angle) const`

Equality operator, result = this == `_angle`.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 145) to check for equality
-----------------	---------------------	---

Returns

true if this == `_angle`

10.2.3.16 `bool gazebo::math::Angle::operator>(const Angle & _angle) const`

Greater than operator.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 145) to check
-----------------	---------------------	--------------------------------

Returns

true if this $>$ `_angle`

10.2.3.17 `bool gazebo::math::Angle::operator>=(const Angle & _angle) const`

Greater or equal operator.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 145) to check
-----------------	---------------------	--------------------------------

Returns

true if this \geq `_angle`

10.2.3.18 `double gazebo::math::Angle::Radian () const`

Get the angle in radians.

Returns

double containing the angle's radian value

10.2.3.19 void gazebo::math::Angle::SetFromDegree (double *_degree*)

Set the value from an angle in degrees.

Parameters

<i>in</i>	<i>_degree</i>	Degree value
-----------	----------------	--------------

10.2.3.20 void gazebo::math::Angle::SetFromRadian (double *_radian*)

Set the value from an angle in radians.

Parameters

<i>in</i>	<i>_radian</i>	Radian value
-----------	----------------	--------------

10.2.4 Friends And Related Function Documentation**10.2.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::math::Angle & *_a*) [friend]**

Stream insertion operator.

Outputs in degrees

Parameters

<i>in</i>	<i>_out</i>	output stream
<i>in</i>	<i>_a</i>	angle to output

Returns

The output stream

10.2.4.2 std::istream& operator>> (std::istream & *_in*, gazebo::math::Angle & *_a*) [friend]

Stream extraction operator.

Assumes input is in degrees

Parameters

<i>in</i>	input stream
<i>pt</i>	angle to read value into

Returns

The input stream

10.2.5 Member Data Documentation

10.2.5.1 `const Angle gazebo::math::Angle::HalfPi` [static]

`math::Angle` (p. 145)($M_PI * 0.5$)

10.2.5.2 `const Angle gazebo::math::Angle::Pi` [static]

`math::Angle`(M_PI)

10.2.5.3 `const Angle gazebo::math::Angle::TwoPi` [static]

`math::Angle`($M_PI * 2$)

10.2.5.4 `const Angle gazebo::math::Angle::Zero` [static]

`math::Angle`(0)

The documentation for this class was generated from the following file:

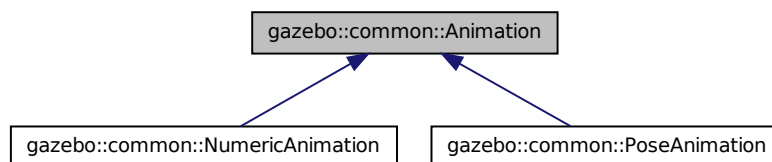
- **Angle.hh**

10.3 gazebo::common::Animation Class Reference

Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.

```
#include <common/common.hh>
```

Inheritance diagram for `gazebo::common::Animation`:



Public Member Functions

- **Animation** (const std::string &_name, double _length, bool _loop)
Constructor.
- virtual **~Animation** ()
Destructor.
- void **AddTime** (double _time)
Add time to the animation.

- **KeyFrame * GetKeyFrame** (unsigned int _index) const
Get a key frame using an index value.
- unsigned int **GetKeyFrameCount** () const
Return the number of key frames in the animation.
- double **GetLength** () const
Return the duration of the animation.
- double **GetTime** () const
Return the current time position.
- void **SetLength** (double _len)
Set the duration of the animation.
- void **SetTime** (double _time)
Set the current time position of the animation.

Protected Types

- typedef std::vector< **KeyFrame * > KeyFrame_V**
array of keyframe type alias

Protected Member Functions

- double **GetKeyFramesAtTime** (double _time, **KeyFrame **_kf1**, **KeyFrame **_kf2**, unsigned int &_firstKeyIndex) const
Get the two key frames that bound a time value.

Protected Attributes

- bool **build**
determines if the interpolation splines need building
- **KeyFrame_V keyFrames**
array of key frames
- double **length**
animation duration
- bool **loop**
true if animation repeats
- std::string **name**
animation name
- double **timePos**
current time position

10.3.1 Detailed Description

Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.

10.3.2 Member Typedef Documentation

10.3.2.1 `typedef std::vector<KeyFrame*> gazebo::common::Animation::KeyFrame_V` [protected]

array of keyframe type alias

10.3.3 Constructor & Destructor Documentation

10.3.3.1 `gazebo::common::Animation::Animation (const std::string & _name, double _length, bool _loop)`

Constructor.

Parameters

in	<code>_name</code>	Name of the animation, should be unique
in	<code>_length</code>	Duration of the animation in seconds
in	<code>_loop</code>	Set to true if the animation should repeat

10.3.3.2 `virtual gazebo::common::Animation::~~Animation ()` [virtual]

Destructor.

10.3.4 Member Function Documentation

10.3.4.1 `void gazebo::common::Animation::AddTime (double _time)`

Add time to the animation.

Parameters

in	<code>_time</code>	The amount of time to add in seconds
----	--------------------	--------------------------------------

10.3.4.2 `KeyFrame* gazebo::common::Animation::GetKeyFrame (unsigned int _index) const`

Get a key frame using an index value.

Parameters

in	<code>_index</code>	The index of the key frame
----	---------------------	----------------------------

Returns

A pointer the keyframe, NULL if the `_index` is invalid

10.3.4.3 `unsigned int gazebo::common::Animation::GetKeyFrameCount () const`

Return the number of key frames in the animation.

Returns

The number of keyframes

10.3.4.4 `double gazebo::common::Animation::GetKeyFramesAtTime (double _time, KeyFrame ** _kf1, KeyFrame ** _kf2, unsigned int & _firstKeyIndex) const` [protected]

Get the two key frames that bound a time value.

Parameters

in	<i>_time</i>	The time in seconds
out	<i>_kf1</i>	Lower bound keyframe that is returned
out	<i>_kf2</i>	Upper bound keyframe that is returned
out	<i>_firstKeyIndex</i>	Index of the lower bound key frame

Returns

The time between the two keyframe

10.3.4.5 `double gazebo::common::Animation::GetLength () const`

Return the duration of the animation.

Returns

Duration of the animation in seconds

10.3.4.6 `double gazebo::common::Animation::GetTime () const`

Return the current time position.

Returns

The time position in seconds

10.3.4.7 `void gazebo::common::Animation::SetLength (double _len)`

Set the duration of the animation.

Parameters

in	<i>_len</i>	The length of the animation in seconds
----	-------------	--

10.3.4.8 `void gazebo::common::Animation::SetTime (double _time)`

Set the current time position of the animation.

Parameters

in	<code>_time</code>	The time position in seconds
----	--------------------	------------------------------

10.3.5 Member Data Documentation

10.3.5.1 `bool gazebo::common::Animation::build` [mutable],[protected]

determines if the interpolation splines need building

10.3.5.2 `KeyFrame_V gazebo::common::Animation::keyFrames` [protected]

array of key frames

10.3.5.3 `double gazebo::common::Animation::length` [protected]

animation duration

10.3.5.4 `bool gazebo::common::Animation::loop` [protected]

true if animation repeats

10.3.5.5 `std::string gazebo::common::Animation::name` [protected]

animation name

10.3.5.6 `double gazebo::common::Animation::timePos` [protected]

current time position

The documentation for this class was generated from the following file:

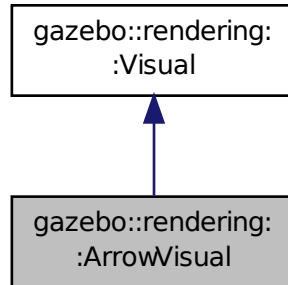
- **Animation.hh**

10.4 gazebo::rendering::ArrowVisual Class Reference

Basic arrow visualization.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::ArrowVisual:



Public Member Functions

- **ArrowVisual** (const std::string &_name, **VisualPtr** _vis)
Constructor.
- virtual ~**ArrowVisual** ()
Destructor.
- virtual void **Load** ()
Load the visual with default parameters.
- void **ShowRotation** ()
Show the rotation of the visual.

Additional Inherited Members

10.4.1 Detailed Description

Basic arrow visualization.

10.4.2 Constructor & Destructor Documentation

10.4.2.1 gazebo::rendering::ArrowVisual::ArrowVisual (const std::string & .name, **VisualPtr** .vis)

Constructor.

Parameters

in	<code>_name</code>	Name of the arrow visual
in	<code>_vis</code>	Pointer to the parent visual

10.4.2.2 virtual gazebo::rendering::ArrowVisual::~~ArrowVisual () [virtual]

Destructor.

10.4.3 Member Function Documentation

10.4.3.1 virtual void gazebo::rendering::ArrowVisual::Load () [virtual]

Load the visual with default parameters.

Reimplemented from **gazebo::rendering::Visual** (p. 1210).

10.4.3.2 void gazebo::rendering::ArrowVisual::ShowRotation ()

Show the rotation of the visual.

The documentation for this class was generated from the following file:

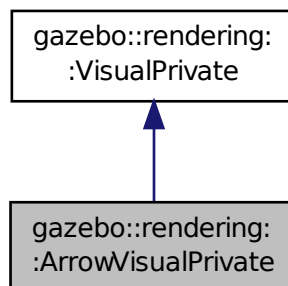
- **ArrowVisual.hh**

10.5 gazebo::rendering::ArrowVisualPrivate Class Reference

Private data for the Arrow **Visual** (p. 1196) class.

```
#include <ArrowVisualPrivate.hh>
```

Inheritance diagram for gazebo::rendering::ArrowVisualPrivate:



Public Attributes

- Ogre::SceneNode * **headNode**
- Ogre::SceneNode * **rotationNode**
- Ogre::SceneNode * **shaftNode**

Additional Inherited Members

10.5.1 Detailed Description

Private data for the Arrow **Visual** (p. 1196) class.

10.5.2 Member Data Documentation

10.5.2.1 `Ogre::SceneNode*` `gazebo::rendering::ArrowVisualPrivate::headNode`

10.5.2.2 `Ogre::SceneNode*` `gazebo::rendering::ArrowVisualPrivate::rotationNode`

10.5.2.3 `Ogre::SceneNode*` `gazebo::rendering::ArrowVisualPrivate::shaftNode`

The documentation for this class was generated from the following file:

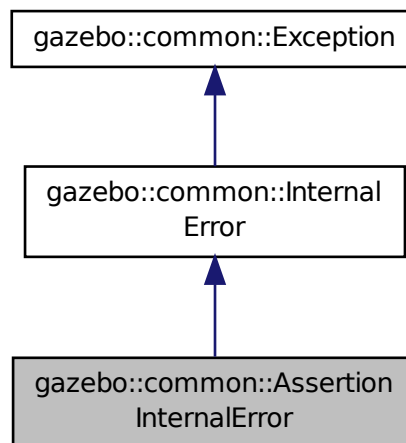
- **ArrowVisualPrivate.hh**

10.6 gazebo::common::AssertionInternalError Class Reference

Class for generating Exceptions which come from gazebo assertions.

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::common::AssertionInternalError:



Public Member Functions

- **AssertionInternalError** (`const char *_file`, `int _line`, `const std::string &_expr`, `const std::string &_function`, `const`

```
std::string &_msg="")
```

Constructor for assertions.

- virtual **~AssertionInternalError** ()

Destructor.

10.6.1 Detailed Description

Class for generating Exceptions which come from gazebo assertions.

They include information about the assertion expression violated, function where problem appeared and assertion debug message.

10.6.2 Constructor & Destructor Documentation

- 10.6.2.1 `gazebo::common::AssertionInternalError::AssertionInternalError (const char * _file, int _line, const std::string & _expr, const std::string & _function, const std::string & _msg = " ")`

Constructor for assertions.

Parameters

in	<i>_file</i>	File name
in	<i>_line</i>	Line number where the error occurred
in	<i>_expr</i>	Assertion expression failed resulting in an internal error
in	<i>_function</i>	Function where assertion failed
in	<i>_msg</i>	Function where assertion failed

- 10.6.2.2 `virtual gazebo::common::AssertionInternalError::~~AssertionInternalError () [virtual]`

Destructor.

The documentation for this class was generated from the following file:

- **Exception.hh**

10.7 gazebo::common::AudioDecoder Class Reference

An audio decoder based on FFmpeg.

```
#include <common/common.hh>
```

Public Member Functions

- **AudioDecoder** ()
Constructor.
- virtual **~AudioDecoder** ()
Destructor.
- bool **Decode** (uint8_t **_outBuffer, unsigned int *_outBufferSize)
Decode the loaded audio file.

- `std::string GetFile ()` const
Get the audio filename that was set.
- `int GetSampleRate ()`
Get the sample rate from the latest decoded file.
- `bool SetFile (const std::string &_filename)`
Set the file to decode.

10.7.1 Detailed Description

An audio decoder based on FFMPEG.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 `gazebo::common::AudioDecoder::AudioDecoder ()`

Constructor.

10.7.2.2 `virtual gazebo::common::AudioDecoder::~~AudioDecoder () [virtual]`

Destructor.

10.7.3 Member Function Documentation

10.7.3.1 `bool gazebo::common::AudioDecoder::Decode (uint8_t ** _outBuffer, unsigned int * _outBufferSize)`

Decode the loaded audio file.

See Also

AudioDecoder::SetFile (p. 163)

Parameters

out	<code>_outBuffer</code>	Buffer that holds the decoded audio data.
out	<code>_outBufferSize</code>	Size of the <code>_outBuffer</code> .

Returns

True if decoding was succesful.

10.7.3.2 `std::string gazebo::common::AudioDecoder::GetFile ()` const

Get the audio filename that was set.

Returns

The name of the set audio file.

See Also

AudioDecoder::SetFile (p. 163)

10.7.3.3 `int gazebo::common::AudioDecoder::GetSampleRate ()`

Get the sample rate from the latest decoded file.

Returns

Integer sample rate, such as 44100.

10.7.3.4 `bool gazebo::common::AudioDecoder::SetFile (const std::string & _filename)`

Set the file to decode.

Parameters

<code>in</code>	<code>_filename</code>	Path to an audio file.
-----------------	------------------------	------------------------

Returns

True if the file was successfull opened.

The documentation for this class was generated from the following file:

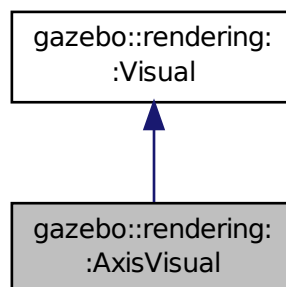
- **AudioDecoder.hh**

10.8 gazebo::rendering::AxisVisual Class Reference

Basic axis visualization.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::AxisVisual:



Public Member Functions

- **AxisVisual** (const std::string &_name, **VisualPtr** _vis)
Constructor.
- virtual **~AxisVisual** ()
Destructor.
- virtual void **Load** ()
Load the axis visual.
- void **ScaleXAxis** (const **math::Vector3** &_scale)
Scale the X axis.
- void **ScaleYAxis** (const **math::Vector3** &_scale)
Scale the Y axis.
- void **ScaleZAxis** (const **math::Vector3** &_scale)
Scale the Z axis.
- void **SetAxisMaterial** (unsigned int _axis, const std::string &_material)
Set the material used to render and axis.
- void **ShowRotation** (unsigned int _axis)
Load the rotation tube.

Additional Inherited Members

10.8.1 Detailed Description

Basic axis visualization.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 gazebo::rendering::AxisVisual::AxisVisual (const std::string & _name, VisualPtr _vis)

Constructor.

Parameters

in	<code>_name</code>	Name of the AxisVisual (p. 163)
in	<code>_vis</code>	Parent visual

10.8.2.2 virtual gazebo::rendering::AxisVisual::~~AxisVisual () [virtual]

Destructor.

10.8.3 Member Function Documentation

10.8.3.1 virtual void gazebo::rendering::AxisVisual::Load () [virtual]

Load the axis visual.

Reimplemented from **gazebo::rendering::Visual** (p. 1210).

10.8.3.2 void gazebo::rendering::AxisVisual::ScaleXAxis (const math::Vector3 & *_scale*)

Scale the X axis.

Parameters

in	<i>_scale</i>	Scaling factor
----	---------------	----------------

10.8.3.3 void gazebo::rendering::AxisVisual::ScaleYAxis (const math::Vector3 & *_scale*)

Scale the Y axis.

Parameters

in	<i>_scale</i>	Scaling factor
----	---------------	----------------

10.8.3.4 void gazebo::rendering::AxisVisual::ScaleZAxis (const math::Vector3 & *_scale*)

Scale the Z axis.

Parameters

in	<i>_scale</i>	Scaling factor
----	---------------	----------------

10.8.3.5 void gazebo::rendering::AxisVisual::SetAxisMaterial (unsigned int *_axis*, const std::string & *_material*)

Set the material used to render and axis.

Parameters

in	<i>_axis</i>	The number of the axis (0, 1, 2 = x,y,z)
in	<i>_material</i>	The name of the material to apply to the axis

10.8.3.6 void gazebo::rendering::AxisVisual::ShowRotation (unsigned int *_axis*)

Load the rotation tube.

The documentation for this class was generated from the following file:

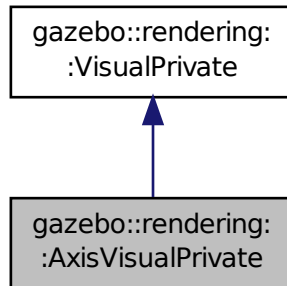
- **AxisVisual.hh**

10.9 gazebo::rendering::AxisVisualPrivate Class Reference

Private data for the Axis **Visual** (p. 1196) class.

```
#include <AxisVisualPrivate.hh>
```

Inheritance diagram for gazebo::rendering::AxisVisualPrivate:



Public Attributes

- **ArrowVisualPtr xAxis**
Pointer to the x-axis visual.
- **ArrowVisualPtr yAxis**
Pointer to the y-axis visual.
- **ArrowVisualPtr zAxis**
Pointer to the z-axis visual.

Additional Inherited Members

10.9.1 Detailed Description

Private data for the Axis **Visual** (p. 1196) class.

10.9.2 Member Data Documentation

10.9.2.1 ArrowVisualPtr gazebo::rendering::AxisVisualPrivate::xAxis

Pointer to the x-axis visual.

10.9.2.2 ArrowVisualPtr gazebo::rendering::AxisVisualPrivate::yAxis

Pointer to the y-axis visual.

10.9.2.3 ArrowVisualPtr gazebo::rendering::AxisVisualPrivate::zAxis

Pointer to the z-axis visual.

The documentation for this class was generated from the following file:

- [AxisVisualPrivate.hh](#)

10.10 gazebo::physics::BallJoint< T > Class Template Reference

Base (p. 168) class for a ball joint.

```
#include <physics/physics.hh>
```

Public Member Functions

- **BallJoint** (**BasePtr** _parent)
Constructor.
- virtual **~BallJoint** ()
Destructor.
- virtual unsigned int **GetAngleCount** () const
- void **Load** (sdf::ElementPtr _sdf)
*Template to ::Load the **BallJoint** (p. 167).*

Protected Member Functions

- virtual void **Init** ()
Initialize joint.

10.10.1 Detailed Description

```
template<class T>class gazebo::physics::BallJoint< T >
```

Base (p. 168) class for a ball joint.

Each physics engine should implement this class.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 `template<class T> gazebo::physics::BallJoint< T >::BallJoint (BasePtr _parent) [inline], [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Pointer to the parent link.
----	----------------------	-----------------------------

10.10.2.2 `template<class T> virtual gazebo::physics::BallJoint< T >::~~BallJoint () [inline], [virtual]`

Destructor.

10.10.3 Member Function Documentation

10.10.3.1 `template<class T> virtual unsigned int gazebo::physics::BallJoint< T >::GetAngleCount () const`
`[inline],[virtual]`

10.10.3.2 `template<class T> virtual void gazebo::physics::BallJoint< T >::Init () [inline],[protected],[virtual]`

Initialize joint.

Reimplemented in `gazebo::physics::DARTBallJoint` (p. 306).

10.10.3.3 `template<class T> void gazebo::physics::BallJoint< T >::Load (sdf::ElementPtr _sdf) [inline]`

Template to `::Load` the **BallJoint** (p. 167).

Parameters

in	<code>_sdf</code>	SDF to load the joint from.
----	-------------------	-----------------------------

Reimplemented in `gazebo::physics::SimbodyBallJoint` (p. 940), and `gazebo::physics::DARTBallJoint` (p. 306).

The documentation for this class was generated from the following file:

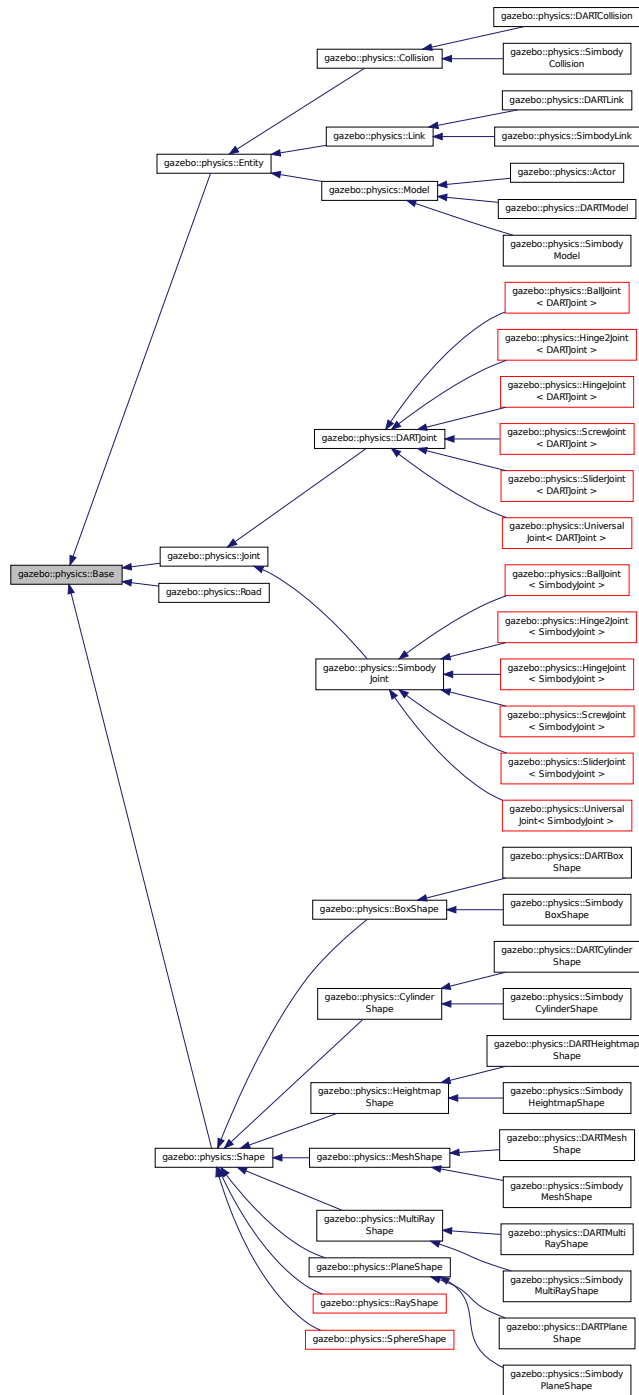
- **BallJoint.hh**

10.11 gazebo::physics::Base Class Reference

Base (p. 168) class for most physics classes.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::Base:



Public Types

- enum **EntityType** {
BASE = 0x00000000, **ENTITY** = 0x00000001, **MODEL** = 0x00000002, **LINK** = 0x00000004,
COLLISION = 0x00000008, **ACTOR** = 0x00000016, **LIGHT** = 0x00000010, **VISUAL** = 0x00000020,
JOINT = 0x00000040, **BALL_JOINT** = 0x00000080, **HINGE2_JOINT** = 0x00000100, **HINGE_JOINT** =
0x00000200,
SLIDER_JOINT = 0x00000400, **SCREW_JOINT** = 0x00000800, **UNIVERSAL_JOINT** = 0x00001000, **GEARB-
OX_JOINT** = 0x00002000,
SHAPE = 0x00010000, **BOX_SHAPE** = 0x00020000, **CYLINDER_SHAPE** = 0x00040000, **HEIGHTMAP_SHA-
PE** = 0x00080000,
MAP_SHAPE = 0x00100000, **MULTIRAY_SHAPE** = 0x00200000, **RAY_SHAPE** = 0x00400000, **PLANE_SHA-
PE** = 0x00800000,
SPHERE_SHAPE = 0x01000000, **MESH_SHAPE** = 0x02000000, **SENSOR_COLLISION** = 0x10000000 }

Unique identifiers for all entity types.

Public Member Functions

- **Base** (**BasePtr** _parent)
Constructor.
- virtual **~Base** ()
Destructor.
- void **AddChild** (**BasePtr** _child)
Add a child to this entity.
- void **AddType** (**EntityType** _type)
Add a type specifier.
- virtual void **Fini** ()
Finalize the object.
- **BasePtr** **GetById** (unsigned int _id) const
This is an internal function.
- **BasePtr** **GetByName** (const std::string &_name)
Get by name.
- **BasePtr** **GetChild** (unsigned int _i) const
Get a child by index.
- **BasePtr** **GetChild** (const std::string &_name)
Get a child by name.
- unsigned int **GetChildCount** () const
Get the number of children.
- uint32_t **GetId** () const
Return the ID of this entity.
- std::string **GetName** () const
Return the name of the entity.
- **BasePtr** **GetParent** () const
Get the parent.
- int **GetParentId** () const
Return the ID of the parent.
- bool **GetSaveable** () const
Get whether the object should be "saved", when the user selects to save the world to xml.
- std::string **GetScopedName** () const

- Return the name of this entity with the model scope world::model1::...::modelN::entityName.*

 - virtual const sdf::ElementPtr **GetSDF** ()

Get the SDF values for the object.
- unsigned int **GetType** () const

Get the full type definition.
- const **WorldPtr** & **GetWorld** () const

*Get the **World** (p. 1239) this object is in.*
- bool **HasType** (const **EntityType** &_t) const

Returns true if this object's type definition has the given type.
- virtual void **Init** ()

Initialize the object.
- bool **IsSelected** () const

True if the entity is selected by the user.
- virtual void **Load** (sdf::ElementPtr _sdf)

Load.
- bool **operator==** (const **Base** &_ent) const

Returns true if the entities are the same.
- void **Print** (const std::string &_prefix)

Print this object to screen via gzmsg.
- virtual void **RemoveChild** (unsigned int _id)

Remove a child from this entity.
- void **RemoveChild** (const std::string &_name)

Remove a child by name.
- void **RemoveChildren** ()

Remove all children.
- virtual void **Reset** ()

Reset the object.
- virtual void **Reset** (**Base::EntityType** _resetType)

*Calls recursive Reset on one of the **Base::EntityType** (p. 172)'s.*
- virtual void **SetName** (const std::string &_name)

Set the name of the entity.
- void **SetParent** (**BasePtr** _parent)

Set the parent.
- void **SetSaveable** (bool _v)

Set whether the object should be "saved", when the user selects to save the world to xml.
- virtual bool **SetSelected** (bool _show)

Set whether this entity has been selected by the user through the gui.
- void **SetWorld** (const **WorldPtr** &_newWorld)

Set the world this object belongs to.
- virtual void **Update** ()

Update the object.
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

Update the parameters using new sdf values.

Protected Member Functions

- void **ComputeScopedName** ()

Compute the scoped name of this object based on its parents.

Protected Attributes

- **Base_V children**
Children of this entity.
- **BasePtr parent**
Parent of this entity.
- sdf::ElementPtr **sdf**
The SDF values for this object.
- **WorldPtr world**
Pointer to the world.

10.11.1 Detailed Description

Base (p. 168) class for most physics classes.

10.11.2 Member Enumeration Documentation

10.11.2.1 enum gazebo::physics::Base::EntityType

Unique identifiers for all entity types.

Enumerator:

- BASE** **Base** (p. 168) type.
- ENTITY** **Entity** (p. 404) type.
- MODEL** **Model** (p. 678) type.
- LINK** **Link** (p. 595) type.
- COLLISION** **Collision** (p. 235) type.
- ACTOR** **Actor** (p. 139) type.
- LIGHT** **Light** type.
- VISUAL** **Visual** type.
- JOINT** **Joint** (p. 541) type.
- BALL_JOINT** **BallJoint** (p. 167) type.
- HINGE2_JOINT** **Hing2Joint** type.
- HINGE_JOINT** **HingeJoint** (p. 513) type.
- SLIDER_JOINT** **SliderJoint** (p. 1044) type.
- SCREW_JOINT** **ScrewJoint** (p. 896) type.
- UNIVERSAL_JOINT** **UniversalJoint** (p. 1135) type.
- GEARBOX_JOINT** **GearboxJoint** (p. 462) type.
- SHAPE** **Shape** (p. 932) type.
- BOX_SHAPE** **BoxShape** (p. 185) type.
- CYLINDER_SHAPE** **CylinderShape** (p. 298) type.
- HEIGHTMAP_SHAPE** **HeightmapShape** (p. 506) type.
- MAP_SHAPE** **MapShape** type.
- MULTIRAY_SHAPE** **MultiRayShape** (p. 723) type.

RAY_SHAPE RayShape (p. 849) type.

PLANE_SHAPE PlaneShape (p. 791) type.

SPHERE_SHAPE SphereShape (p. 1054) type.

MESH_SHAPE MeshShape (p. 675) type.

SENSOR_COLLISION Indicates a collision shape used for sensing.

10.11.3 Constructor & Destructor Documentation

10.11.3.1 gazebo::physics::Base::Base (BasePtr *_parent*) [explicit]

Constructor.

Parameters

<i>in</i>	<i>_parent</i>	Parent of this object
-----------	----------------	-----------------------

10.11.3.2 virtual gazebo::physics::Base::~Base () [virtual]

Destructor.

10.11.4 Member Function Documentation

10.11.4.1 void gazebo::physics::Base::AddChild (BasePtr *_child*)

Add a child to this entity.

Parameters

<i>in</i>	<i>_child</i>	Child entity.
-----------	---------------	---------------

10.11.4.2 void gazebo::physics::Base::AddType (EntityType *_type*)

Add a type specifier.

Parameters

<i>in</i>	<i>_type</i>	New type to append to this objects type definition.
-----------	--------------	---

10.11.4.3 void gazebo::physics::Base::ComputeScopedName () [protected]

Compute the scoped name of this object based on its parents.

See Also

Base::GetScopedName (p. 176)

10.11.4.4 virtual void gazebo::physics::Base::Fini () [virtual]

Finalize the object.

Reimplemented in **gazebo::physics::Joint** (p. 549), **gazebo::physics::Actor** (p. 142), **gazebo::physics::Link** (p. 604), **gazebo::physics::Model** (p. 683), **gazebo::physics::Entity** (p. 407), **gazebo::physics::DARTModel** (p. 351), **gazebo::physics::Collision** (p. 239), **gazebo::physics::DARTLink** (p. 342), **gazebo::physics::Simbody-Link** (p. 976), and **gazebo::physics::DARTCollision** (p. 311).

10.11.4.5 BasePtr gazebo::physics::Base::GetById (unsigned int *_id*) const

This is an internal function.

Get a child or self by id.

Parameters

<i>in</i>	<i>_id</i>	ID of the object to retrieve.
-----------	------------	-------------------------------

Returns

A pointer to the object, NULL if not found

10.11.4.6 BasePtr gazebo::physics::Base::GetByName (const std::string & *_name*)

Get by name.

Parameters

<i>in</i>	<i>_name</i>	Get a child (or self) object by name
-----------	--------------	--------------------------------------

Returns

A pointer to the object, NULL if not found

10.11.4.7 BasePtr gazebo::physics::Base::GetChild (unsigned int *_i*) const

Get a child by index.

Parameters

<i>in</i>	<i>_i</i>	Index of the child to retrieve.
-----------	-----------	---------------------------------

Returns

A pointer to the object, NULL if the index is invalid.

10.11.4.8 BasePtr gazebo::physics::Base::GetChild (const std::string & *_name*)

Get a child by name.

Parameters

<code>in</code>	<code>_name</code>	Name of the child.
-----------------	--------------------	--------------------

Returns

A pointer to the object, NULL if not found

10.11.4.9 unsigned int gazebo::physics::Base::GetChildCount () const

Get the number of children.

Returns

The number of children.

10.11.4.10 uint32_t gazebo::physics::Base::GetId () const

Return the ID of this entity.

This id is unique.

Returns

Integer ID.

10.11.4.11 std::string gazebo::physics::Base::GetName () const

Return the name of the entity.

Returns

Name of the entity.

10.11.4.12 BasePtr gazebo::physics::Base::GetParent () const

Get the parent.

Returns

Pointer to the parent entity.

Reimplemented in **gazebo::physics::Joint** (p. 557).

10.11.4.13 int gazebo::physics::Base::GetParentId () const

Return the ID of the parent.

Returns

Integer ID.

10.11.4.14 `bool gazebo::physics::Base::GetSaveable () const`

Get whether the object should be "saved", when the user selects to save the world to xml.

Returns

True if the object is saveable.

10.11.4.15 `std::string gazebo::physics::Base::GetScopedName () const`

Return the name of this entity with the model scope world::model1::...::modelN::entityName.

Returns

The scoped name.

10.11.4.16 `virtual const sdf::ElementPtr gazebo::physics::Base::GetSDF () [virtual]`

Get the SDF values for the object.

Returns

The SDF values for the object.

Reimplemented in `gazebo::physics::Actor` (p. 142), and `gazebo::physics::Model` (p. 686).

10.11.4.17 `unsigned int gazebo::physics::Base::GetType () const`

Get the full type definition.

Returns

The full type definition.

10.11.4.18 `const WorldPtr& gazebo::physics::Base::GetWorld () const`

Get the **World** (p. 1239) this object is in.

Returns

The **World** (p. 1239) this object is part of.

10.11.4.19 `bool gazebo::physics::Base::HasType (const EntityType & _t) const`

Returns true if this object's type definition has the given type.

Parameters

<code>in</code>	<code>_t</code>	Type to check.
-----------------	-----------------	----------------

Returns

True if this object's type definition has the.

10.11.4.20 `virtual void gazebo::physics::Base::Init() [inline],[virtual]`

Initialize the object.

Reimplemented in `gazebo::physics::RayShape` (p. 853), `gazebo::physics::Joint` (p. 559), `gazebo::physics::ScrewJoint< DARTJoint >` (p. 898), `gazebo::physics::ScrewJoint< SimbodyJoint >` (p. 898), `gazebo::physics::Actor` (p. 142), `gazebo::physics::Link` (p. 611), `gazebo::physics::UniversalJoint< DARTJoint >` (p. 1136), `gazebo::physics::UniversalJoint< SimbodyJoint >` (p. 1136), `gazebo::physics::BallJoint< DARTJoint >` (p. 168), `gazebo::physics::BallJoint< SimbodyJoint >` (p. 168), `gazebo::physics::Model` (p. 688), `gazebo::physics::HeightmapShape` (p. 510), `gazebo::physics::HingeJoint< DARTJoint >` (p. 514), `gazebo::physics::HingeJoint< SimbodyJoint >` (p. 514), `gazebo::physics::Collision` (p. 242), `gazebo::physics::MeshShape` (p. 677), `gazebo::physics::DARTLink` (p. 345), `gazebo::physics::MultiRayShape` (p. 730), `gazebo::physics::PlaneShape` (p. 793), `gazebo::physics::SimbodyLink` (p. 978), `gazebo::physics::DARTModel` (p. 352), `gazebo::physics::DARTScrewJoint` (p. 369), `gazebo::physics::Road` (p. 870), `gazebo::physics::Shape` (p. 935), `gazebo::physics::DARTJoint` (p. 334), `gazebo::physics::SphereShape` (p. 1056), `gazebo::physics::BoxShape` (p. 187), `gazebo::physics::CylinderShape` (p. 300), `gazebo::physics::DARTCollision` (p. 312), `gazebo::physics::DARTHinge2Joint` (p. 321), `gazebo::physics::DARTHingeJoint` (p. 326), `gazebo::physics::SimbodyHeightmapShape` (p. 950), `gazebo::physics::SimbodyModel` (p. 984), `gazebo::physics::SimbodyMeshShape` (p. 982), `gazebo::physics::DARTBallJoint` (p. 306), `gazebo::physics::DARTHeightmapShape` (p. 317), `gazebo::physics::DARTSliderJoint` (p. 374), `gazebo::physics::DARTUniversalJoint` (p. 381), and `gazebo::physics::DARTMeshShape` (p. 349).

10.11.4.21 `bool gazebo::physics::Base::IsSelected() const`

True if the entity is selected by the user.

Returns

True if the entity is selected.

10.11.4.22 `virtual void gazebo::physics::Base::Load(sdf::ElementPtr _sdf) [virtual]`

Load.

Parameters

<code>in</code>	<code>node</code>	Pointer to an SDF parameters
-----------------	-------------------	------------------------------

Reimplemented in `gazebo::physics::Joint` (p. 560), `gazebo::physics::Actor` (p. 142), `gazebo::physics::Link` (p. 612), `gazebo::physics::SimbodySliderJoint` (p. 1013), `gazebo::physics::Entity` (p. 411), `gazebo::physics::Model` (p. 688), `gazebo::physics::BallJoint< DARTJoint >` (p. 168), `gazebo::physics::BallJoint< SimbodyJoint >` (p. 168), `gazebo::physics::UniversalJoint< DARTJoint >` (p. 1136), `gazebo::physics::UniversalJoint< SimbodyJoint >` (p. 1136), `gazebo::physics::Hinge2Joint< DARTJoint >` (p. 513), `gazebo::physics::Hinge2Joint< SimbodyJoint >` (p. 513), `gazebo::physics::HeightmapShape` (p. 511), `gazebo::physics::Collision` (p. 243), `gazebo::physics::ScrewJoint< DARTJoint >` (p. 898), `gazebo::physics::ScrewJoint< SimbodyJoint >` (p. 898), `gazebo::physics::HingeJoint< DARTJoint >` (p. 514), `gazebo::physics::HingeJoint< SimbodyJoint >` (p. 514), `gazebo::physics::SliderJoint< DARTJoint >` (p. 1045), `gazebo::physics::SliderJoint< SimbodyJoint >` (p. 1045), `gazebo::physics::SimbodyCollision` (p. 945), `gazebo::physics::DARTLink` (p. 345), `gazebo::physics::`

SimbodyHingeJoint (p. 958), **gazebo::physics::SimbodyLink** (p. 978), **gazebo::physics::SimbodyUniversalJoint** (p. 1019), **gazebo::physics::DARTModel** (p. 352), **gazebo::physics::Road** (p. 870), **gazebo::physics::SimbodyHinge2Joint** (p. 954), **gazebo::physics::SimbodyScrewJoint** (p. 1007), **gazebo::physics::SimbodyJoint** (p. 966), **gazebo::physics::DARTJoint** (p. 334), **gazebo::physics::DARTCollision** (p. 312), **gazebo::physics::SimbodyBallJoint** (p. 940), **gazebo::physics::DARTHinge2Joint** (p. 321), **gazebo::physics::DARTHingeJoint** (p. 326), **gazebo::physics::SimbodyModel** (p. 984), **gazebo::physics::SimbodyMeshShape** (p. 982), **gazebo::physics::DARTBallJoint** (p. 306), **gazebo::physics::DARTScrewJoint** (p. 369), **gazebo::physics::DARTSliderJoint** (p. 374), **gazebo::physics::DARTUniversalJoint** (p. 381), and **gazebo::physics::DARTMeshShape** (p. 349).

10.11.4.23 `bool gazebo::physics::Base::operator==(const Base & _ent) const`

Returns true if the entities are the same.

Checks only the name.

Parameters

<code>in</code>	<code>_ent</code>	Base (p. 168) object to compare with.
-----------------	-------------------	--

Returns

True if the entities are the same.

10.11.4.24 `void gazebo::physics::Base::Print (const std::string & _prefix)`

Print this object to screen via gzmsg.

Parameters

<code>in</code>	<code>_prefix</code>	Usually a set of spaces.
-----------------	----------------------	--------------------------

10.11.4.25 `virtual void gazebo::physics::Base::RemoveChild (unsigned int _id) [virtual]`

Remove a child from this entity.

Parameters

<code>in</code>	<code>_id</code>	ID of the child to remove.
-----------------	------------------	----------------------------

10.11.4.26 `void gazebo::physics::Base::RemoveChild (const std::string & _name)`

Remove a child by name.

Parameters

<code>in</code>	<code>_name</code>	Name of the child.
-----------------	--------------------	--------------------

10.11.4.27 void gazebo::physics::Base::RemoveChildren ()

Remove all children.

10.11.4.28 virtual void gazebo::physics::Base::Reset () [virtual]

Reset the object.

Reimplemented in **gazebo::physics::Joint** (p. 560), **gazebo::physics::Model** (p. 689), **gazebo::physics::Link** (p. 613), **gazebo::physics::Entity** (p. 412), **gazebo::physics::DARTJoint** (p. 335), and **gazebo::physics::SimbodyJoint** (p. 967).

10.11.4.29 virtual void gazebo::physics::Base::Reset (Base::EntityType _resetType) [virtual]

Calls recursive Reset on one of the **Base::EntityType** (p. 172)'s.

Parameters

in	<code>_resetType</code>	The type of reset operation
----	-------------------------	-----------------------------

10.11.4.30 virtual void gazebo::physics::Base::SetName (const std::string & _name) [virtual]

Set the name of the entity.

Parameters

in	<code>_name</code>	New name.
----	--------------------	-----------

Reimplemented in **gazebo::physics::Entity** (p. 413).

10.11.4.31 void gazebo::physics::Base::SetParent (BasePtr _parent)

Set the parent.

Parameters

in	<code>_parent</code>	Parent object.
----	----------------------	----------------

10.11.4.32 void gazebo::physics::Base::SetSaveable (bool _v)

Set whether the object should be "saved", when the user selects to save the world to xml.

Parameters

in	<code>_v</code>	Set to True if the object should be saved.
----	-----------------	--

10.11.4.33 virtual bool gazebo::physics::Base::SetSelected (bool _show) [virtual]

Set whether this entity has been selected by the user through the gui.

Parameters

in	<code>_show</code>	True to set this entity as selected.
----	--------------------	--------------------------------------

Reimplemented in `gazebo::physics::Link` (p. 616).

10.11.4.34 `void gazebo::physics::Base::SetWorld (const WorldPtr & _newWorld)`

Set the world this object belongs to.

This will also set the world for all children.

Parameters

in	<code>_newWorld</code>	The new World (p. 1239) this object is part of.
----	------------------------	--

10.11.4.35 `virtual void gazebo::physics::Base::Update () [inline],[virtual]`

Update the object.

Reimplemented in `gazebo::physics::MultiRayShape` (p. 730), `gazebo::physics::Joint` (p. 566), `gazebo::physics::Actor` (p. 143), `gazebo::physics::RayShape` (p. 854), `gazebo::physics::Model` (p. 692), `gazebo::physics::DARTModel` (p. 352), `gazebo::physics::DARTRayShape` (p. 78), `gazebo::physics::MeshShape` (p. 678), `gazebo::physics::SimbodyRayShape` (p. 1002), and `gazebo::physics::DARTMeshShape` (p. 350).

10.11.4.36 `virtual void gazebo::physics::Base::UpdateParameters (sdf::ElementPtr _sdf) [virtual]`

Update the parameters using new sdf values.

Parameters

in	<code>_sdf</code>	Update the object's parameters based on SDF values.
----	-------------------	---

Reimplemented in `gazebo::physics::Joint` (p. 566), `gazebo::physics::Actor` (p. 143), `gazebo::physics::Link` (p. 617), `gazebo::physics::Model` (p. 692), `gazebo::physics::Entity` (p. 414), and `gazebo::physics::Collision` (p. 245).

10.11.5 Member Data Documentation

10.11.5.1 `Base_V gazebo::physics::Base::children [protected]`

Children of this entity.

10.11.5.2 `BasePtr gazebo::physics::Base::parent [protected]`

Parent of this entity.

10.11.5.3 `sdf::ElementPtr gazebo::physics::Base::sdf [protected]`

The SDF values for this object.

10.11.5.4 WorldPtr gazebo::physics::Base::world [protected]

Pointer to the world.

The documentation for this class was generated from the following file:

- **Base.hh**

10.12 gazebo::math::Box Class Reference

Mathematical representation of a box and related functions.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Box** ()
Default constructor.
- **Box** (const **Vector3** &_min, const **Vector3** &_max)
Constructor.
- **Box** (const **Box** &_b)
Copy Constructor.
- virtual ~**Box** ()
Destructor.
- **math::Vector3 GetCenter** () const
Get the box center.
- **math::Vector3 GetSize** () const
Get the size of the box.
- double **GetXLength** () const
Get the length along the x dimension.
- double **GetYLength** () const
Get the length along the y dimension.
- double **GetZLength** () const
Get the length along the z dimension.
- void **Merge** (const **Box** &_box)
Merge a box with this box.
- **Box operator+** (const **Box** &_b) const
Addition operator.
- const **Box** & **operator+=** (const **Box** &_b)
Addition set operator.
- **Box operator-** (const **Vector3** &_v)
Subtract a vector from the min and max values.
- **Box** & **operator=** (const **Box** &_b)
Assignment operator.
- bool **operator==** (const **Box** &_b) const
Equality test operator.

Public Attributes

- **Vector3 max**
Maximum corner of the box.
- **Vector3 min**
Minimum corner of the box.

Friends

- `std::ostream & operator<< (std::ostream &_out, const gazebo::math::Box &_b)`
Output operator.

10.12.1 Detailed Description

Mathematical representation of a box and related functions.

10.12.2 Constructor & Destructor Documentation

10.12.2.1 gazebo::math::Box::Box ()

Default constructor.

10.12.2.2 gazebo::math::Box::Box (const Vector3 & _min, const Vector3 & _max)

Constructor.

Parameters

<code>in</code>	<code>_min</code>	Minimum corner of the box
<code>in</code>	<code>_max</code>	Maximum corner of the box

10.12.2.3 gazebo::math::Box::Box (const Box & _b)

Copy Constructor.

Parameters

<code>in</code>	<code>_b</code>	Box (p. 181) to copy
-----------------	-----------------	-----------------------------

10.12.2.4 virtual gazebo::math::Box::~~Box () [virtual]

Destructor.

10.12.3 Member Function Documentation

10.12.3.1 `math::Vector3 gazebo::math::Box::GetCenter () const`

Get the box center.

Returns

The center position of the box

10.12.3.2 `math::Vector3 gazebo::math::Box::GetSize () const`

Get the size of the box.

Returns

Size of the box

10.12.3.3 `double gazebo::math::Box::GetXLength () const`

Get the length along the x dimension.

Returns

Double value of the length in the x dimension

10.12.3.4 `double gazebo::math::Box::GetYLength () const`

Get the length along the y dimension.

Returns

Double value of the length in the y dimension

10.12.3.5 `double gazebo::math::Box::GetZLength () const`

Get the length along the z dimension.

Returns

Double value of the length in the z dimension

10.12.3.6 `void gazebo::math::Box::Merge (const Box & _box)`

Merge a box with this box.

Parameters

<code>in</code>	<code>_box</code>	Box (p. 181) to add to this box
-----------------	-------------------	--

10.12.3.7 `Box gazebo::math::Box::operator+ (const Box & _b) const`

Addition operator.

result = this + _b

Parameters

<code>in</code>	<code>_b</code>	Box (p. 181) to add
-----------------	-----------------	----------------------------

Returns

The new box

10.12.3.8 `const Box& gazebo::math::Box::operator+=(const Box & _b)`

Addition set operator.

this = this + _b

Parameters

<code>in</code>	<code>_b</code>	Box (p. 181) to add
-----------------	-----------------	----------------------------

Returns

This new box

10.12.3.9 `Box gazebo::math::Box::operator- (const Vector3 & _v)`

Subtract a vector from the min and max values.

Parameters

	<code>_v</code>	The vector to use during subtraction
--	-----------------	--------------------------------------

Returns

The new box

10.12.3.10 `Box& gazebo::math::Box::operator= (const Box & _b)`

Assignment operator.

Set this box to the parameter

Parameters

<code>in</code>	<code>_b</code>	Box (p. 181) to copy
-----------------	-----------------	-----------------------------

Returns

The new box.

10.12.3.11 `bool gazebo::math::Box::operator==(const Box & _b) const`

Equality test operator.

Parameters

<code>in</code>	<code>_b</code>	Box (p. 181) to test
-----------------	-----------------	-----------------------------

Returns

True if equal

10.12.4 Friends And Related Function Documentation

10.12.4.1 `std::ostream& operator<<(std::ostream & _out, const gazebo::math::Box & _b)` [friend]

Output operator.

Parameters

<code>in</code>	<code>_out</code>	Output stream
<code>in</code>	<code>_b</code>	Box (p. 181) to output to the stream

Returns

The stream

10.12.5 Member Data Documentation

10.12.5.1 `Vector3 gazebo::math::Box::max`

Maximum corner of the box.

10.12.5.2 `Vector3 gazebo::math::Box::min`

Minimum corner of the box.

The documentation for this class was generated from the following file:

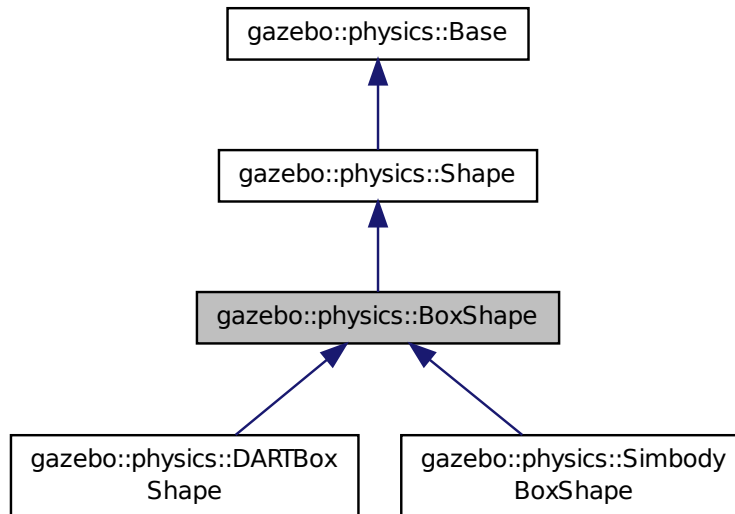
- **Box.hh**

10.13 gazebo::physics::BoxShape Class Reference

Box geometry primitive.

```
#include <physics/physcs.hh>
```

Inheritance diagram for gazebo::physics::BoxShape:



Public Member Functions

- **BoxShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~BoxShape** ()
Destructor.
- void **FillMsg** (msgs::Geometry &_msg)
Fill in the values for a geometry message.
- **math::Vector3** **GetSize** () const
Get the size of the box.
- virtual void **Init** ()
Initialize the box.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
Process a geometry message.
- virtual void **SetScale** (const **math::Vector3** &_scale)
Set the scale of the box.
- virtual void **SetSize** (const **math::Vector3** &_size)
Set the size of the box.

Additional Inherited Members

10.13.1 Detailed Description

Box geometry primitive.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 gazebo::physics::BoxShape::BoxShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent Collision (p. 235).
----	----------------	-----------------------------------

10.13.2.2 virtual gazebo::physics::BoxShape::~~BoxShape () [virtual]

Destructor.

10.13.3 Member Function Documentation

10.13.3.1 void gazebo::physics::BoxShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Fill in the values for a geometry message.

Parameters

out	<i>_msg</i>	The geometry message to fill.
-----	-------------	-------------------------------

Implements **gazebo::physics::Shape** (p. 934).

10.13.3.2 math::Vector3 gazebo::physics::BoxShape::GetSize () const

Get the size of the box.

Returns

The size of each side of the box.

10.13.3.3 virtual void gazebo::physics::BoxShape::Init () [virtual]

Initialize the box.

Implements **gazebo::physics::Shape** (p. 935).

10.13.3.4 virtual void gazebo::physics::BoxShape::ProcessMsg (const msgs::Geometry & *_msg*) [virtual]

Process a geometry message.

Parameters

in	<i>_msg</i>	The message to set values from.
----	-------------	---------------------------------

Implements **gazebo::physics::Shape** (p. 935).

10.13.3.5 `virtual void gazebo::physics::BoxShape::SetScale (const math::Vector3 & _scale) [virtual]`

Set the scale of the box.

Parameters

<code>in</code>	<code>_scale</code>	Scale of the box.
-----------------	---------------------	-------------------

Implements `gazebo::physics::Shape` (p. 935).

10.13.3.6 `virtual void gazebo::physics::BoxShape::SetSize (const math::Vector3 & _size) [virtual]`

Set the size of the box.

Parameters

<code>in</code>	<code>_size</code>	Size of each side of the box.
-----------------	--------------------	-------------------------------

Reimplemented in `gazebo::physics::DARTBoxShape` (p. 309), and `gazebo::physics::SimbodyBoxShape` (p. 943).

Referenced by `gazebo::physics::SimbodyBoxShape::SetSize()`, and `gazebo::physics::DARTBoxShape::SetSize()`.

The documentation for this class was generated from the following file:

- **BoxShape.hh**

10.14 gazebo::common::Logger::Buffer Class Reference

String buffer for the base logger.

```
#include <Console.hh>
```

Public Member Functions

- **Buffer (LogType _type, int _color)**
Constructor.
- `virtual ~Buffer ()`
Destructor.
- `virtual int sync ()`
Sync the stream (output the string buffer contents).

Public Attributes

- `int color`
ANSI color code using Select Graphic Rendition parameters (SGR).
- **LogType type**
Destination type for the messages.

10.14.1 Detailed Description

String buffer for the base logger.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 gazebo::common::Logger::Buffer::Buffer (LogType *_type*, int *_color*)

Constructor.

Parameters

in	<i>_type</i>	Output destination type (STDOUT, or STDERR)
in	<i>_color</i>	Color (p. 249) of the output stream.

10.14.2.2 virtual gazebo::common::Logger::Buffer::~Buffer () [virtual]

Destructor.

10.14.3 Member Function Documentation

10.14.3.1 virtual int gazebo::common::Logger::Buffer::sync () [virtual]

Sync the stream (output the string buffer contents).

Returns

Return 0 on success.

10.14.4 Member Data Documentation

10.14.4.1 int gazebo::common::Logger::Buffer::color

ANSI color code using Select Graphic Rendition parameters (SGR).

See http://en.wikipedia.org/wiki/ANSI_escape_code#Colors

10.14.4.2 LogType gazebo::common::Logger::Buffer::type

Destination type for the messages.

The documentation for this class was generated from the following file:

- **Console.hh**

10.15 gazebo::common::FileLogger::Buffer Class Reference

String buffer for the file logger.

```
#include <Console.hh>
```

Public Member Functions

- **Buffer** (const std::string &*_filename*)

Constructor.

- virtual `~Buffer ()`

Destructor.

- virtual int `sync ()`

Sync the stream (output the string buffer contents).

Public Attributes

- `std::ofstream * stream`

Stream to output information into.

10.15.1 Detailed Description

String buffer for the file logger.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 gazebo::common::FileLogger::Buffer::Buffer (const std::string & *_filename*)

Constructor.

Parameters

<code>in</code>	<code>_filename</code>	Filename to write into.
-----------------	------------------------	-------------------------

10.15.2.2 virtual gazebo::common::FileLogger::Buffer::~Buffer () [virtual]

Destructor.

10.15.3 Member Function Documentation

10.15.3.1 virtual int gazebo::common::FileLogger::Buffer::sync () [virtual]

Sync the stream (output the string buffer contents).

Returns

Return 0 on success.

10.15.4 Member Data Documentation

10.15.4.1 std::ofstream* gazebo::common::FileLogger::Buffer::stream

Stream to output information into.

The documentation for this class was generated from the following file:

- **Console.hh**

10.16 gazebo::common::BVHLoader Class Reference

Handles loading BVH animation files.

```
#include <common/common.hh>
```

Public Member Functions

- **BVHLoader** ()
Constructor.
- **~BVHLoader** ()
Desutrcctor.
- **Skeleton * Load** (const std::string &_filename, double _scale)
Load a BVH file.

10.16.1 Detailed Description

Handles loading BVH animation files.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 gazebo::common::BVHLoader::BVHLoader ()

Constructor.

10.16.2.2 gazebo::common::BVHLoader::~~BVHLoader ()

Desutrcctor.

10.16.3 Member Function Documentation

10.16.3.1 Skeleton* gazebo::common::BVHLoader::Load (const std::string &_filename, double _scale)

Load a BVH file.

Parameters

in	_filename	BVH file to load
in	_scale	Scaling factor to apply to the skeleton

Returns

A pointer to a new **Skeleton** (p. 1024)

The documentation for this class was generated from the following file:

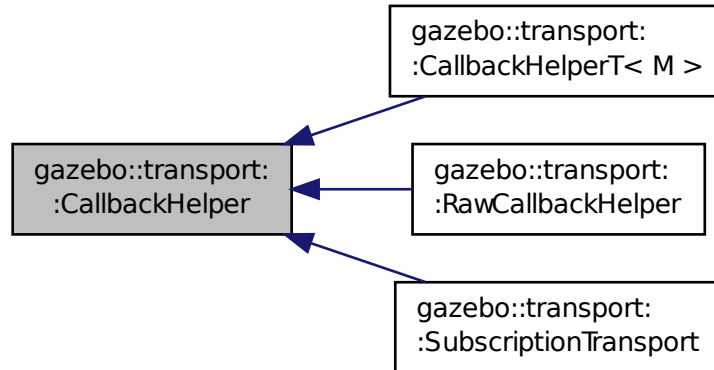
- **BVHLoader.hh**

10.17 gazebo::transport::CallbackHelper Class Reference

A helper class to handle callbacks when messages arrive.

```
#include <transport/transport.hh>
```

Inheritance diagram for gazebo::transport::CallbackHelper:



Public Member Functions

- **CallbackHelper** (bool *_latching*=false)
Constructor.
- virtual **~CallbackHelper** ()
Destructor.
- unsigned int **GetId** () const
Get the unique ID of this callback.
- bool **GetLatching** () const
Is the callback latching?
- virtual std::string **GetMsgType** () const
Get the typename of the message that is handled.
- virtual bool **HandleData** (const std::string &*_newdata*, boost::function< void(uint32_t)> *_cb*, uint32_t *_id*)=0
Process new incoming data.
- virtual bool **HandleMessage** (**MessagePtr** *_newMsg*)=0
Process new incoming message.
- virtual bool **IsLocal** () const =0
Is the callback local?
- void **SetLatching** (bool *_latch*)
Set whether this callback is latching.

Protected Attributes

- bool **latching**

True means that the callback helper will get the last published message on the topic.

10.17.1 Detailed Description

A helper class to handle callbacks when messages arrive.

10.17.2 Constructor & Destructor Documentation

10.17.2.1 gazebo::transport::CallbackHelper::CallbackHelper (bool *latching* = false)

Constructor.

Parameters

<i>in</i>	<i>_latching</i>	Set to true to make the callback helper latching.
-----------	------------------	---

10.17.2.2 virtual gazebo::transport::CallbackHelper::~~CallbackHelper () [virtual]

Destructor.

10.17.3 Member Function Documentation

10.17.3.1 unsigned int gazebo::transport::CallbackHelper::GetId () const

Get the unique ID of this callback.

Returns

The unique ID of this callback.

10.17.3.2 bool gazebo::transport::CallbackHelper::GetLatching () const

Is the callback latching?

Returns

true if the callback is latching, false otherwise

10.17.3.3 virtual std::string gazebo::transport::CallbackHelper::GetMsgType () const [virtual]

Get the typename of the message that is handled.

Returns

String representation of the message type

Reimplemented in [gazebo::transport::RawCallbackHelper](#) (p. 841), and [gazebo::transport::CallbackHelperT< M >](#) (p. 196).

10.17.3.4 `virtual bool gazebo::transport::CallbackHelper::HandleData (const std::string & _newdata, boost::function< void(uint32_t)> _cb, uint32_t _id) [pure virtual]`

Process new incoming data.

Parameters

<code>in</code>	<code>_newdata</code>	Incoming data to be processed
-----------------	-----------------------	-------------------------------

Returns

true if successfully processed; false otherwise

Parameters

<code>in</code>	<code>_cb</code>	If non-null, callback to be invoked which signals that transmission is complete.
<code>in</code>	<code>_id</code>	ID associated with the message data.

Implemented in `gazebo::transport::RawCallbackHelper` (p. 841), `gazebo::transport::CallbackHelperT< M >` (p. 196), and `gazebo::transport::SubscriptionTransport` (p. 1089).

10.17.3.5 `virtual bool gazebo::transport::CallbackHelper::HandleMessage (MessagePtr _newMsg) [pure virtual]`

Process new incoming message.

Parameters

<code>in</code>	<code>_newMsg</code>	Incoming message to be processed
-----------------	----------------------	----------------------------------

Returns

true if successfully processed; false otherwise

Implemented in `gazebo::transport::RawCallbackHelper` (p. 841), `gazebo::transport::CallbackHelperT< M >` (p. 197), and `gazebo::transport::SubscriptionTransport` (p. 1089).

10.17.3.6 `virtual bool gazebo::transport::CallbackHelper::IsLocal () const [pure virtual]`

Is the callback local?

Returns

true if the callback is local, false if the callback is tied to a remote connection

Implemented in `gazebo::transport::RawCallbackHelper` (p. 842), `gazebo::transport::CallbackHelperT< M >` (p. 197), and `gazebo::transport::SubscriptionTransport` (p. 1090).

10.17.3.7 `void gazebo::transport::CallbackHelper::SetLatching (bool _latch)`

Set whether this callback is latching.

This function should only be used by the Transport library.

Parameters

in	<code>_latch</code>	False to turn off latching.
----	---------------------	-----------------------------

10.17.4 Member Data Documentation

10.17.4.1 `bool gazebo::transport::CallbackHelper::latching` [protected]

True means that the callback helper will get the last published message on the topic.

The documentation for this class was generated from the following file:

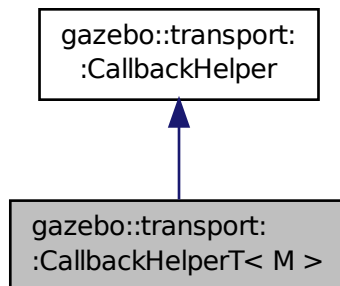
- **CallbackHelper.hh**

10.18 gazebo::transport::CallbackHelperT< M > Class Template Reference

Callback helper Template.

```
#include <transport/transport.hh>
```

Inheritance diagram for gazebo::transport::CallbackHelperT< M >:



Public Member Functions

- **CallbackHelperT** (const boost::function< void(const boost::shared_ptr< M const > &)> &_cb, bool _latching=false)
Constructor.
- std::string **GetMsgType** () const
Get the typename of the message that is handled.
- virtual bool **HandleData** (const std::string &_newdata, boost::function< void(uint32_t)> _cb, uint32_t _id)
Process new incoming data.
- virtual bool **HandleMessage** (MessagePtr _newMsg)
Process new incoming message.
- virtual bool **IsLocal** () const
Is the callback local?

Additional Inherited Members

10.18.1 Detailed Description

```
template<class M>class gazebo::transport::CallbackHelperT< M >
```

Callback helper Template.

10.18.2 Constructor & Destructor Documentation

```
10.18.2.1 template<class M > gazebo::transport::CallbackHelperT< M >::CallbackHelperT ( const boost::function<
void(const boost::shared_ptr< M const > &> & _cb, bool _latching = false ) [inline]
```

Constructor.

Parameters

in	<code>_cb</code>	boost function to call on incoming messages
in	<code>_latching</code>	Set to true to make the callback helper latching.

10.18.3 Member Function Documentation

```
10.18.3.1 template<class M > std::string gazebo::transport::CallbackHelperT< M >::GetMsgType ( ) const
[inline],[virtual]
```

Get the typename of the message that is handled.

Returns

String representation of the message type

Reimplemented from `gazebo::transport::CallbackHelper` (p. 193).

References `gzthrow`, and `NULL`.

```
10.18.3.2 template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::HandleData ( const std::string &
_newdata, boost::function< void(uint32_t)> _cb, uint32_t _id ) [inline],[virtual]
```

Process new incoming data.

Parameters

in	<code>_newdata</code>	Incoming data to be processed
----	-----------------------	-------------------------------

Returns

true if successfully processed; false otherwise

Parameters

in	<code>_cb</code>	If non-null, callback to be invoked which signals that transmission is complete.
in	<code>_id</code>	ID associated with the message data.

Implements **gazebo::transport::CallbackHelper** (p. 194).

10.18.3.3 `template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::HandleMessage (MessagePtr _newMsg) [inline],[virtual]`

Process new incoming message.

Parameters

<code>in</code>	<code>_newMsg</code>	Incoming message to be processed
-----------------	----------------------	----------------------------------

Returns

true if successfully processed; false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 194).

10.18.3.4 `template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::IsLocal () const [inline],[virtual]`

Is the callback local?

Returns

true if the callback is local, false if the callback is tied to a remote connection

Implements **gazebo::transport::CallbackHelper** (p. 194).

The documentation for this class was generated from the following file:

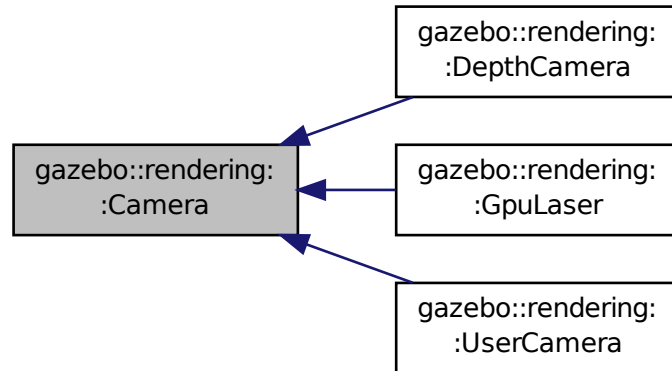
- **CallbackHelper.hh**

10.19 gazebo::rendering::Camera Class Reference

Basic camera sensor.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::Camera:



Public Member Functions

- **Camera** (const std::string &_namePrefix, **ScenePtr** _scene, bool _autoRender=true)
Constructor.
- virtual ~**Camera** ()
Destructor.
- void **AttachToVisual** (const std::string &_visualName, bool _inheritOrientation, double _minDist=0.0, double _maxDist=0.0)
Attach the camera to a scene node.
- void **AttachToVisual** (uint32_t _id, bool _inheritOrientation, double _minDist=0.0, double _maxDist=0.0)
Attach the camera to a scene node.
- template<typename T >
event::ConnectionPtr ConnectNewImageFrame (T _subscriber)
Connect to the new image signal.
- void **CreateRenderTexture** (const std::string &_textureName)
Set the render target.
- void **DisconnectNewImageFrame** (**event::ConnectionPtr** &_c)
Disconnect from an image frame.
- void **EnableSaveFrame** (bool _enable)
Enable or disable saving.
- virtual void **Fini** ()
Finalize the camera.
- float **GetAspectRatio** () const
Get the aspect ratio.
- virtual float **GetAvgFPS** () const
Get the average FPS.
- void **GetCameraToViewportRay** (int _screenx, int _screeny, **math::Vector3** &_origin, **math::Vector3** &_dir)
Get a world space ray as cast from the camera through the viewport.

- bool **GetCaptureData** () const
Return the value of this->captureData.
- **math::Vector3 GetDirection** () const
Get the camera's direction vector.
- double **GetFarClip** ()
Get the far clip distance.
- **math::Angle GetHFOV** () const
Get the camera FOV (horizontal)
- size_t **GetImageByteSize** () const
Get the image size in bytes.
- virtual const unsigned char * **GetImageData** (unsigned int i=0)
Get a pointer to the image data.
- unsigned int **GetImageDepth** () const
Get the depth of the image.
- std::string **GetImageFormat** () const
Get the string representation of the image format.
- virtual unsigned int **GetImageHeight** () const
Get the height of the image.
- virtual unsigned int **GetImageWidth** () const
Get the width of the image.
- bool **GetInitialized** () const
Return true if the camera has been initialized.
- **common::Time GetLastRenderWallTime** ()
Get the last time the camera was rendered.
- std::string **GetName** () const
Get the camera's unscoped name.
- double **GetNearClip** ()
Get the near clip distance.
- Ogre::Camera * **GetOgreCamera** () const
Get a pointer to the ogre camera.
- Ogre::SceneNode * **GetPitchNode** () const **GAZEBO_DEPRECATED(3.0)**
Deprecated: Get the camera's pitch scene node.
- double **GetRenderRate** () const
Get the render Hz rate.
- Ogre::Texture * **GetRenderTexture** () const
Get the render texture.
- **math::Vector3 GetRight** ()
Get the viewport right vector.
- **ScenePtr GetScene** () const
Get the scene this camera is in.
- Ogre::SceneNode * **GetSceneNode** () const
Get the camera's scene node.
- std::string **GetScopedName** () const
Get the camera's scoped name (scene_name::camera_name)
- std::string **GetScreenshotPath** () const
Get the path to saved screenshots.
- unsigned int **GetTextureHeight** () const

- Get the height of the off-screen render texture.*

 - unsigned int **GetTextureWidth** () const
- Get the width of the off-screen render texture.*

 - virtual unsigned int **GetTriangleCount** () const
- Get the triangle count.*

 - **math::Vector3** **GetUp** ()
- Get the viewport up vector.*

 - **math::Angle** **GetVFOV** () const
- Get the camera FOV (vertical)*

 - Ogre::Viewport * **GetViewport** () const
- Get a pointer to the Ogre::Viewport.*

 - unsigned int **GetViewportHeight** () const
- Get the viewport height in pixels.*

 - unsigned int **GetViewportWidth** () const
- Get the viewport width in pixels.*

 - unsigned int **GetWindowId** () const
- Get the ID of the window this camera is rendering into.*

 - bool **GetWorldPointOnPlane** (int _x, int _y, const **math::Plane** &_plane, **math::Vector3** &_result)
- Get point on a plane.*

 - **math::Pose** **GetWorldPose** () const
- Get the world pose.*

 - **math::Vector3** **GetWorldPosition** () const
- Get the camera position in the world.*

 - **math::Quaternion** **GetWorldRotation** () const
- Get the camera's orientation in the world.*

 - double **GetZValue** (int _x, int _y)
- Get the Z-buffer value at the given image coordinate.*

 - virtual void **Init** ()
- Initialize the camera.*

 - bool **IsAnimating** () const
- Return true if the camera is moving due to an animation.*

 - bool **IsVisible** (**VisualPtr** _visual)
- Return true if the visual is within the camera's view frustum.*

 - bool **IsVisible** (const std::string &_visualName)
- Return true if the visual is within the camera's view frustum.*

 - virtual void **Load** (sdf::ElementPtr _sdf)
- Load the camera with a set of parameters.*

 - virtual void **Load** ()
- Load the camera with default parameters.*

 - virtual bool **MoveToPosition** (const **math::Pose** &_pose, double _time)
- Move the camera to a position (this is an animated motion).*

 - bool **MoveToPositions** (const std::vector< **math::Pose** > &_pts, double _time, boost::function< void()> _on-Complete=NULL)
- Move the camera to a series of poses (this is an animated motion).*

 - virtual void **PostRender** ()
- Post render.*

 - void **Render** ()

- Render the camera.*

 - void **Render** (bool _force)
- Render the camera.*

 - void **RotatePitch** (math::Angle _angle)

Rotate the camera around the pitch axis.
- void **RotateYaw** (math::Angle _angle)

Rotate the camera around the yaw axis.
- bool **SaveFrame** (const std::string &_filename)

Save the last frame to disk.
- void **SetAspectRatio** (float _ratio)

Set the aspect ratio.
- void **SetCaptureData** (bool _value)

Set whether to capture data.
- void **SetCaptureDataOnce** ()

Capture data once and save to disk.
- void **SetClipDist** (float _near, float _far)

Set the clip distances.
- void **SetHFOV** (math::Angle _angle)

Set the camera FOV (horizontal)
- void **SetImageHeight** (unsigned int _h)

Set the image height.
- void **SetImageSize** (unsigned int _w, unsigned int _h)

Set the image size.
- void **SetImageWidth** (unsigned int _w)

Set the image height.
- void **SetName** (const std::string &_name)

Set the camera's name.
- void **SetRenderRate** (double _hz)

Set the render Hz rate.
- virtual void **SetRenderTarget** (Ogre::RenderTarget *_target)

Set the camera's render target.
- void **SetSaveFramePathname** (const std::string &_pathname)

Set the save frame pathname.
- void **SetScene** (ScenePtr _scene)

Set the scene this camera is viewing.
- void **SetSceneNode** (Ogre::SceneNode *_node)

Set the camera's scene node.
- void **SetWindowId** (unsigned int _windowId)
- virtual void **SetWorldPose** (const math::Pose &_pose)

Set the global pose of the camera.
- void **SetWorldPosition** (const math::Vector3 &_pos)

Set the world position.
- void **SetWorldRotation** (const math::Quaternion &_quat)

Set the world orientation.
- void **ShowWireframe** (bool _s)

Set whether to view the world in wireframe.
- void **ToggleShowWireframe** ()

- *Toggle whether to view the world in wireframe.*
- void **TrackVisual** (const std::string &_visualName)
Set the camera to track a scene node.
- void **Translate** (const **math::Vector3** &_direction)
Translate the camera.
- virtual void **Update** ()

Static Public Member Functions

- static size_t **GetImageByteSize** (unsigned int _width, unsigned int _height, const std::string &_format)
Calculate image byte size base on a few parameters.
- static bool **SaveFrame** (const unsigned char *_image, unsigned int _width, unsigned int _height, int _depth, const std::string &_format, const std::string &_filename)
Save a frame using an image buffer.

Protected Member Functions

- virtual void **AnimationComplete** ()
Internal function used to indicate that an animation has completed.
- virtual bool **AttachToVisualImpl** (const std::string &_name, bool _inheritOrientation, double _minDist=0, double _maxDist=0)
Attach the camera to a scene node.
- virtual bool **AttachToVisualImpl** (uint32_t _id, bool _inheritOrientation, double _minDist=0, double _maxDist=0)
Attach the camera to a scene node.
- virtual bool **AttachToVisualImpl** (**VisualPtr** _visual, bool _inheritOrientation, double _minDist=0, double _maxDist=0)
Attach the camera to a visual.
- std::string **GetFrameFilename** ()
Get the next frame filename based on SDF parameters.
- void **ReadPixelBuffer** ()
Read image data from pixel buffer.
- virtual void **RenderImpl** ()
Implementation of the render call.
- bool **TrackVisualImpl** (const std::string &_visualName)
*Implementation of the **Camera::TrackVisual** (p. 221) call.*
- virtual bool **TrackVisualImpl** (**VisualPtr** _visual)
Set the camera to track a scene node.

Protected Attributes

- Ogre::AnimationState * **animState**
Animation state, used to animate the camera.
- unsigned char * **bayerFrameBuffer**
Buffer for a bayer image frame.
- Ogre::Camera * **camera**
The OGRE camera.
- bool **captureData**

- True to capture frames into an image buffer.*
- **bool captureDataOnce**

True to capture a frame once and save to disk.
- **std::vector< event::ConnectionPtr > connections**

The camera's event connections.
- **int imageFormat**

Format for saving images.
- **int imageHeight**

Save image height.
- **int imageWidth**

Save image width.
- **bool initialized**

True if initialized.
- **common::Time lastRenderWallTime**

Time the last frame was rendered.
- **std::string name**

Name of the camera.
- **bool newData**

True if new data is available.
- **event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)> newImageFrame**

Event triggered when a new frame is generated.
- **boost::function< void()> onAnimationComplete**

User callback for when an animation completes.
- **common::Time prevAnimTime**

Previous time the camera animation was updated.
- **Ogre::RenderTarget * renderTarget**

Target that renders frames.
- **Ogre::Texture * renderTexture**

Texture that receives results from rendering.
- **std::list< msgs::Request > requests**

List of requests.
- **unsigned int saveCount**

Number of saved frames.
- **unsigned char * saveFrameBuffer**
- **ScenePtr scene**

Pointer to the scene.
- **Ogre::SceneNode * sceneNode**

Scene (p. 879) node that controls camera position and orientation.
- **std::string scopedName**

Scene (p. 879) scoped name of the camera.
- **std::string scopedUniqueName**

Scene (p. 879) scoped name of the camera with a unique ID.
- **std::string screenshotPath**

Path to saved screenshots.
- **sdf::ElementPtr sdf**

Camera (p. 197)'s SDF values.

- unsigned int **textureHeight**
Height of the render texture.
- unsigned int **textureWidth**
Width of the render texture.
- Ogre::Viewport * **viewport**
Viewport the ogre camera uses.
- unsigned int **windowId**
ID of the window that the camera is attached to.

10.19.1 Detailed Description

Basic camera sensor.

This is the base class for all cameras.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 gazebo::rendering::Camera::Camera (const std::string & *_namePrefix*, ScenePtr *_scene*, bool *_autoRender* = true)

Constructor.

Parameters

in	<i>_namePrefix</i>	Unique prefix name for the camera.
in	<i>_scene</i>	Scene (p. 879) that will contain the camera
in	<i>_autoRender</i>	Almost everyone should leave this as true.

10.19.2.2 virtual gazebo::rendering::Camera::~~Camera () [virtual]

Destructor.

10.19.3 Member Function Documentation

10.19.3.1 virtual void gazebo::rendering::Camera::AnimationComplete () [protected],[virtual]

Internal function used to indicate that an animation has completed.

Reimplemented in **gazebo::rendering::UserCamera** (p. 1140).

10.19.3.2 void gazebo::rendering::Camera::AttachToVisual (const std::string & *_visualName*, bool *_inheritOrientation*, double *_minDist* = 0.0, double *_maxDist* = 0.0)

Attach the camera to a scene node.

Parameters

in	<i>_visualName</i>	Name of the visual to attach the camera to
in	<i>_inheritOrientation</i>	True means camera acquires the visual's orientation

in	<i>_minDist</i>	Minimum distance the camera is allowed to get to the visual
in	<i>_maxDist</i>	Maximum distance the camera is allowed to get from the visual

10.19.3.3 void gazebo::rendering::Camera::AttachToVisual (uint32_t *_id*, bool *_inheritOrientation*, double *_minDist* = 0.0, double *_maxDist* = 0.0)

Attach the camera to a scene node.

Parameters

in	<i>_id</i>	ID of the visual to attach the camera to
in	<i>_inheritOrientation</i>	True means camera acquires the visual's orientation
in	<i>_minDist</i>	Minimum distance the camera is allowed to get to the visual
in	<i>_maxDist</i>	Maximum distance the camera is allowed to get from the visual

10.19.3.4 virtual bool gazebo::rendering::Camera::AttachToVisualImpl (const std::string & *_name*, bool *_inheritOrientation*, double *_minDist* = 0, double *_maxDist* = 0) [protected],[virtual]

Attach the camera to a scene node.

Parameters

in	<i>_visualName</i>	Name of the visual to attach the camera to
in	<i>_inheritOrientation</i>	True means camera acquires the visual's orientation
in	<i>_minDist</i>	Minimum distance the camera is allowed to get to the visual
in	<i>_maxDist</i>	Maximum distance the camera is allowed to get from the visual

Returns

True on success

10.19.3.5 virtual bool gazebo::rendering::Camera::AttachToVisualImpl (uint32_t *_id*, bool *_inheritOrientation*, double *_minDist* = 0, double *_maxDist* = 0) [protected],[virtual]

Attach the camera to a scene node.

Parameters

in	<i>_id</i>	ID of the visual to attach the camera to
in	<i>_inheritOrientation</i>	True means camera acquires the visual's orientation
in	<i>_minDist</i>	Minimum distance the camera is allowed to get to the visual
in	<i>_maxDist</i>	Maximum distance the camera is allowed to get from the visual

Returns

True on success

10.19.3.6 `virtual bool gazebo::rendering::Camera::AttachToVisualImpl (VisualIPtr _visual, bool _inheritOrientation, double _minDist = 0, double _maxDist = 0)` `[protected],[virtual]`

Attach the camera to a visual.

Parameters

<code>in</code>	<code>_visual</code>	The visual to attach the camera to
<code>in</code>	<code>_inheritOrientation</code>	True means camera acquires the visual's orientation
<code>in</code>	<code>_minDist</code>	Minimum distance the camera is allowed to get to the visual
<code>in</code>	<code>_maxDist</code>	Maximum distance the camera is allowed to get from the visual

Returns

True on success

Reimplemented in `gazebo::rendering::UserCamera` (p. 1140).

10.19.3.7 `template<typename T > event::ConnectionPtr gazebo::rendering::Camera::ConnectNewImageFrame (T _subscriber)` `[inline]`

Connect to the new image signal.

Parameters

<code>in</code>	<code>_subscriber</code>	Callback that is called when a new image is generated
-----------------	--------------------------	---

Returns

A pointer to the connection. This must be kept in scope.

10.19.3.8 `void gazebo::rendering::Camera::CreateRenderTexture (const std::string & _textureName)`

Set the render target.

Parameters

<code>in</code>	<code>_textureName</code>	Name of the new render texture
-----------------	---------------------------	--------------------------------

10.19.3.9 `void gazebo::rendering::Camera::DisconnectNewImageFrame (event::ConnectionPtr & _c)` `[inline]`

Disconnect from an image frame.

Parameters

<code>in</code>	<code>_c</code>	The connection to disconnect
-----------------	-----------------	------------------------------

10.19.3.10 void gazebo::rendering::Camera::EnableSaveFrame (bool *_enable*)

Enable or disable saving.

Parameters

in	<i>_enable</i>	Set to True to enable saving of frames
----	----------------	--

10.19.3.11 virtual void gazebo::rendering::Camera::Fini () [virtual]

Finalize the camera.

This function is called before the camera is destructed

Reimplemented in **gazebo::rendering::GpuLaser** (p. 471), **gazebo::rendering::DepthCamera** (p. 386), and **gazebo::rendering::UserCamera** (p. 1141).

10.19.3.12 float gazebo::rendering::Camera::GetAspectRatio () const

Get the aspect ratio.

Returns

The aspect ratio (width / height) in pixels

10.19.3.13 virtual float gazebo::rendering::Camera::GetAvgFPS () const [inline],[virtual]

Get the average FPS.

Returns

The average frames per second

Reimplemented in **gazebo::rendering::UserCamera** (p. 1141).

10.19.3.14 void gazebo::rendering::Camera::GetCameraToViewportRay (int *_screenx*, int *_screeny*, math::Vector3 & *_origin*, math::Vector3 & *_dir*)

Get a world space ray as cast from the camera through the viewport.

Parameters

in	<i>_screenx</i>	X coordinate in the camera's viewport, in pixels.
in	<i>_screeny</i>	Y coordinate in the camera's viewport, in pixels.
out	<i>_origin</i>	Origin in the world coordinate frame of the resulting ray
out	<i>_dir</i>	Direction of the resulting ray

10.19.3.15 bool gazebo::rendering::Camera::GetCaptureData () const

Return the value of this->captureData.

Returns

True if the camera is set to capture data.

10.19.3.16 `math::Vector3 gazebo::rendering::Camera::GetDirection () const`

Get the camera's direction vector.

Returns

Direction the camera is facing

10.19.3.17 `double gazebo::rendering::Camera::GetFarClip ()`

Get the far clip distance.

Returns

Far clip distance

10.19.3.18 `std::string gazebo::rendering::Camera::GetFrameFilename () [protected]`

Get the next frame filename based on SDF parameters.

Returns

The frame's filename

10.19.3.19 `math::Angle gazebo::rendering::Camera::GetHFOV () const`

Get the camera FOV (horizontal)

Returns

The horizontal field of view

10.19.3.20 `size_t gazebo::rendering::Camera::GetImageByteSize () const`

Get the image size in bytes.

Returns

Size in bytes

10.19.3.21 `static size_t gazebo::rendering::Camera::GetImageByteSize (unsigned int _width, unsigned int _height, const std::string & _format) [static]`

Calculate image byte size base on a few parameters.

Parameters

<code>in</code>	<code>_width</code>	Width of an image
<code>in</code>	<code>_height</code>	Height of an image
<code>in</code>	<code>_format</code>	Image format

Returns

Size of an image based on the parameters

10.19.3.22 `virtual const unsigned char* gazebo::rendering::Camera::GetImageData (unsigned int i = 0) [virtual]`

Get a pointer to the image data.

Get the raw image data from a camera's buffer.

Parameters

<code>in</code>	<code>_i</code>	Index of the camera's texture (0 = RGB, 1 = depth).
-----------------	-----------------	---

Returns

Pointer to the raw data, null if data is not available.

10.19.3.23 `unsigned int gazebo::rendering::Camera::GetImageDepth () const`

Get the depth of the image.

Returns

Depth of the image

10.19.3.24 `std::string gazebo::rendering::Camera::GetImageFormat () const`

Get the string representation of the image format.

Returns

String representation of the image format.

10.19.3.25 `virtual unsigned int gazebo::rendering::Camera::GetImageHeight () const [virtual]`

Get the height of the image.

Returns

Image height

Reimplemented in **`gazebo::rendering::UserCamera`** (p. 1141).

10.19.3.26 `virtual unsigned int gazebo::rendering::Camera::GetImageWidth () const` [virtual]

Get the width of the image.

Returns

Image width

Reimplemented in **`gazebo::rendering::UserCamera`** (p. 1141).

10.19.3.27 `bool gazebo::rendering::Camera::GetInitialized () const`

Return true if the camera has been initialized.

Returns

True if initialized was successful

10.19.3.28 `common::Time gazebo::rendering::Camera::GetLastRenderWallTime ()`

Get the last time the camera was rendered.

Returns

Time the camera was last rendered

10.19.3.29 `std::string gazebo::rendering::Camera::GetName () const`

Get the camera's unscoped name.

Returns

The name of the camera

10.19.3.30 `double gazebo::rendering::Camera::GetNearClip ()`

Get the near clip distance.

Returns

Near clip distance

10.19.3.31 `Ogre::Camera* gazebo::rendering::Camera::GetOgreCamera () const`

Get a pointer to the ogre camera.

Returns

Pointer to the OGRE camera

10.19.3.32 `Ogre::SceneNode* gazebo::rendering::Camera::GetPitchNode () const`

Deprecated: Get the camera's pitch scene node.

Returns

NULL. Use `GetSceheNode()` instead.

10.19.3.33 `double gazebo::rendering::Camera::GetRenderRate () const`

Get the render Hz rate.

Returns

The Hz rate

10.19.3.34 `Ogre::Texture* gazebo::rendering::Camera::GetRenderTexture () const`

Get the render texture.

Returns

Pointer to the render texture

10.19.3.35 `math::Vector3 gazebo::rendering::Camera::GetRight ()`

Get the viewport right vector.

Returns

The viewport right vector

10.19.3.36 `ScenePtr gazebo::rendering::Camera::GetScene () const`

Get the scene this camera is in.

Returns

Pointer to scene containing this camera

10.19.3.37 `Ogre::SceneNode* gazebo::rendering::Camera::GetSceneNode () const`

Get the camera's scene node.

Returns

The scene node the camera is attached to

10.19.3.38 `std::string gazebo::rendering::Camera::GetScopedName () const`

Get the camera's scoped name (scene_name::camera_name)

Returns

The name of the camera

10.19.3.39 `std::string gazebo::rendering::Camera::GetScreenshotPath () const`

Get the path to saved screenshots.

Returns

Path to saved screenshots.

10.19.3.40 `unsigned int gazebo::rendering::Camera::GetTextureHeight () const`

Get the height of the off-screen render texture.

Returns

Render texture height

10.19.3.41 `unsigned int gazebo::rendering::Camera::GetTextureWidth () const`

Get the width of the off-screen render texture.

Returns

Render texture width

10.19.3.42 `virtual unsigned int gazebo::rendering::Camera::GetTriangleCount () const` `[inline],[virtual]`

Get the triangle count.

Returns

The current triangle count

Reimplemented in **`gazebo::rendering::UserCamera`** (p. 1142).

10.19.3.43 `math::Vector3 gazebo::rendering::Camera::GetUp ()`

Get the viewport up vector.

Returns

The viewport up vector

10.19.3.44 `math::Angle gazebo::rendering::Camera::GetVFOV () const`

Get the camera FOV (vertical)

Returns

The vertical field of view

10.19.3.45 `Ogre::Viewport* gazebo::rendering::Camera::GetViewport () const`

Get a pointer to the `Ogre::Viewport`.

Returns

Pointer to the `Ogre::Viewport`

10.19.3.46 `unsigned int gazebo::rendering::Camera::GetViewportHeight () const`

Get the viewport height in pixels.

Returns

The viewport height

10.19.3.47 `unsigned int gazebo::rendering::Camera::GetViewportWidth () const`

Get the viewport width in pixels.

Returns

The viewport width

10.19.3.48 `unsigned int gazebo::rendering::Camera::GetWindowId () const`

Get the ID of the window this camera is rendering into.

Returns

The ID of the window.

10.19.3.49 `bool gazebo::rendering::Camera::GetWorldPointOnPlane (int _x, int _y, const math::Plane & _plane, math::Vector3 & _result)`

Get point on a plane.

Parameters

<code>in</code>	<code>_x</code>	X coordinate in camera's viewport, in pixels
<code>in</code>	<code>_y</code>	Y coordinate in camera's viewport, in pixels
<code>in</code>	<code>_plane</code>	Plane on which to find the intersecting point
<code>out</code>	<code>_result</code>	Point on the plane

Returns

True if a valid point was found

10.19.3.50 `math::Pose gazebo::rendering::Camera::GetWorldPose () const`

Get the world pose.

Returns

The pose of the camera in the world coordinate frame.

10.19.3.51 `math::Vector3 gazebo::rendering::Camera::GetWorldPosition () const`

Get the camera position in the world.

Returns

The world position of the camera

10.19.3.52 `math::Quaternion gazebo::rendering::Camera::GetWorldRotation () const`

Get the camera's orientation in the world.

Returns

The camera's orientation as a **math::Quaternion** (p. 824)

10.19.3.53 `double gazebo::rendering::Camera::GetZValue (int _x, int _y)`

Get the Z-buffer value at the given image coordinate.

Parameters

<code>in</code>	<code>_x</code>	Image coordinate; (0, 0) specifies the top-left corner.
<code>in</code>	<code>_y</code>	Image coordinate; (0, 0) specifies the top-left corner.

Returns

Image z value; note that this is arbitrarily scaled and is *not* the same as the depth value.

10.19.3.54 `virtual void gazebo::rendering::Camera::Init () [virtual]`

Initialize the camera.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 473), **gazebo::rendering::DepthCamera** (p. 386), and **gazebo::rendering::UserCamera** (p. 1143).

10.19.3.55 `bool gazebo::rendering::Camera::IsAnimating () const`

Return true if the camera is moving due to an animation.

10.19.3.56 `bool gazebo::rendering::Camera::IsVisible (VisualPtr _visual)`

Return true if the visual is within the camera's view frustum.

Parameters

<code>in</code>	<code>_visual</code>	The visual to check for visibility
-----------------	----------------------	------------------------------------

Returns

True if the `_visual` is in the camera's frustum

10.19.3.57 `bool gazebo::rendering::Camera::IsVisible (const std::string & _visualName)`

Return true if the visual is within the camera's view frustum.

Parameters

<code>in</code>	<code>_visualName</code>	Name of the visual to check for visibility
-----------------	--------------------------	--

Returns

True if the `_visual` is in the camera's frustum

10.19.3.58 `virtual void gazebo::rendering::Camera::Load (sdf::ElementPtr _sdf) [virtual]`

Load the camera with a set of parameters.

Parameters

<code>in</code>	<code>_sdf</code>	The SDF camera info
-----------------	-------------------	---------------------

Reimplemented in `gazebo::rendering::GpuLaser` (p. 473), `gazebo::rendering::DepthCamera` (p. 386), and `gazebo::rendering::UserCamera` (p. 1143).

10.19.3.59 `virtual void gazebo::rendering::Camera::Load () [virtual]`

Load the camera with default parameters.

Reimplemented in `gazebo::rendering::GpuLaser` (p. 473), `gazebo::rendering::DepthCamera` (p. 386), and `gazebo::rendering::UserCamera` (p. 1143).

10.19.3.60 `virtual bool gazebo::rendering::Camera::MoveToPosition (const math::Pose & _pose, double _time) [virtual]`

Move the camera to a position (this is an animated motion).

See Also

Camera::MoveToPositions (p. 216)

Parameters

in	<i>_pose</i>	End position of the camera
in	<i>_time</i>	Duration of the camera's movement

Reimplemented in **gazebo::rendering::UserCamera** (p. 1143).

```
10.19.3.61 bool gazebo::rendering::Camera::MoveToPositions ( const std::vector< math::Pose > & _pts, double _time,
boost::function< void()> _onComplete = NULL )
```

Move the camera to a series of poses (this is an animated motion).

See Also

Camera::MoveToPosition (p. 215)

Parameters

in	<i>_pts</i>	Vector of poses to move to
in	<i>_time</i>	Duration of the entire move
in	<i>_onComplete</i>	Callback that is called when the move is complete

```
10.19.3.62 virtual void gazebo::rendering::Camera::PostRender ( ) [virtual]
```

Post render.

Called after the render signal.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 473), **gazebo::rendering::DepthCamera** (p. 386), and **gazebo::rendering::UserCamera** (p. 1144).

```
10.19.3.63 void gazebo::rendering::Camera::ReadPixelBuffer ( ) [protected]
```

Read image data from pixel buffer.

```
10.19.3.64 void gazebo::rendering::Camera::Render ( )
```

Render the camera.

Called after the pre-render signal. This function will generate camera images.

```
10.19.3.65 void gazebo::rendering::Camera::Render ( bool _force )
```

Render the camera.

Called after the pre-render signal. This function will generate camera images.

Parameters

in	<i>_force</i>	Force camera to render. Ignore camera update rate.
----	---------------	--

10.19.3.66 virtual void gazebo::rendering::Camera::RenderImpl () [protected],[virtual]

Implementation of the render call.

10.19.3.67 void gazebo::rendering::Camera::RotatePitch (math::Angle *_angle*)

Rotate the camera around the pitch axis.

Parameters

in	<i>_angle</i>	Pitch amount
----	---------------	--------------

10.19.3.68 void gazebo::rendering::Camera::RotateYaw (math::Angle *_angle*)

Rotate the camera around the yaw axis.

Parameters

in	<i>_angle</i>	Rotation amount
----	---------------	-----------------

10.19.3.69 bool gazebo::rendering::Camera::SaveFrame (const std::string & *_filename*)

Save the last frame to disk.

Parameters

in	<i>_filename</i>	File in which to save a single frame
----	------------------	--------------------------------------

Returns

True if saving was successful

10.19.3.70 static bool gazebo::rendering::Camera::SaveFrame (const unsigned char * *_image*, unsigned int *_width*, unsigned int *_height*, int *_depth*, const std::string & *_format*, const std::string & *_filename*) [static]

Save a frame using an image buffer.

Parameters

in	<i>_image</i>	The raw image buffer
in	<i>_width</i>	Width of the image
in	<i>_height</i>	Height of the image
in	<i>_depth</i>	Depth of the image data
in	<i>_format</i>	Format the image data is in
in	<i>_filename</i>	Name of the file in which to write the frame

Returns

True if saving was successful

10.19.3.71 `void gazebo::rendering::Camera::SetAspectRatio (float _ratio)`

Set the aspect ratio.

Parameters

<i>in</i>	<i>_ratio</i>	The aspect ratio (width / height) in pixels
-----------	---------------	---

10.19.3.72 `void gazebo::rendering::Camera::SetCaptureData (bool _value)`

Set whether to capture data.

Parameters

<i>in</i>	<i>_value</i>	Set to true to capture data into a memory buffer.
-----------	---------------	---

10.19.3.73 `void gazebo::rendering::Camera::SetCaptureDataOnce ()`

Capture data once and save to disk.

10.19.3.74 `void gazebo::rendering::Camera::SetClipDist (float _near, float _far)`

Set the clip distances.

Parameters

<i>in</i>	<i>_near</i>	Near clip distance in meters
<i>in</i>	<i>_far</i>	Far clip distance in meters

10.19.3.75 `void gazebo::rendering::Camera::SetHFOV (math::Angle _angle)`

Set the camera FOV (horizontal)

Parameters

<i>in</i>	<i>_radians</i>	Horizontal field of view
-----------	-----------------	--------------------------

10.19.3.76 `void gazebo::rendering::Camera::SetImageHeight (unsigned int _h)`

Set the image height.

Parameters

<i>in</i>	<i>_h</i>	Image height
-----------	-----------	--------------

10.19.3.77 void gazebo::rendering::Camera::SetImageSize (unsigned int *_w*, unsigned int *_h*)

Set the image size.

Parameters

in	<i>_w</i>	Image width
in	<i>_h</i>	Image height

10.19.3.78 void gazebo::rendering::Camera::SetImageWidth (unsigned int *_w*)

Set the image height.

Parameters

in	<i>_w</i>	Image width
----	-----------	-------------

10.19.3.79 void gazebo::rendering::Camera::SetName (const std::string & *_name*)

Set the camera's name.

Parameters

in	<i>_name</i>	New name for the camera
----	--------------	-------------------------

10.19.3.80 void gazebo::rendering::Camera::SetRenderRate (double *_hz*)

Set the render Hz rate.

Parameters

in	<i>_hz</i>	The Hz rate
----	------------	-------------

10.19.3.81 virtual void gazebo::rendering::Camera::SetRenderTarget (Ogre::RenderTarget * *_target*) [virtual]

Set the camera's render target.

Parameters

in	<i>_target</i>	Pointer to the render target
----	----------------	------------------------------

Reimplemented in **gazebo::rendering::UserCamera** (p. 1145).

10.19.3.82 void gazebo::rendering::Camera::SetSaveFramePathname (const std::string & *_pathname*)

Set the save frame pathname.

Parameters

in	<code>_pathname</code>	Directory in which to store saved image frames
----	------------------------	--

10.19.3.83 void gazebo::rendering::Camera::SetScene (ScenePtr *_scene*)

Set the scene this camera is viewing.

Parameters

in	<code>_scene</code>	Pointer to the scene
----	---------------------	----------------------

10.19.3.84 void gazebo::rendering::Camera::SetSceneNode (Ogre::SceneNode * *_node*)

Set the camera's scene node.

Parameters

in	<code>_node</code>	The scene nodes to attach the camera to
----	--------------------	---

10.19.3.85 void gazebo::rendering::Camera::SetWindowId (unsigned int *_windowId*)

10.19.3.86 virtual void gazebo::rendering::Camera::SetWorldPose (const math::Pose & *_pose*) [virtual]

Set the global pose of the camera.

Parameters

in	<code>_pose</code>	The new math::Pose (p. 797) of the camera
----	--------------------	--

Reimplemented in **gazebo::rendering::UserCamera** (p. 1146).

10.19.3.87 void gazebo::rendering::Camera::SetWorldPosition (const math::Vector3 & *_pos*)

Set the world position.

Parameters

in	<code>_pos</code>	The new position of the camera
----	-------------------	--------------------------------

10.19.3.88 void gazebo::rendering::Camera::SetWorldRotation (const math::Quaternion & *_quat*)

Set the world orientation.

Parameters

in	<code>_quat</code>	The new orientation of the camera
----	--------------------	-----------------------------------

10.19.3.89 void gazebo::rendering::Camera::ShowWireframe (bool *_s*)

Set whether to view the world in wireframe.

Parameters

in	<i>_s</i>	Set to True to render objects as wireframe
----	-----------	--

10.19.3.90 void gazebo::rendering::Camera::ToggleShowWireframe ()

Toggle whether to view the world in wireframe.

10.19.3.91 void gazebo::rendering::Camera::TrackVisual (const std::string & *_visualName*)

Set the camera to track a scene node.

Parameters

in	<i>_visualName</i>	Name of the visual to track
----	--------------------	-----------------------------

10.19.3.92 bool gazebo::rendering::Camera::TrackVisualImpl (const std::string & *_visualName*) [protected]

Implementation of the **Camera::TrackVisual** (p. 221) call.

Parameters

in	<i>_visualName</i>	Name of the visual to track
----	--------------------	-----------------------------

Returns

True if able to track the visual

10.19.3.93 virtual bool gazebo::rendering::Camera::TrackVisualImpl (VisualPtr *_visual*) [protected],[virtual]

Set the camera to track a scene node.

Parameters

in	<i>_visual</i>	The visual to track
----	----------------	---------------------

Returns

True if able to track the visual

Reimplemented in **gazebo::rendering::UserCamera** (p. 1146).

10.19.3.94 void gazebo::rendering::Camera::Translate (const math::Vector3 & *_direction*)

Translate the camera.

Parameters

<code>in</code>	<code>_direction</code>	The translation vector
-----------------	-------------------------	------------------------

10.19.3.95 `virtual void gazebo::rendering::Camera::Update () [virtual]`

Reimplemented in `gazebo::rendering::UserCamera` (p. 1146).

10.19.4 Member Data Documentation

10.19.4.1 `Ogre::AnimationState* gazebo::rendering::Camera::animState [protected]`

Animation state, used to animate the camera.

10.19.4.2 `unsigned char* gazebo::rendering::Camera::bayerFrameBuffer [protected]`

Buffer for a bayer image frame.

10.19.4.3 `Ogre::Camera* gazebo::rendering::Camera::camera [protected]`

The OGRE camera.

10.19.4.4 `bool gazebo::rendering::Camera::captureData [protected]`

True to capture frames into an image buffer.

10.19.4.5 `bool gazebo::rendering::Camera::captureDataOnce [protected]`

True to capture a frame once and save to disk.

10.19.4.6 `std::vector<event::ConnectionPtr> gazebo::rendering::Camera::connections [protected]`

The camera's event connections.

10.19.4.7 `int gazebo::rendering::Camera::imageFormat [protected]`

Format for saving images.

10.19.4.8 `int gazebo::rendering::Camera::imageHeight [protected]`

Save image height.

10.19.4.9 `int gazebo::rendering::Camera::imageWidth [protected]`

Save image width.

10.19.4.10 `bool gazebo::rendering::Camera::initialized` [protected]

True if initialized.

10.19.4.11 `common::Time gazebo::rendering::Camera::lastRenderWallTime` [protected]

Time the last frame was rendered.

10.19.4.12 `std::string gazebo::rendering::Camera::name` [protected]

Name of the camera.

10.19.4.13 `bool gazebo::rendering::Camera::newData` [protected]

True if new data is available.

10.19.4.14 `event::EventT<void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>
gazebo::rendering::Camera::newImageFrame` [protected]

Event triggered when a new frame is generated.

10.19.4.15 `boost::function<void()> gazebo::rendering::Camera::onAnimationComplete` [protected]

User callback for when an animation completes.

10.19.4.16 `common::Time gazebo::rendering::Camera::prevAnimTime` [protected]

Previous time the camera animation was updated.

10.19.4.17 `Ogre::RenderTarget* gazebo::rendering::Camera::renderTarget` [protected]

Target that renders frames.

10.19.4.18 `Ogre::Texture* gazebo::rendering::Camera::renderTexture` [protected]

Texture that receives results from rendering.

10.19.4.19 `std::list<msgs::Request> gazebo::rendering::Camera::requests` [protected]

List of requests.

10.19.4.20 `unsigned int gazebo::rendering::Camera::saveCount` [protected]

Number of saved frames.

10.19.4.21 `unsigned char* gazebo::rendering::Camera::saveFrameBuffer` [protected]

10.19.4.22 `ScenePtr gazebo::rendering::Camera::scene` [protected]

Pointer to the scene.

10.19.4.23 `Ogre::SceneNode* gazebo::rendering::Camera::sceneNode` [protected]

Scene (p. 879) node that controls camera position and orientation.

10.19.4.24 `std::string gazebo::rendering::Camera::scopedName` [protected]

Scene (p. 879) scoped name of the camera.

10.19.4.25 `std::string gazebo::rendering::Camera::scopedUniqueName` [protected]

Scene (p. 879) scoped name of the camera with a unique ID.

10.19.4.26 `std::string gazebo::rendering::Camera::screenshotPath` [protected]

Path to saved screenshots.

10.19.4.27 `sdf::ElementPtr gazebo::rendering::Camera::sdf` [protected]

Camera (p. 197)'s SDF values.

10.19.4.28 `unsigned int gazebo::rendering::Camera::textureHeight` [protected]

Height of the render texture.

10.19.4.29 `unsigned int gazebo::rendering::Camera::textureWidth` [protected]

Width of the render texture.

10.19.4.30 `Ogre::Viewport* gazebo::rendering::Camera::viewport` [protected]

Viewport the ogre camera uses.

10.19.4.31 `unsigned int gazebo::rendering::Camera::windowId` [protected]

ID of the window that the camera is attached to.

The documentation for this class was generated from the following file:

- **Camera.hh**

10.20 gazebo::rendering::CameraPrivate Class Reference

Private data for the **Camera** (p. 197) class.

```
#include <CameraPrivate.hh>
```

Public Types

- typedef std::list
< boost::shared_ptr
< msgs::CameraCmd const > > **CameraCmdMsgs_L**

Public Attributes

- **transport::SubscriberPtr cmdSub**
Subscribe to camera command topic.
- **CameraCmdMsgs_L commandMsgs**
List of camera cmd messages.
- Ogre::CompositorInstance * **dIGBufferInstance**
Deferred lighting geometry buffer.
- Ogre::CompositorInstance * **dIMergeInstance**
Deferred lighting merge compositor.
- Ogre::CompositorInstance * **dsGBufferInstance**
Deferred shading geometry buffer.
- Ogre::CompositorInstance * **dsMergeInstance**
Deferred shading merge compositor.
- std::deque< std::pair
< **math::Pose**, double > > **moveToPositionQueue**
Queue of move positions.
- **transport::NodePtr node**
Communication Node.
- boost::mutex **receiveMutex**
Mutex to lock the various message buffers.
- **common::Time renderPeriod**
Render period.
- Ogre::CompositorInstance * **ssaInstance**
Screen space ambient occlusion compositor.
- **VisualPtr trackedVisual**
***Visual** (p. 1196) that the camera is tracking.*
- **common::PID trackVisualPID**
Position PID used to track a visual smoothly.
- **common::PID trackVisualPitchPID**
Pitch PID used to track a visual smoothly.
- **common::PID trackVisualYawPID**
Yaw PID used to track a visual smoothly.

Static Public Attributes

- static unsigned int **cameraCounter**
Counter used to create unique camera names.

10.20.1 Detailed Description

Private data for the **Camera** (p. 197) class.

10.20.2 Member Typedef Documentation

10.20.2.1 typedef std::list<boost::shared_ptr<msgs::CameraCmd const> > gazebo::rendering::CameraPrivate::CameraCmdMsgs_L

10.20.3 Member Data Documentation

10.20.3.1 unsigned int gazebo::rendering::CameraPrivate::cameraCounter [static]

Counter used to create unique camera names.

10.20.3.2 transport::SubscriberPtr gazebo::rendering::CameraPrivate::cmdSub

Subscribe to camera command topic.

10.20.3.3 CameraCmdMsgs_L gazebo::rendering::CameraPrivate::commandMsgs

List of camera cmd messages.

10.20.3.4 Ogre::CompositorInstance* gazebo::rendering::CameraPrivate::dlGBufferInstance

Deferred lighting geometry buffer.

10.20.3.5 Ogre::CompositorInstance* gazebo::rendering::CameraPrivate::dlMergeInstance

Deferred lighting merge compositor.

10.20.3.6 Ogre::CompositorInstance* gazebo::rendering::CameraPrivate::dsGBufferInstance

Deferred shading geometry buffer.

10.20.3.7 Ogre::CompositorInstance* gazebo::rendering::CameraPrivate::dsMergeInstance

Deferred shading merge compositor.

10.20.3.8 std::deque<std::pair<math::Pose, double> > gazebo::rendering::CameraPrivate::moveToPositionQueue

Queue of move positions.

10.20.3.9 `transport::NodePtr` gazebo::rendering::CameraPrivate::node

Communication Node.

10.20.3.10 `boost::mutex` gazebo::rendering::CameraPrivate::receiveMutex

Mutex to lock the various message buffers.

10.20.3.11 `common::Time` gazebo::rendering::CameraPrivate::renderPeriod

Render period.

10.20.3.12 `Ogre::CompositorInstance*` gazebo::rendering::CameraPrivate::ssaInstance

Screen space ambient occlusion compositor.

10.20.3.13 `VisualPtr` gazebo::rendering::CameraPrivate::trackedVisual

Visual (p. 1196) that the camera is tracking.

10.20.3.14 `common::PID` gazebo::rendering::CameraPrivate::trackVisualPID

Position PID used to track a visual smoothly.

10.20.3.15 `common::PID` gazebo::rendering::CameraPrivate::trackVisualPitchPID

Pitch PID used to track a visual smoothly.

10.20.3.16 `common::PID` gazebo::rendering::CameraPrivate::trackVisualYawPID

Yaw PID used to track a visual smoothly.

The documentation for this class was generated from the following file:

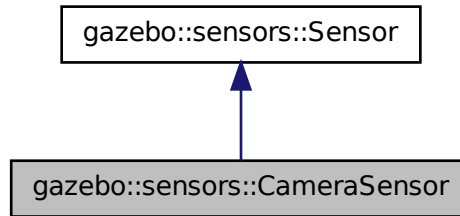
- **CameraPrivate.hh**

10.21 gazebo::sensors::CameraSensor Class Reference

Basic camera sensor.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::CameraSensor:



Public Member Functions

- **CameraSensor** ()
Constructor.
- virtual **~CameraSensor** ()
Destructor.
- **rendering::CameraPtr GetCamera** () const
*Returns a pointer to the **rendering::Camera** (p. 197).*
- const unsigned char * **GetImageData** ()
Gets the raw image data from the sensor.
- unsigned int **GetImageHeight** () const
Gets the height of the image in pixels.
- unsigned int **GetImageWidth** () const
Gets the width of the image in pixels.
- virtual std::string **GetTopic** () const
Gets the topic name of the sensor.
- virtual void **Init** ()
Initialize the camera.
- virtual bool **IsActive** ()
Returns true if sensor generation is active.
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)
Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.
- bool **SaveFrame** (const std::string &_filename)
Saves the image to the disk.

Protected Member Functions

- virtual void **Fini** ()
Finalize the camera.
- virtual bool **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.21.1 Detailed Description

Basic camera sensor.

This sensor is used for simulating standard monocular cameras

10.21.2 Constructor & Destructor Documentation

10.21.2.1 gazebo::sensors::CameraSensor::CameraSensor ()

Constructor.

10.21.2.2 virtual gazebo::sensors::CameraSensor::~~CameraSensor () [virtual]

Destructor.

10.21.3 Member Function Documentation

10.21.3.1 virtual void gazebo::sensors::CameraSensor::Fini () [protected],[virtual]

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 912).

10.21.3.2 rendering::CameraPtr gazebo::sensors::CameraSensor::GetCamera () const [inline]

Returns a pointer to the **rendering::Camera** (p. 197).

Returns

The Pointer to the camera sensor.

10.21.3.3 const unsigned char* gazebo::sensors::CameraSensor::GetImageData ()

Gets the raw image data from the sensor.

Returns

The pointer to the image data array.

10.21.3.4 unsigned int gazebo::sensors::CameraSensor::GetImageHeight () const

Gets the height of the image in pixels.

Returns

The image height in pixels.

10.21.3.5 `unsigned int gazebo::sensors::CameraSensor::GetImageWidth () const`

Gets the width of the image in pixels.

Returns

The image width in pixels.

10.21.3.6 `virtual std::string gazebo::sensors::CameraSensor::GetTopic () const` [virtual]

Gets the topic name of the sensor.

Returns

Topic name

Todo to be implemented

Reimplemented from `gazebo::sensors::Sensor` (p. 914).

10.21.3.7 `virtual void gazebo::sensors::CameraSensor::Init ()` [virtual]

Initialize the camera.

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.21.3.8 `virtual bool gazebo::sensors::CameraSensor::IsActive ()` [virtual]

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.21.3.9 `virtual void gazebo::sensors::CameraSensor::Load (const std::string & _worldName, sdf::ElementPtr _sdf)`
[virtual]

Load the sensor with SDF parameters.

Parameters

<code>in</code>	<code>_sdf</code>	SDF Sensor (p. 907) parameters
<code>in</code>	<code>_worldName</code>	Name of world to load from

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.21.3.10 virtual void gazebo::sensors::CameraSensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	<i>_worldName</i>	Name of world to load from
----	-------------------	----------------------------

Reimplemented from **gazebo::sensors::Sensor** (p.915).

10.21.3.11 bool gazebo::sensors::CameraSensor::SaveFrame (const std::string & *_filename*)

Saves the image to the disk.

Parameters

in	<i>_filename</i>	The name of the file to be saved.
----	------------------	-----------------------------------

Returns

True if successful, false if unsuccessful.

10.21.3.12 virtual bool gazebo::sensors::CameraSensor::UpdateImpl (bool) [protected],[virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	<i>_force</i>	True if update is forced, false if not
----	---------------	--

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p.917).

The documentation for this class was generated from the following file:

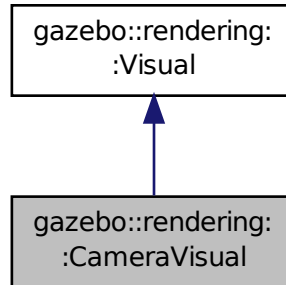
- **CameraSensor.hh**

10.22 gazebo::rendering::CameraVisual Class Reference

Basic camera visualization.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::CameraVisual:



Public Member Functions

- **CameraVisual** (const std::string &_name, **VisualPtr** _vis)
Constructor.
- virtual ~**CameraVisual** ()
Destructor.
- void **Load** (unsigned int _width, unsigned int _height)
*Load the **Visual** (p. 1196).*

Additional Inherited Members

10.22.1 Detailed Description

Basic camera visualization.

This class is used to visualize a camera image generated from a CameraSensor. The sensor's image is drawn on a billboard in the 3D environment.

10.22.2 Constructor & Destructor Documentation

10.22.2.1 gazebo::rendering::CameraVisual::CameraVisual (const std::string & _name, **VisualPtr** _vis)

Constructor.

Parameters

in	<code>_name</code>	Name of the Visual (p. 1196)
in	<code>_vis</code>	Pointer to the parent Visual (p. 1196)

10.22.2.2 virtual gazebo::rendering::CameraVisual::~~CameraVisual () [virtual]

Destructor.

10.22.3 Member Function Documentation

10.22.3.1 void gazebo::rendering::CameraVisual::Load (unsigned int *_width*, unsigned int *_height*)

Load the **Visual** (p. 1196).

Parameters

in	<i>_width</i>	Width of the Camera (p. 197) image
in	<i>_height</i>	Height of the Camera (p. 197) image

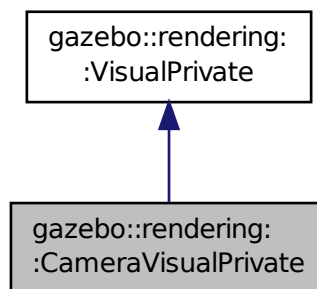
The documentation for this class was generated from the following file:

- **CameraVisual.hh**

10.23 gazebo::rendering::CameraVisualPrivate Class Reference

```
#include <CameraVisualPrivate.hh>
```

Inheritance diagram for gazebo::rendering::CameraVisualPrivate:



Public Attributes

- **CameraPtr camera**
Pointer to the camera.
- `std::vector< event::ConnectionPtr > connections`
Event connections.

Additional Inherited Members

10.23.1 Member Data Documentation

10.23.1.1 CameraPtr gazebo::rendering::CameraVisualPrivate::camera

Pointer to the camera.

10.23.1.2 std::vector<event::ConnectionPtr> gazebo::rendering::CameraVisualPrivate::connections

Event connections.

The documentation for this class was generated from the following file:

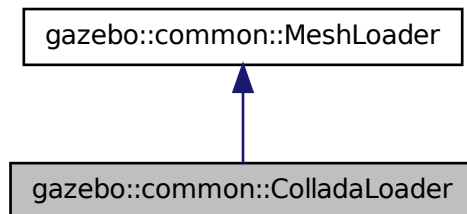
- **CameraVisualPrivate.hh**

10.24 gazebo::common::ColladaLoader Class Reference

Class used to load Collada mesh files.

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::common::ColladaLoader:



Public Member Functions

- **ColladaLoader** ()
Constructor.
- virtual **~ColladaLoader** ()
Destructor.
- virtual **Mesh * Load** (const std::string &_filename)
Load a mesh.

10.24.1 Detailed Description

Class used to load Collada mesh files.

10.24.2 Constructor & Destructor Documentation

10.24.2.1 gazebo::common::ColladaLoader::ColladaLoader ()

Constructor.

10.24.2.2 virtual gazebo::common::ColladaLoader::~~ColladaLoader () [virtual]

Destructor.

10.24.3 Member Function Documentation

10.24.3.1 virtual Mesh* gazebo::common::ColladaLoader::Load (const std::string & *_filename*) [virtual]

Load a mesh.

Parameters

in	<i>_filename</i>	Collada file to load
----	------------------	----------------------

Returns

Pointer to a new **Mesh** (p. 660)

Implements **gazebo::common::MeshLoader** (p. 670).

The documentation for this class was generated from the following file:

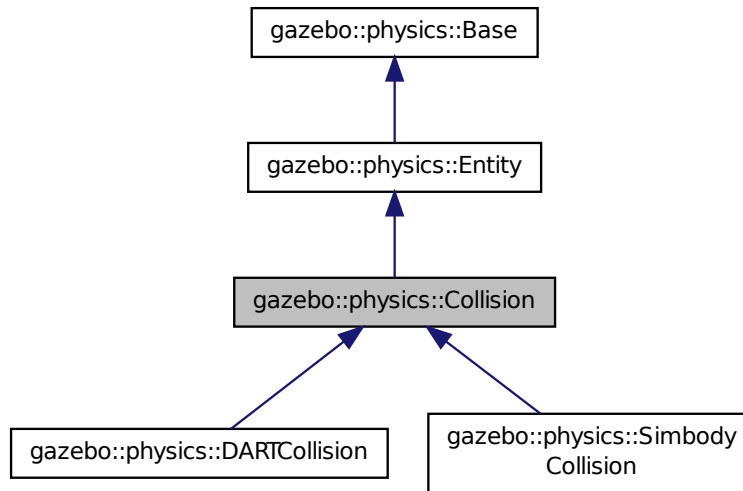
- **ColladaLoader.hh**

10.25 gazebo::physics::Collision Class Reference

Base (p. 168) class for all collision entities.

```
#include <Collision.hh>
```

Inheritance diagram for gazebo::physics::Collision:



Public Member Functions

- **Collision** (**LinkPtr** _link)
Constructor.
- virtual **~Collision** ()
Destructor.
- void **AddContact** (const **Contact** &_contact) **GAZEBO_DEPRECATED**(2.0)
Add an occurrence of a contact to this collision.
- void **FillMsg** (msgs::Collision &_msg)
Fill a collision message.
- virtual void **Fini** ()
Finalize the collision.
- virtual **math::Box GetBoundingBox** () const =0
Get the bounding box for this collision.
- bool **GetContactsEnabled** () const **GAZEBO_DEPRECATED**(2.0)
Return true if contacts are on.
- float **GetLaserRetro** () const
Get the laser retro reflectiveness.
- **LinkPtr GetLink** () const
Get the link this collision belongs to.
- virtual unsigned int **GetMaxContacts** ()
returns number of contacts allowed for this collision.
- **ModelPtr GetModel** () const
Get the model this collision belongs to.
- virtual **math::Vector3 GetRelativeAngularAccel** () const

- Get the angular acceleration of the collision.*

 - virtual **math::Vector3 GetRelativeAngularVel** () const

Get the angular velocity of the collision.

 - virtual **math::Vector3 GetRelativeLinearAccel** () const

Get the linear acceleration of the collision.

 - virtual **math::Vector3 GetRelativeLinearVel** () const

Get the linear velocity of the collision.

 - **ShapePtr GetShape** () const

Get the collision shape.

 - unsigned int **GetShapeType** ()

Get the shape type.

 - **CollisionState GetState** ()

Get the collision state.

 - **SurfaceParamsPtr GetSurface** () const

Get the surface parameters.

 - virtual **math::Vector3 GetWorldAngularAccel** () const

Get the angular acceleration of the collision in the world frame.

 - virtual **math::Vector3 GetWorldAngularVel** () const

Get the angular velocity of the collision in the world frame.

 - virtual **math::Vector3 GetWorldLinearAccel** () const

Get the linear acceleration of the collision in the world frame.

 - virtual **math::Vector3 GetWorldLinearVel** () const

Get the linear velocity of the collision in the world frame.

 - virtual void **Init** ()

Initialize the collision.

 - bool **IsPlaceable** () const

Return whether this collision is movable.

 - virtual void **Load** (sdf::ElementPtr _sdf)

Load the collision.

 - void **ProcessMsg** (const msgs::Collision &_msg)

Update parameters from a message.

 - virtual void **SetCategoryBits** (unsigned int _bits)=0

Set the category bits, used during collision detection.

 - virtual void **SetCollideBits** (unsigned int _bits)=0

Set the collide bits, used during collision detection.

 - void **SetCollision** (bool _placeable)

Set the encapsulated collision object.

 - void **SetContactsEnabled** (bool _enable) **GAZEBO_DEPRECATED(2.0)**

Turn contact recording on or off.

 - void **SetLaserRetro** (float _retro)

Set the laser retro reflectiveness.

 - virtual void **SetMaxContacts** (unsigned int _maxContacts)

Number of contacts allowed for this collision.

 - void **SetScale** (const **math::Vector3** &_scale)

Set the scale of the collision.

 - void **SetShape** (**ShapePtr** _shape)

Set the shape for this collision.

- void **SetState** (const **CollisionState** &_state)
Set the current collision state.
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)
Update the parameters using new sdf values.

Protected Attributes

- **LinkPtr link**
The link this collision belongs to.
- bool **placeable**
Flag for placeable.
- **ShapePtr shape**
*Pointer to **physics::Shape** (p. 932).*
- **SurfaceParamsPtr surface**
The surface parameters.

Additional Inherited Members

10.25.1 Detailed Description

Base (p. 168) class for all collision entities.

10.25.2 Constructor & Destructor Documentation

10.25.2.1 gazebo::physics::Collision::Collision (**LinkPtr link**) [explicit]

Constructor.

Parameters

in	link	Link (p. 595) that contains this collision object.
----	-------------	---

10.25.2.2 virtual gazebo::physics::Collision::~~Collision () [virtual]

Destructor.

10.25.3 Member Function Documentation

10.25.3.1 void gazebo::physics::Collision::AddContact (const **Contact** & **contact**)

Add an occurrence of a contact to this collision.

Deprecated by?

Parameters

in	contact	The contact which was detected by a collision engine.
----	----------------	---

10.25.3.2 void gazebo::physics::Collision::FillMsg (msgs::Collision & _msg)

Fill a collision message.

Parameters

out	_msg	The message to fill with this collision's data.
-----	------	---

10.25.3.3 virtual void gazebo::physics::Collision::Fini () [virtual]

Finalize the collision.

Reimplemented from **gazebo::physics::Entity** (p. 407).

Reimplemented in **gazebo::physics::DARTCollision** (p. 311).

10.25.3.4 virtual math::Box gazebo::physics::Collision::GetBoundingBox () const [pure virtual]

Get the bounding box for this collision.

Returns

The bounding box.

Reimplemented from **gazebo::physics::Entity** (p. 407).

Implemented in **gazebo::physics::DARTCollision** (p. 311), and **gazebo::physics::SimbodyCollision** (p. 945).

10.25.3.5 bool gazebo::physics::Collision::GetContactsEnabled () const

Return true if contacts are on.

Deprecated by?

Returns

True if contacts are on.

10.25.3.6 float gazebo::physics::Collision::GetLaserRetro () const

Get the laser retro reflectiveness.

Returns

The laser retro value.

10.25.3.7 LinkPtr gazebo::physics::Collision::GetLink () const

Get the link this collision belongs to.

Returns

The parent **Link** (p. 595).

10.25.3.8 `virtual unsigned int gazebo::physics::Collision::GetMaxContacts () [virtual]`

returns number of contacts allowed for this collision.

This overrides global value (in **PhysicsEngine** (p. 766)) if specified.

Returns

max num contacts allowed for this collision.

10.25.3.9 `ModelPtr gazebo::physics::Collision::GetModel () const`

Get the model this collision belongs to.

Returns

The parent model.

10.25.3.10 `virtual math::Vector3 gazebo::physics::Collision::GetRelativeAngularAccel () const [virtual]`

Get the angular acceleration of the collision.

Returns

The angular acceleration of the collision.

Reimplemented from **gazebo::physics::Entity** (p. 409).

10.25.3.11 `virtual math::Vector3 gazebo::physics::Collision::GetRelativeAngularVel () const [virtual]`

Get the angular velocity of the collision.

Returns

The angular velocity of the collision.

Reimplemented from **gazebo::physics::Entity** (p. 409).

10.25.3.12 `virtual math::Vector3 gazebo::physics::Collision::GetRelativeLinearAccel () const [virtual]`

Get the linear acceleration of the collision.

Returns

The linear acceleration of the collision.

Reimplemented from **gazebo::physics::Entity** (p. 409).

10.25.3.13 `virtual math::Vector3 gazebo::physics::Collision::GetRelativeLinearVel () const` [virtual]

Get the linear velocity of the collision.

Returns

The linear velocity relative to the parent model.

Reimplemented from **gazebo::physics::Entity** (p. 410).

10.25.3.14 `ShapePtr gazebo::physics::Collision::GetShape () const`

Get the collision shape.

Returns

The collision shape.

10.25.3.15 `unsigned int gazebo::physics::Collision::GetShapeType ()`

Get the shape type.

Returns

The shape type.

See Also

EntityType (p. 172)

10.25.3.16 `CollisionState gazebo::physics::Collision::GetState ()`

Get the collision state.

Returns

The collision state.

10.25.3.17 `SurfaceParamsPtr gazebo::physics::Collision::GetSurface () const` [inline]

Get the surface parameters.

Returns

The surface parameters.

10.25.3.18 `virtual math::Vector3 gazebo::physics::Collision::GetWorldAngularAccel () const [virtual]`

Get the angular acceleration of the collision in the world frame.

Returns

The angular acceleration of the collision in the world frame.

Reimplemented from `gazebo::physics::Entity` (p. 410).

10.25.3.19 `virtual math::Vector3 gazebo::physics::Collision::GetWorldAngularVel () const [virtual]`

Get the angular velocity of the collision in the world frame.

Returns

The angular velocity of the collision in the world frame.

Reimplemented from `gazebo::physics::Entity` (p. 410).

10.25.3.20 `virtual math::Vector3 gazebo::physics::Collision::GetWorldLinearAccel () const [virtual]`

Get the linear acceleration of the collision in the world frame.

Returns

The linear acceleration of the collision in the world frame.

Reimplemented from `gazebo::physics::Entity` (p. 411).

10.25.3.21 `virtual math::Vector3 gazebo::physics::Collision::GetWorldLinearVel () const [virtual]`

Get the linear velocity of the collision in the world frame.

Returns

The linear velocity of the collision in the world frame.

Reimplemented from `gazebo::physics::Entity` (p. 411).

10.25.3.22 `virtual void gazebo::physics::Collision::Init () [virtual]`

Initialize the collision.

Reimplemented from `gazebo::physics::Base` (p. 177).

Reimplemented in `gazebo::physics::DARTCollision` (p. 312).

10.25.3.23 `bool gazebo::physics::Collision::IsPlaceable () const`

Return whether this collision is movable.

Example on an immovable object is a ray.

Returns

True if the object is immovable.

10.25.3.24 virtual void gazebo::physics::Collision::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the collision.

Parameters

in	<i>_sdf</i>	SDF to load from.
----	-------------	-------------------

Reimplemented from **gazebo::physics::Entity** (p. 411).

Reimplemented in **gazebo::physics::SimbodyCollision** (p. 945), and **gazebo::physics::DARTCollision** (p. 312).

10.25.3.25 void gazebo::physics::Collision::ProcessMsg (const msgs::Collision & *_msg*)

Update parameters from a message.

Parameters

in	<i>_msg</i>	Message to update from.
----	-------------	-------------------------

10.25.3.26 virtual void gazebo::physics::Collision::SetCategoryBits (unsigned int *_bits*) [pure virtual]

Set the category bits, used during collision detection.

Parameters

in	<i>_bits</i>	The bits to set.
----	--------------	------------------

Implemented in **gazebo::physics::DARTCollision** (p. 313), and **gazebo::physics::SimbodyCollision** (p. 946).

10.25.3.27 virtual void gazebo::physics::Collision::SetCollideBits (unsigned int *_bits*) [pure virtual]

Set the collide bits, used during collision detection.

Parameters

in	<i>_bits</i>	The bits to set.
----	--------------	------------------

Implemented in **gazebo::physics::DARTCollision** (p. 313), and **gazebo::physics::SimbodyCollision** (p. 946).

10.25.3.28 void gazebo::physics::Collision::SetCollision (bool *_placeable*)

Set the encapsulated collision object.

Parameters

in	<i>_placeable</i>	True to make the object movable.
----	-------------------	----------------------------------

10.25.3.29 void gazebo::physics::Collision::SetContactsEnabled (bool *_enable*)

Turn contact recording on or off.

Deprecated by?

Parameters

in	<i>_enable</i>	True to enable collision contacts.
----	----------------	------------------------------------

10.25.3.30 void gazebo::physics::Collision::SetLaserRetro (float *_retro*)

Set the laser retro reflectiveness.

Parameters

in	<i>_retro</i>	The laser retro value.
----	---------------	------------------------

10.25.3.31 virtual void gazebo::physics::Collision::SetMaxContacts (unsigned int *_maxContacts*) [virtual]

Number of contacts allowed for this collision.

This overrides global value (in **PhysicsEngine** (p. 766)) if specified.

Parameters

in	<i>_maxContacts</i>	max num contacts allowed for this collision.
----	---------------------	--

10.25.3.32 void gazebo::physics::Collision::SetScale (const math::Vector3 & *_scale*)

Set the scale of the collision.

Parameters

in	<i>_scale</i>	Scale to set the collision to.
----	---------------	--------------------------------

10.25.3.33 void gazebo::physics::Collision::SetShape (ShapePtr *_shape*)

Set the shape for this collision.

Parameters

in	<i>_shape</i>	The shape for this collision object.
----	---------------	--------------------------------------

10.25.3.34 void gazebo::physics::Collision::SetState (const CollisionState & *_state*)

Set the current collision state.

Parameters

in	<i>The</i>	collision state.
----	------------	------------------

10.25.3.35 virtual void gazebo::physics::Collision::UpdateParameters (sdf::ElementPtr *.sdf*) [virtual]

Update the parameters using new sdf values.

Parameters

in	<i>_sdf</i>	SDF values to update from.
----	-------------	----------------------------

Reimplemented from **gazebo::physics::Entity** (p. 414).

10.25.4 Member Data Documentation

10.25.4.1 **LinkPtr** gazebo::physics::Collision::link [protected]

The link this collision belongs to.

10.25.4.2 **bool** gazebo::physics::Collision::placeable [protected]

Flag for placeable.

10.25.4.3 **ShapePtr** gazebo::physics::Collision::shape [protected]

Pointer to **physics::Shape** (p. 932).

10.25.4.4 **SurfaceParamsPtr** gazebo::physics::Collision::surface [protected]

The surface parameters.

The documentation for this class was generated from the following file:

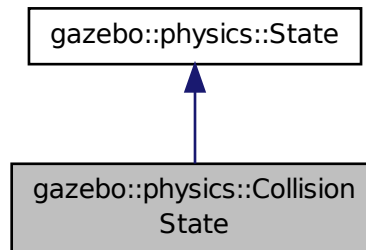
- **Collision.hh**

10.26 gazebo::physics::CollisionState Class Reference

Store state information of a **physics::Collision** (p. 235) object.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::CollisionState:



Public Member Functions

- **CollisionState** ()
Default constructor.
- **CollisionState** (const **CollisionPtr** _collision)
Constructor.
- **CollisionState** (const sdf::ElementPtr _sdf)
Constructor.
- virtual ~**CollisionState** ()
Destructor.
- void **FillSDF** (sdf::ElementPtr _sdf)
Populate a state SDF element with data from the object.
- const **math::Pose** & **GetPose** () const
*Get the **Collision** (p. 235) pose.*
- bool **IsZero** () const
Return true if the values in the state are zero.
- virtual void **Load** (const sdf::ElementPtr _elem)
Load state from SDF element.
- **CollisionState operator+** (const **CollisionState** &_state) const
Addition operator.
- **CollisionState operator-** (const **CollisionState** &_state) const
Subtraction operator.
- **CollisionState & operator=** (const **CollisionState** &_state)
Assignment operator.

Friends

- std::ostream & **operator<<** (std::ostream &_out, const **gazebo::physics::CollisionState** &_state)
Stream insertion operator.

Additional Inherited Members

10.26.1 Detailed Description

Store state information of a **physics::Collision** (p. 235) object.

This class captures the entire state of a **Collision** (p. 235) at one specific time during a simulation run.

State (p. 1068) of a **Collision** (p. 235) is its Pose.

10.26.2 Constructor & Destructor Documentation

10.26.2.1 gazebo::physics::CollisionState::CollisionState ()

Default constructor.

10.26.2.2 gazebo::physics::CollisionState::CollisionState (const CollisionPtr *_collision*) [explicit]

Constructor.

Build a **CollisionState** (p. 245) from an existing **Collision** (p. 235).

Parameters

in	<i>_model</i>	Pointer to the Link (p. 595) from which to gather state info.
----	---------------	--

10.26.2.3 gazebo::physics::CollisionState::CollisionState (const sdf::ElementPtr *_sdf*) [explicit]

Constructor.

Build a **CollisionState** (p. 245) from SDF data

Parameters

in	<i>_sdf</i>	SDF data to load a collision state from.
----	-------------	--

10.26.2.4 virtual gazebo::physics::CollisionState::~~CollisionState () [virtual]

Destructor.

10.26.3 Member Function Documentation

10.26.3.1 void gazebo::physics::CollisionState::FillSDF (sdf::ElementPtr *_sdf*)

Populate a state SDF element with data from the object.

Parameters

out	<i>_sdf</i>	SDF element to populate.
-----	-------------	--------------------------

10.26.3.2 `const math::Pose& gazebo::physics::CollisionState::GetPose () const`

Get the **Collision** (p. 235) pose.

Returns

The pose of the **CollisionState** (p. 245)

10.26.3.3 `bool gazebo::physics::CollisionState::IsZero () const`

Return true if the values in the state are zero.

Returns

True if the values in the state are zero.

10.26.3.4 `virtual void gazebo::physics::CollisionState::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Load **CollisionState** (p. 245) information from stored data in and SDF::Element

Parameters

<code>in</code>	<code>_elem</code>	Pointer to the SDF::Element containing state info.
-----------------	--------------------	--

Reimplemented from **gazebo::physics::State** (p. 1070).

10.26.3.5 `CollisionState gazebo::physics::CollisionState::operator+ (const CollisionState & _state) const`

Addition operator.

Parameters

<code>in</code>	<code>_pt</code>	A state to add.
-----------------	------------------	-----------------

Returns

The resulting state.

10.26.3.6 `CollisionState gazebo::physics::CollisionState::operator- (const CollisionState & _state) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_pt</code>	A state to subtract.
-----------------	------------------	----------------------

Returns

The resulting state.

10.26.3.7 CollisionState& gazebo::physics::CollisionState::operator= (const CollisionState & *_state*)

Assignment operator.

Parameters

<i>in</i>	<i>_state</i>	State (p. 1068) value
-----------	---------------	------------------------------

Returns

Reference to this

10.26.4 Friends And Related Function Documentation

10.26.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::physics::CollisionState & *_state*) [friend]

Stream insertion operator.

Parameters

<i>in</i>	<i>_out</i>	output stream
<i>in</i>	<i>_state</i>	Collision (p. 235) state to output

Returns

the stream

The documentation for this class was generated from the following file:

- **CollisionState.hh**

10.27 gazebo::common::Color Class Reference

Defines a color.

```
#include <common/common.hh>
```

Public Types

- typedef unsigned int **ABGR**
- typedef unsigned int **ARGB**
- typedef unsigned int **BGRA**
- typedef unsigned int **RGBA**

Public Member Functions

- **Color** ()
Constructor.
- **Color** (float _r, float _g, float _b, float _a=1.0)
Constructor.
- **Color** (const **Color** &_clr)
Copy Constructor.
- virtual ~**Color** ()
Destructor.
- **ABGR GetAsABGR** () const
Get as uint32 ABGR packed value.
- **ARGB GetAsARGB** () const
Get as uint32 ARGB packed value.
- **BGRA GetAsBGRA** () const
Get as uint32 BGRA packed value.
- **math::Vector3 GetAsHSV** () const
Get the color in HSV colorspace.
- **RGBA GetAsRGBA** () const
Get as uint32 RGBA packed value.
- **math::Vector3 GetAsYUV** () const
Get the color in YUV colorspace.
- bool **operator!=** (const **Color** &_pt) const
Inequality operator.
- const **Color operator*** (const **Color** &_pt) const
Multiplication operator.
- const **Color operator*** (const float &_v) const
Multiply all color components by _v.
- const **Color** & **operator*=** (const **Color** &_pt)
Multiplication equal operator.
- **Color operator+** (const **Color** &_pt) const
Addition operator (this + _pt)
- **Color operator+** (const float &_v) const
Add _v to all color components.
- const **Color** & **operator+=** (const **Color** &_pt)
Addition equal operator.
- **Color operator-** (const **Color** &_pt) const
Subtraction operator.
- **Color operator-** (const float &_v) const
Subtract _v from all color components.
- const **Color** & **operator-=** (const **Color** &_pt)
Subtraction equal operator.
- const **Color operator/** (const **Color** &_pt) const
Division operator.
- const **Color operator/** (const float &_v) const
Divide all color component by _v.
- const **Color** & **operator/=** (const **Color** &_pt)

Division equal operator.

- **Color & operator=** (const **Color** &_pt)

Equal operator.

- bool **operator==** (const **Color** &_pt) const

Equality operator.

- float **operator[]** (unsigned int _index)

Array index operator.

- void **Reset** ()

Reset the color to default values.

- void **Set** (float _r=1, float _g=1, float _b=1, float _a=1)

Set the contents of the vector.

- void **SetFromABGR** (const **ABGR** _v)

Set from uint32 ABGR packed value.

- void **SetFromARGB** (const **ARGB** _v)

Set from uint32 ARGB packed value.

- void **SetFromBGRA** (const **BGRA** _v)

Set from uint32 BGRA packed value.

- void **SetFromHSV** (float _h, float _s, float _v)

Set a color based on HSV values.

- void **SetFromRGBA** (const **RGBA** _v)

Set from uint32 RGBA packed value.

- void **SetFromYUV** (float _y, float _u, float _v)

Set from yuv.

Public Attributes

- float **a**
- float **b**
- float **g**
- float **r**

Static Public Attributes

- static const **Color Black**
(0, 0, 0)
- static const **Color Blue**
(0, 0, 1)
- static const **Color Green**
(0, 1, 0)
- static const **Color Purple**
(1, 0, 1)
- static const **Color Red**
(1, 0, 0)
- static const **Color White**
(1, 1, 1)
- static const **Color Yellow**
(1, 1, 0)

Friends

- `std::ostream & operator<<` (`std::ostream &_out`, `const Color &_pt`)
Stream insertion operator.
- `std::istream & operator>>` (`std::istream &_in`, `Color &_pt`)
Stream insertion operator.

10.27.1 Detailed Description

Defines a color.

10.27.2 Member Typedef Documentation

10.27.2.1 typedef unsigned int gazebo::common::Color::ABGR

10.27.2.2 typedef unsigned int gazebo::common::Color::ARGB

10.27.2.3 typedef unsigned int gazebo::common::Color::BGRA

10.27.2.4 typedef unsigned int gazebo::common::Color::RGBA

10.27.3 Constructor & Destructor Documentation

10.27.3.1 gazebo::common::Color::Color ()

Constructor.

10.27.3.2 gazebo::common::Color::Color (float *_r*, float *_g*, float *_b*, float *_a* = 1.0)

Constructor.

Parameters

in	<i>_r</i>	Red value (range 0 to 1)
in	<i>_g</i>	Green value (range 0 to 1)
in	<i>_b</i>	Blue value (range 0 to 1)
in	<i>_a</i>	Alpha value (0=transparent, 1=opaque)

10.27.3.3 gazebo::common::Color::Color (const Color & *_clr*)

Copy Constructor.

Parameters

in	<i>_clr</i>	Color (p. 249) to copy
----	-------------	-------------------------------

10.27.3.4 virtual gazebo::common::Color::~~Color () [virtual]

Destructor.

10.27.4 Member Function Documentation

10.27.4.1 **ABGR** gazebo::common::Color::GetAsABGR () const

Get as uint32 ABGR packed value.

Returns

the color

10.27.4.2 **ARGB** gazebo::common::Color::GetAsARGB () const

Get as uint32 ARGB packed value.

Returns

the color

10.27.4.3 **BGRA** gazebo::common::Color::GetAsBGRA () const

Get as uint32 BGRA packed value.

Returns

the color

10.27.4.4 **math::Vector3** gazebo::common::Color::GetAsHSV () const

Get the color in HSV colorspace.

Returns

HSV values in a **math::Vector3** (p. 1165) format

10.27.4.5 **RGBA** gazebo::common::Color::GetAsRGBA () const

Get as uint32 RGBA packed value.

Returns

the color

10.27.4.6 `math::Vector3 gazebo::common::Color::GetAsYUV () const`

Get the color in YUV colorspace.

Returns

the YUV color

10.27.4.7 `bool gazebo::common::Color::operator!=(const Color & _pt) const`

Inequality operator.

Parameters

<code>in</code>	<code>_pt</code>	The color to check for inequality
-----------------	------------------	-----------------------------------

Returns

True if the this color does not equal `_pt`

10.27.4.8 `const Color gazebo::common::Color::operator*(const Color & _pt) const`

Multiplication operator.

Parameters

<code>in</code>	<code>_pt</code>	The color to multiply by
-----------------	------------------	--------------------------

Returns

The resulting color

10.27.4.9 `const Color gazebo::common::Color::operator*(const float & _v) const`

Multiply all color components by `_v`.

Parameters

<code>in</code>	<code>_v</code>	The value to multiply by
-----------------	-----------------	--------------------------

Returns

The resulting color

10.27.4.10 `const Color& gazebo::common::Color::operator*=(const Color & _pt)`

Multiplication equal operator.

Parameters

<code>in</code>	<code>_pt</code>	The color to multiply by
-----------------	------------------	--------------------------

Returns

The resulting color

10.27.4.11 **Color** gazebo::common::Color::operator+ (const Color & _pt) const

Addition operator (this + _pt)

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 249) to add
-----------------	------------------	------------------------------

Returns

The resulting color

10.27.4.12 **Color** gazebo::common::Color::operator+ (const float & _v) const

Add _v to all color components.

Parameters

<code>in</code>	<code>_v</code>	Value to add to each color component
-----------------	-----------------	--------------------------------------

Returns

The resulting color

10.27.4.13 **const Color&** gazebo::common::Color::operator+= (const Color & _pt)

Addition equal operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 249) to add
-----------------	------------------	------------------------------

Returns

The resulting color

10.27.4.14 **Color** gazebo::common::Color::operator- (const Color & _pt) const

Subtraction operator.

Parameters

in	_pt	The color to subtract
----	-----	-----------------------

Returns

The resulting color

10.27.4.15 **Color** gazebo::common::Color::operator- (const float & _v) const

Subtract _v from all color components.

Parameters

in	_v	Value to subtract
----	----	-------------------

Returns

The resulting color

10.27.4.16 **const Color&** gazebo::common::Color::operator-= (const **Color** & _pt)

Subtraction equal operator.

Parameters

in	_pt	Color (p. 249) to subtract
----	-----	-----------------------------------

Returns

The resulting color

10.27.4.17 **const Color** gazebo::common::Color::operator/ (const **Color** & _pt) const

Division operator.

Parameters

in	_pt	Color (p. 249) to divide by
----	-----	------------------------------------

Returns

The resulting color

10.27.4.18 **const Color** gazebo::common::Color::operator/ (const float & _v) const

Divide all color component by _v.

Parameters

in	_v	The value to divide by
----	----	------------------------

Returns

The resulting color

10.27.4.19 `const Color& gazebo::common::Color::operator/= (const Color & _pt)`

Division equal operator.

Parameters

in	_pt	Color (p. 249) to divide by
----	-----	------------------------------------

Returns

The resulting color

10.27.4.20 `Color& gazebo::common::Color::operator= (const Color & _pt)`

Equal operator.

Parameters

in	_pt	Color (p. 249) to copy
----	-----	-------------------------------

Returns

Reference to this color

10.27.4.21 `bool gazebo::common::Color::operator==(const Color & _pt) const`

Equality operator.

Parameters

in	_pt	The color to check for equality
----	-----	---------------------------------

Returns

True if the this color equals _pt

10.27.4.22 `float gazebo::common::Color::operator[] (unsigned int _index)`

Array index operator.

Parameters

in	<code>_index</code>	Color (p. 249) component index(0=red, 1=green, 2=blue)
----	---------------------	---

Returns

r, g, b, or a when `_index` is 0, 1, 2 or 3

10.27.4.23 `void gazebo::common::Color::Reset ()`

Reset the color to default values.

10.27.4.24 `void gazebo::common::Color::Set (float _r = 1, float _g = 1, float _b = 1, float _a = 1)`

Set the contents of the vector.

Parameters

in	<code>_r</code>	Red value (range 0 to 1)
in	<code>_g</code>	Green value (range 0 to 1)
in	<code>_b</code>	Blue value (range 0 to 1)
in	<code>_a</code>	Alpha value (0=transparent, 1=opaque)

10.27.4.25 `void gazebo::common::Color::SetFromABGR (const ABGR _v)`

Set from uint32 ABGR packed value.

Parameters

in	<code>_v</code>	the new color
----	-----------------	---------------

10.27.4.26 `void gazebo::common::Color::SetFromARGB (const ARGB _v)`

Set from uint32 ARGB packed value.

Parameters

in	<code>_v</code>	the new color
----	-----------------	---------------

10.27.4.27 `void gazebo::common::Color::SetFromBGRA (const BGRA _v)`

Set from uint32 BGRA packed value.

Parameters

in	<code>_v</code>	the new color
----	-----------------	---------------

10.27.4.28 void gazebo::common::Color::SetFromHSV (float *_h*, float *_s*, float *_v*)

Set a color based on HSV values.

Parameters

in	<i>_h</i>	Hue(0..360)
in	<i>_s</i>	Saturation(0..1)
in	<i>_v</i>	Value(0..1)

10.27.4.29 void gazebo::common::Color::SetFromRGBA (const RGBA *_v*)

Set from uint32 RGBA packed value.

Parameters

in	<i>_v</i>	the new color
----	-----------	---------------

10.27.4.30 void gazebo::common::Color::SetFromYUV (float *_y*, float *_u*, float *_v*)

Set from yuv.

Parameters

in	<i>_y</i>	value
in	<i>_u</i>	value
in	<i>_v</i>	value

10.27.5 Friends And Related Function Documentation

10.27.5.1 std::ostream& operator<< (std::ostream & *_out*, const Color & *_pt*) [friend]

Stream insertion operator.

Parameters

in	<i>_out</i>	the output stream
in	<i>_pt</i>	the color

Returns

the output stream

10.27.5.2 std::istream& operator>> (std::istream & *_in*, Color & *_pt*) [friend]

Stream insertion operator.

Parameters

in	<i>_in</i>	the input stream
in	<i>_pt</i>	

10.27.6 Member Data Documentation

10.27.6.1 float gazebo::common::Color::a

10.27.6.2 float gazebo::common::Color::b

10.27.6.3 const Color gazebo::common::Color::Black [static]

(0, 0, 0)

10.27.6.4 const Color gazebo::common::Color::Blue [static]

(0, 0, 1)

10.27.6.5 float gazebo::common::Color::g

10.27.6.6 const Color gazebo::common::Color::Green [static]

(0, 1, 0)

10.27.6.7 const Color gazebo::common::Color::Purple [static]

(1, 0, 1)

10.27.6.8 float gazebo::common::Color::r

10.27.6.9 const Color gazebo::common::Color::Red [static]

(1, 0, 0)

10.27.6.10 const Color gazebo::common::Color::White [static]

(1, 1, 1)

10.27.6.11 const Color gazebo::common::Color::Yellow [static]

(1, 1, 0)

The documentation for this class was generated from the following file:

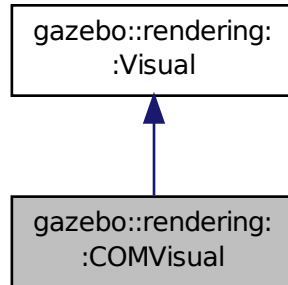
- **Color.hh**

10.28 gazebo::rendering::COMVisual Class Reference

Basic Center of Mass visualization.

```
#include <rendering/rendering.hh>
```


Inheritance diagram for gazebo::rendering::COMVisual:



Public Member Functions

- **COMVisual** (const std::string &_name, **VisualPtr** _vis)
Constructor.
- virtual ~**COMVisual** ()
Destructor.
- virtual void **Load** (sdf::ElementPtr _elem)
*Load the **Visual** (p. 1196) from an SDF pointer.*
- virtual void **Load** (ConstLinkPtr &_msg)
Load from a message.

Additional Inherited Members

10.28.1 Detailed Description

Basic Center of Mass visualization.

10.28.2 Constructor & Destructor Documentation

10.28.2.1 gazebo::rendering::COMVisual::COMVisual (const std::string & _name, **VisualPtr** _vis)

Constructor.

Parameters

in	<code>_name</code>	Name of the Visual (p. 1196)
in	<code>_vis</code>	Parent Visual (p. 1196)

10.28.2.2 `virtual gazebo::rendering::COMVisual::~~COMVisual () [virtual]`

Destructor.

10.28.3 Member Function Documentation

10.28.3.1 `virtual void gazebo::rendering::COMVisual::Load (sdf::ElementPtr _elem) [virtual]`

Load the **Visual** (p. 1196) from an SDF pointer.

Parameters

in	_elem	SDF Element pointer
----	-------	---------------------

Reimplemented from `gazebo::rendering::Visual` (p. 1210).

10.28.3.2 `virtual void gazebo::rendering::COMVisual::Load (ConstLinkPtr & _msg) [virtual]`

Load from a message.

Parameters

in	_msg	Pointer to the message
----	------	------------------------

The documentation for this class was generated from the following file:

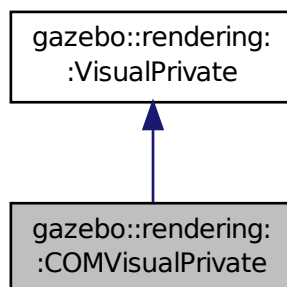
- `COMVisual.hh`

10.29 gazebo::rendering::COMVisualPrivate Class Reference

Private data for the COM **Visual** (p. 1196) class.

```
#include <COMVisualPrivate.hh>
```

Inheritance diagram for `gazebo::rendering::COMVisualPrivate`:



Public Attributes

- `Ogre::SceneNode * boxNode`
Box that make the cross marking the center of mass.
- `DynamicLines * crossLines`
Lines that make the cross marking the center of mass.

Additional Inherited Members

10.29.1 Detailed Description

Private data for the COM **Visual** (p. 1196) class.

10.29.2 Member Data Documentation

10.29.2.1 `Ogre::SceneNode* gazebo::rendering::COMVisualPrivate::boxNode`

Box that make the cross marking the center of mass.

10.29.2.2 `DynamicLines* gazebo::rendering::COMVisualPrivate::crossLines`

Lines that make the cross marking the center of mass.

The documentation for this class was generated from the following file:

- **COMVisualPrivate.hh**

10.30 gazebo::event::Connection Class Reference

A class that encapsulates a connection.

```
#include <Event.hh>
```

Public Member Functions

- **Connection** ()
Constructor.
- **Connection** (**Event** *_e, int _i)
Constructor.
- **~Connection** ()
Destructor.
- int **GetId** () const
Get the id of this connection.

10.30.1 Detailed Description

A class that encapsulates a connection.

10.30.2 Constructor & Destructor Documentation

10.30.2.1 gazebo::event::Connection::Connection ()

Constructor.

10.30.2.2 gazebo::event::Connection::Connection (Event * _e, int _i)

Constructor.

Parameters

in	<code>_e</code>	Event (p. 416) pointer to connect with.
in	<code>_i</code>	Unique id.

10.30.2.3 gazebo::event::Connection::~~Connection ()

Destructor.

10.30.3 Member Function Documentation

10.30.3.1 int gazebo::event::Connection::GetId () const

Get the id of this connection.

Returns

The id of this connection.

The documentation for this class was generated from the following file:

- **Event.hh**

10.31 gazebo::transport::Connection Class Reference

Single TCP/IP connection manager.

```
#include <transport/transport.hh>
```

Public Types

- typedef boost::function< void(const **ConnectionPtr** &)> **AcceptCallback**
The signature of a connection accept callback.
- typedef boost::function< void(const std::string &_data)> **ReadCallback**
The signature of a connection read callback.

Public Member Functions

- **Connection** ()
Constructor.
- virtual **~Connection** ()
Destructor.
- template<typename Handler >
void **AsyncRead** (Handler _handler)
Perform an asynchronous read param[in] _handler Callback to invoke on received data.
- void **Cancel** ()
Cancel all async operations on an open socket.
- bool **Connect** (const std::string &_host, unsigned int _port)
Connect to a remote host.
- **event::ConnectionPtr ConnectToShutdown** (boost::function< void()> _subscriber)
Register a function to be called when the connection is shut down.
- void **DisconnectShutdown** (**event::ConnectionPtr** _subscriber)
Unregister a function to be called when the connection is shut down.
- void **EnqueueMsg** (const std::string &_buffer, boost::function< void(uint32_t)> _cb, uint32_t _id, bool _force=false)
Write data to the socket.
- void **EnqueueMsg** (const std::string &_buffer, bool _force=false)
Write data to the socket.
- unsigned int **GetId** () const
Get the ID of the connection.
- std::string **GetIPWhiteList** () const
Get the IP white list, from GAZEBO_IP_WHITE_LIST environment variable.
- std::string **GetLocalAddress** () const
Get the local address of this connection.
- unsigned int **GetLocalPort** () const
Get the port of this connection.
- std::string **GetLocalURI** () const
Get the local URI.
- std::string **GetRemoteAddress** () const
Get the remote address.
- std::string **GetRemoteHostname** () const
Get the remote hostname.
- unsigned int **GetRemotePort** () const
Get the remote port number.
- std::string **GetRemoteURI** () const
Get the remote URI.
- bool **IsOpen** () const
Is the connection open?
- void **Listen** (unsigned int _port, const **AcceptCallback** &_acceptCB)
Start a server that listens on a port.
- void **ProcessWriteQueue** (bool _blocking=false)
Handle on-write callbacks.
- bool **Read** (std::string &_data)
Read data from the socket.

- void **Shutdown** ()
Shutdown the socket.
- void **StartRead** (const **ReadCallback** &_cb)
Start a thread that reads from the connection and passes new message to the ReadCallback.
- void **StopRead** ()
Stop the read loop.

Static Public Member Functions

- static std::string **GetLocalHostname** ()
Get the local hostname.
- static bool **ValidateIP** (const std::string &_ip)
Return true if the _ip is a valid.

10.31.1 Detailed Description

Single TCP/IP connection manager.

10.31.2 Member Typedef Documentation

10.31.2.1 typedef boost::function<void(const **ConnectionPtr**&)> **gazebo::transport::Connection::AcceptCallback**

The signature of a connection accept callback.

10.31.2.2 typedef boost::function<void(const std::string &_data)> **gazebo::transport::Connection::ReadCallback**

The signature of a connection read callback.

10.31.3 Constructor & Destructor Documentation

10.31.3.1 **gazebo::transport::Connection::Connection** ()

Constructor.

10.31.3.2 virtual **gazebo::transport::Connection::~~Connection** () [virtual]

Destructor.

10.31.4 Member Function Documentation

10.31.4.1 template<typename **Handler** > void **gazebo::transport::Connection::AsyncRead** (**Handler** *_handler*) [inline]

Perform an asynchronous read param[in] *_handler* Callback to invoke on received data.

References `gzerr`, and `HEADER_LENGTH`.

10.31.4.2 void gazebo::transport::Connection::Cancel ()

Cancel all async operations on an open socket.

10.31.4.3 bool gazebo::transport::Connection::Connect (const std::string & *_host*, unsigned int *_port*)

Connect to a remote host.

Parameters

in	<i>_host</i>	The host to connect to
in	<i>_port</i>	The port to connect to

Returns

true if connection succeeded, false otherwise

10.31.4.4 event::ConnectionPtr gazebo::transport::Connection::ConnectToShutdown (boost::function< void()> *_subscriber*)
[inline]

Register a function to be called when the connection is shut down.

Parameters

in	<i>_subscriber</i>	Function to be called
----	--------------------	-----------------------

Returns

Handle that can be used to unregister the function

References gazebo::shutdown().

10.31.4.5 void gazebo::transport::Connection::DisconnectShutdown (event::ConnectionPtr *_subscriber*) [inline]

Unregister a function to be called when the connection is shut down.

Parameters

in	<i>_subscriber</i>	Handle previously returned by ConnectToShutdown() (p. 267)
----	--------------------	---

References gazebo::shutdown().

10.31.4.6 void gazebo::transport::Connection::EnqueueMsg (const std::string & *_buffer*, boost::function< void(uint32_t)> *_cb*,
uint32_t *_id*, bool *_force* = false)

Write data to the socket.

Parameters

in	<i>_buffer</i>	Data to write
in	<i>_force</i>	If true, block until the data has been written to the socket, otherwise just enqueue the data for asynchronous write

<code>in</code>	<code>_cb</code>	If non-null, callback to be invoked after transmission is complete.
<code>in</code>	<code>_id</code>	ID associated with the message data.

10.31.4.7 `void gazebo::transport::Connection::EnqueueMsg (const std::string & _buffer, bool _force = false)`

Write data to the socket.

Parameters

<code>in</code>	<code>_buffer</code>	Data to write
<code>in</code>	<code>_force</code>	If true, block until the data has been written to the socket, otherwise just enqueue the data for asynchronous write

10.31.4.8 `unsigned int gazebo::transport::Connection::GetId () const`

Get the ID of the connection.

Returns

The connection's unique ID.

10.31.4.9 `std::string gazebo::transport::Connection::GetIPWhiteList () const`

Get the IP white list, from GAZEBO_IP_WHITE_LIST environment variable.

Returns

GAZEBO_IP_WHITE_LIST

10.31.4.10 `std::string gazebo::transport::Connection::GetLocalAddress () const`

Get the local address of this connection.

Returns

The local address

10.31.4.11 `static std::string gazebo::transport::Connection::GetLocalHostname () [static]`

Get the local hostname.

Returns

The local hostname

10.31.4.12 `unsigned int gazebo::transport::Connection::GetLocalPort () const`

Get the port of this connection.

Returns

The local port

10.31.4.13 `std::string gazebo::transport::Connection::GetLocalURI () const`

Get the local URI.

Returns

The local URI

10.31.4.14 `std::string gazebo::transport::Connection::GetRemoteAddress () const`

Get the remote address.

Returns

The remote address

10.31.4.15 `std::string gazebo::transport::Connection::GetRemoteHostname () const`

Get the remote hostname.

Returns

The remote hostname

10.31.4.16 `unsigned int gazebo::transport::Connection::GetRemotePort () const`

Get the remote port number.

Returns

The remote port

10.31.4.17 `std::string gazebo::transport::Connection::GetRemoteURI () const`

Get the remote URI.

Returns

The remote URI

10.31.4.18 `bool gazebo::transport::Connection::IsOpen () const`

Is the connection open?

Returns

true if the connection is open; false otherwise

10.31.4.19 `void gazebo::transport::Connection::Listen (unsigned int _port, const AcceptCallback & _acceptCB)`

Start a server that listens on a port.

Parameters

<code>in</code>	<code><i>_port</i></code>	The port to listen on
<code>in</code>	<code><i>_acceptCB</i></code>	The callback to invoke when a new connection has been accepted

10.31.4.20 `void gazebo::transport::Connection::ProcessWriteQueue (bool _blocking = false)`

Handle on-write callbacks.

10.31.4.21 `bool gazebo::transport::Connection::Read (std::string & _data)`

Read data from the socket.

Parameters

<code>out</code>	<code><i>_data</i></code>	Destination for data that is read
------------------	---------------------------	-----------------------------------

Returns

true if data was successfully read, false otherwise

10.31.4.22 `void gazebo::transport::Connection::Shutdown ()`

Shutdown the socket.

10.31.4.23 `void gazebo::transport::Connection::StartRead (const ReadCallback & _cb)`

Start a thread that reads from the connection and passes new message to the ReadCallback.

Parameters

<code>in</code>	<code><i>_cb</i></code>	The callback to invoke when a new message is received
-----------------	-------------------------	---

10.31.4.24 `void gazebo::transport::Connection::StopRead ()`

Stop the read loop.

10.31.4.25 static bool gazebo::transport::Connection::ValidateIP (const std::string & _ip) [static]

Return true if the _ip is a valid.

Parameters

in	_ip	Dotted quad to validate.
----	-----	--------------------------

Returns

True if the _ip is a valid.

The documentation for this class was generated from the following file:

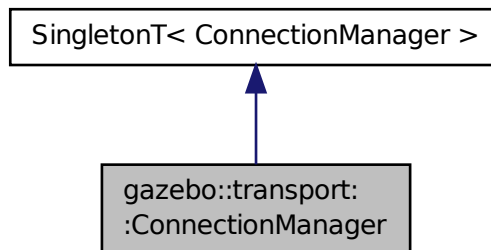
- **Connection.hh**

10.32 gazebo::transport::ConnectionManager Class Reference

Manager of connections.

```
#include <transport/transport.hh>
```

Inheritance diagram for gazebo::transport::ConnectionManager:



Public Member Functions

- void **Advertise** (const std::string &_topic, const std::string &_msgType)
Advertise a topic.
- **ConnectionPtr ConnectToRemoteHost** (const std::string &_host, unsigned int _port)
Connect to a remote server.
- void **Fini** ()
Finalize the connection manager.
- void **GetAllPublishers** (std::list< msgs::Publish > &_publishers)
Explicitly update the publisher list.
- void **GetTopicNamespaces** (std::list< std::string > &_namespaces)

- Get all the topic namespaces.*

 - bool **Init** (const std::string &_masterHost, unsigned int _masterPort, uint32_t _timeoutIterations=30)
Initialize the connection manager.
 - bool **IsRunning** () const
Is the manager running?
 - void **RegisterTopicNamespace** (const std::string &_name)
Register a new topic namespace.
 - void **RemoveConnection** (**ConnectionPtr** &_conn)
Remove a connection from the manager.
 - void **Run** ()
Run the connection manager loop.
 - void **Stop** ()
Stop the connecton manager.
 - void **Subscribe** (const std::string &_topic, const std::string &_msgType, bool _latching)
Subscribe to a topic.
 - void **TriggerUpdate** ()
Inform the connection manager that it needs an update.
 - void **Unadvertise** (const std::string &_topic)
Unadvertise a topic.
 - void **Unsubscribe** (const msgs::Subscribe &_sub)
Unsubscribe from a topic.
 - void **Unsubscribe** (const std::string &_topic, const std::string &_msgType)
Unsubscribe from a topic.

Protected Attributes

- std::vector< **event::ConnectionPtr** > **eventConnections**

Additional Inherited Members

10.32.1 Detailed Description

Manager of connections.

10.32.2 Member Function Documentation

10.32.2.1 void gazebo::transport::ConnectionManager::Advertise (const std::string & *_topic*, const std::string & *_msgType*)

Advertise a topic.

Parameters

in	<i>_topic</i>	The topic to advertise
in	<i>_msgType</i>	The type of the topic

10.32.2.2 `ConnectionPtr` gazebo::transport::ConnectionManager::ConnectToRemoteHost (const std::string & *_host*, unsigned int *_port*)

Connect to a remote server.

Parameters

in	<i>_host</i>	Host to connect to
in	<i>_port</i>	Port to connect to

Returns

Pointer to the connection; can be null (if connection failed)

10.32.2.3 void gazebo::transport::ConnectionManager::Fini ()

Finalize the connection manager.

10.32.2.4 void gazebo::transport::ConnectionManager::GetAllPublishers (std::list< msgs::Publish > & *_publishers*)

Explicitly update the publisher list.

Parameters

out	<i>_publishers</i>	The updated list of publishers is written here
-----	--------------------	--

10.32.2.5 void gazebo::transport::ConnectionManager::GetTopicNamespaces (std::list< std::string > & *_namespaces*)

Get all the topic namespaces.

Parameters

out	<i>_namespaces</i>	The list of namespace is written here
-----	--------------------	---------------------------------------

10.32.2.6 bool gazebo::transport::ConnectionManager::Init (const std::string & *_masterHost*, unsigned int *_masterPort*, uint32_t *_timeoutIterations* = 30)

Initialize the connection manager.

Parameters

in	<i>_masterHost</i>	Host where the master is running.
in	<i>_masterPort</i>	Port where the master is running.
in	<i>_timeoutIterations</i>	Number of times to wait for a connection to master.

Returns

true if initialization succeeded, false otherwise

10.32.2.7 `bool gazebo::transport::ConnectionManager::IsRunning () const`

Is the manager running?

Returns

true if running, false otherwise

10.32.2.8 `void gazebo::transport::ConnectionManager::RegisterTopicNamespace (const std::string & _name)`

Register a new topic namespace.

Parameters

in	<code>_name</code>	The name of the topic namespace to be registered
----	--------------------	--

10.32.2.9 `void gazebo::transport::ConnectionManager::RemoveConnection (ConnectionPtr & _conn)`

Remove a connection from the manager.

Parameters

in	<code>_conn</code>	The connection to be removed
----	--------------------	------------------------------

10.32.2.10 `void gazebo::transport::ConnectionManager::Run ()`

Run the connection manager loop.

Does not return until stopped.

10.32.2.11 `void gazebo::transport::ConnectionManager::Stop ()`

Stop the connection manager.

10.32.2.12 `void gazebo::transport::ConnectionManager::Subscribe (const std::string & _topic, const std::string & _msgType, bool _latching)`

Subscribe to a topic.

Parameters

in	<code>_topic</code>	The topic to subscribe to.
in	<code>_msgType</code>	The type of the topic.
in	<code>_latching</code>	If true, latch the latest incoming message; otherwise don't.

10.32.2.13 `void gazebo::transport::ConnectionManager::TriggerUpdate ()`

Inform the connection manager that it needs an update.

10.32.2.14 void gazebo::transport::ConnectionManager::Unadvertise (const std::string & *_topic*)

Unadvertise a topic.

Parameters

in	<i>_topic</i>	The topic to unadvertise
----	---------------	--------------------------

10.32.2.15 void gazebo::transport::ConnectionManager::Unsubscribe (const msgs::Subscribe & *_sub*)

Unsubscribe from a topic.

Parameters

in	<i>_sub</i>	A subscription object
----	-------------	-----------------------

10.32.2.16 void gazebo::transport::ConnectionManager::Unsubscribe (const std::string & *_topic*, const std::string & *_msgType*)

Unsubscribe from a topic.

Parameters

in	<i>_topic</i>	The topic to unsubscribe from
in	<i>_msgType</i>	The type of the topic

10.32.3 Member Data Documentation

10.32.3.1 std::vector<event::ConnectionPtr> gazebo::transport::ConnectionManager::eventConnections [protected]

The documentation for this class was generated from the following file:

- **ConnectionManager.hh**

10.33 gazebo::event::ConnectionPrivate Class Reference

```
#include <Event.hh>
```

Public Member Functions

- **ConnectionPrivate** ()
Constructor.
- **ConnectionPrivate** (Event * *_e*, int *_i*)
Constructor.

Public Attributes

- **common::Time** creationTime

set during the constructor

- **Event * event**

the event for this connection

- **int id**

the id set in the constructor

10.33.1 Constructor & Destructor Documentation

10.33.1.1 gazebo::event::ConnectionPrivate::ConnectionPrivate ()

Constructor.

10.33.1.2 gazebo::event::ConnectionPrivate::ConnectionPrivate (Event * _e, int _i)

Constructor.

Parameters

<code>in</code>	<code>_e</code>	Event (p. 416) pointer to connect with
<code>in</code>	<code>_i</code>	Unique id

10.33.2 Member Data Documentation

10.33.2.1 common::Time gazebo::event::ConnectionPrivate::creationTime

set during the constructor

10.33.2.2 Event* gazebo::event::ConnectionPrivate::event

the event for this connection

10.33.2.3 int gazebo::event::ConnectionPrivate::id

the id set in the constructor

The documentation for this class was generated from the following file:

- **Event.hh**

10.34 gazebo::transport::ConnectionReadTask Class Reference

```
#include <Connection.hh>
```

Public Member Functions

- **ConnectionReadTask** (boost::function< void(const std::string &)> _func, const std::string &_data)

Constructor.

- `tbb::task * execute ()`
Overridden function from `tbb::task` that executes the data callback.

10.34.1 Detailed Description

A task instance that is created when data is read from a socket and used by TBB

10.34.2 Constructor & Destructor Documentation

- 10.34.2.1 `gazebo::transport::ConnectionReadTask::ConnectionReadTask (boost::function< void(const std::string &);> _func, const std::string & _data) [inline]`

Constructor.

Parameters

	<code>_in]</code>	<code>_func</code> Boost function pointer, which is the function that receives the data.
<code>in</code>	<code>_data</code>	Data to send to the boost function pointer.

10.34.3 Member Function Documentation

- 10.34.3.1 `tbb::task* gazebo::transport::ConnectionReadTask::execute () [inline]`

Overridden function from `tbb::task` that executes the data callback.

References NULL.

The documentation for this class was generated from the following file:

- **Connection.hh**

10.35 gazebo::common::Console Class Reference

Container for loggers, and global logging options (such as verbose vs.

```
#include <common/common.hh>
```

Static Public Member Functions

- static bool **GetQuiet** ()
Get whether quiet output is set.
- static void **SetQuiet** (bool _q)
Set quiet output.

Static Public Attributes

- static **Logger dbg**
Global instance of the debug logger.
- static **Logger err**

Global instance of the error logger.

- static **FileLogger log**

Global instance of the file logger.

- static **Logger msg**

Global instance of the message logger.

- static **Logger warn**

Global instance of the warning logger.

10.35.1 Detailed Description

Container for loggers, and global logging options (such as verbose vs. quiet output).

10.35.2 Member Function Documentation

10.35.2.1 static bool gazebo::common::Console::GetQuiet () [static]

Get whether quiet output is set.

Returns

True to if quiet output is set.

10.35.2.2 static void gazebo::common::Console::SetQuiet (bool _q) [static]

Set quiet output.

Parameters

in	<i>q</i>	True to prevent warning.
----	----------	--------------------------

10.35.3 Member Data Documentation

10.35.3.1 Logger gazebo::common::Console::dbg [static]

Global instance of the debug logger.

10.35.3.2 Logger gazebo::common::Console::err [static]

Global instance of the error logger.

10.35.3.3 FileLogger gazebo::common::Console::log [static]

Global instance of the file logger.

10.35.3.4 Logger gazebo::common::Console::msg [static]

Global instance of the message logger.

10.35.3.5 Logger gazebo::common::Console::warn [static]

Global instance of the warning logger.

The documentation for this class was generated from the following file:

- **Console.hh**

10.36 gazebo::physics::Contact Class Reference

A contact between two collisions.

```
#include <physics/physics.hh>
```

Public Member Functions

- **Contact** ()
Constructor.
- **Contact** (const **Contact** &_contact)
Copy constructor.
- virtual ~**Contact** ()
Destructor.
- std::string **DebugString** () const
Produce a debug string.
- void **FillMsg** (msgs::Contact &_msg) const
Populate a msgs::Contact with data from this.
- **Contact** & **operator=** (const **Contact** &_contact)
Operator =.
- **Contact** & **operator=** (const msgs::Contact &_contact)
Operator =.
- void **Reset** ()
Reset to default values.

Public Attributes

- **Collision** * **collision1**
Pointer to the first collision object.
- **Collision** * **collision2**
Pointer to the second collision object.
- int **count**
Length of all the arrays.
- double **depths** [32]
Array of contact depths.

- **math::Vector3 normals** [32]
Array of force normals.
- **math::Vector3 positions** [32]
Array of force positions.
- **common::Time time**
Time at which the contact occurred.
- **WorldPtr world**
World (p. 1239) in which the contact occurred.
- **JointWrench wrench** [32]
Array of forces for the contact.

10.36.1 Detailed Description

A contact between two collisions.

Each contact can consist of a number of contact points

10.36.2 Constructor & Destructor Documentation

10.36.2.1 gazebo::physics::Contact::Contact ()

Constructor.

10.36.2.2 gazebo::physics::Contact::Contact (const Contact & *_contact*)

Copy constructor.

Parameters

in	<i>_contact</i>	Contact (p. 279) to copy.
----	-----------------	----------------------------------

10.36.2.3 virtual gazebo::physics::Contact::~~Contact () [virtual]

Destructor.

10.36.3 Member Function Documentation

10.36.3.1 std::string gazebo::physics::Contact::DebugString () const

Produce a debug string.

Returns

A string that contains the values of the contact.

10.36.3.2 void gazebo::physics::Contact::FillMsg (msgs::Contact & *_msg*) const

Populate a msgs::Contact with data from this.

Parameters

out	<code>_msg</code>	Contact (p. 279) message the will hold the data.
-----	-------------------	---

10.36.3.3 **Contact&** gazebo::physics::Contact::operator= (const **Contact** & *.contact*)

Operator =.

Parameters

in	<code>_contact</code>	Contact (p. 279) to copy.
----	-----------------------	----------------------------------

Returns

Reference to this contact

10.36.3.4 **Contact&** gazebo::physics::Contact::operator= (const msgs::Contact & *.contact*)

Operator =.

Parameters

in	<code>_contact</code>	msgs::Contact to copy.
----	-----------------------	------------------------

Returns

Reference to this contact

10.36.3.5 void gazebo::physics::Contact::Reset ()

Reset to default values.

10.36.4 Member Data Documentation

10.36.4.1 **Collision*** gazebo::physics::Contact::collision1

Pointer to the first collision object.

10.36.4.2 **Collision*** gazebo::physics::Contact::collision2

Pointer to the second collision object.

10.36.4.3 int gazebo::physics::Contact::count

Length of all the arrays.

10.36.4.4 `double gazebo::physics::Contact::depths[32]`

Array of contact depths.

10.36.4.5 `math::Vector3 gazebo::physics::Contact::normals[32]`

Array of force normals.

10.36.4.6 `math::Vector3 gazebo::physics::Contact::positions[32]`

Array of force positions.

10.36.4.7 `common::Time gazebo::physics::Contact::time`

Time at which the contact occurred.

10.36.4.8 `WorldPtr gazebo::physics::Contact::world`

World (p. 1239) in which the contact occurred.

10.36.4.9 `JointWrench gazebo::physics::Contact::wrench[32]`

Array of forces for the contact.

All forces and torques are relative to the center of mass of the respective links that the collision elements are attached to.

The documentation for this class was generated from the following file:

- **Contact.hh**

10.37 gazebo::physics::ContactManager Class Reference

Aggregates all the contact information generated by the collision detection engine.

```
#include <physics/physics.hh>
```

Public Member Functions

- **ContactManager** ()
Constructor.
- virtual **~ContactManager** ()
Destructor.
- void **Clear** ()
Clear all stored contacts.
- std::string **CreateFilter** (const std::string &_topic, const std::vector< std::string > &_collisions)
Create a filter for contacts.
- std::string **CreateFilter** (const std::string &_topic, const std::string &_collision)
Create a filter for contacts.

- std::string **CreateFilter** (const std::string &_name, const std::map< std::string, **physics::CollisionPtr** > &_collisions)

Create a filter for contacts.
- **Contact * GetContact** (unsigned int _index) const

Get a single contact by index.
- unsigned int **GetContactCount** () const

Return the number of valid contacts.
- const std::vector< **Contact * > & GetContacts** () const

Get all the contacts.
- unsigned int **GetFilterCount** ()

Get the number of filters in the contact manager.
- bool **HasFilter** (const std::string &_name)

Check if a filter with the specified name exists.
- void **Init** (**WorldPtr** _world)

*Initialize the **ContactManager** (p. 282).*
- **Contact * NewContact** (**Collision * _collision1**, **Collision * _collision2**, const **common::Time** &_time)

Add a new contact.
- void **PublishContacts** ()

Publish all contacts in a msgs::Contacts message.
- void **RemoveFilter** (const std::string &_name)

Remove a contacts filter and the associated custom publisher param[in]_name Filter name.
- void **ResetCount** ()

Set the contact count to zero.

10.37.1 Detailed Description

Aggregates all the contact information generated by the collision detection engine.

10.37.2 Constructor & Destructor Documentation

10.37.2.1 gazebo::physics::ContactManager::ContactManager ()

Constructor.

10.37.2.2 virtual gazebo::physics::ContactManager::~~ContactManager () [virtual]

Destructor.

10.37.3 Member Function Documentation

10.37.3.1 void gazebo::physics::ContactManager::Clear ()

Clear all stored contacts.

10.37.3.2 `std::string gazebo::physics::ContactManager::CreateFilter (const std::string & _topic, const std::vector< std::string > & _collisions)`

Create a filter for contacts.

A new publisher will be created that publishes contacts associated to the input collisions. param[in] `_name` Filter name.
param[in] `_collisions` A list of collision names used for filtering.

Returns

New topic where filtered messages will be published to.

10.37.3.3 `std::string gazebo::physics::ContactManager::CreateFilter (const std::string & _topic, const std::string & _collision)`

Create a filter for contacts.

A new publisher will be created that publishes contacts associated to the input collision. param[in] `_name` Filter name.
param[in] `_collision` A collision name used for filtering.

Returns

New topic where filtered messages will be published to.

10.37.3.4 `std::string gazebo::physics::ContactManager::CreateFilter (const std::string & _name, const std::map< std::string, physics::CollisionPtr > & _collisions)`

Create a filter for contacts.

A new publisher will be created that publishes contacts associated to the input collision. param[in] `_name` Filter name.
param[in] `_collisions` A map of collision name to collision object.

Returns

New topic where filtered messages will be published to.

10.37.3.5 `Contact* gazebo::physics::ContactManager::GetContact (unsigned int _index) const`

Get a single contact by index.

The index must be between 0 and `ContactManager::GetContactCount` (p. 284).

Parameters

<code>in</code>	<code>_index</code>	Index of the <code>Contact</code> (p. 279) to return.
-----------------	---------------------	---

Returns

Pointer to a contact, NULL If index is invalid.

10.37.3.6 `unsigned int gazebo::physics::ContactManager::GetContactCount () const`

Return the number of valid contacts.

10.37.3.7 `const std::vector<Contact *>& gazebo::physics::ContactManager::GetContacts () const`

Get all the contacts.

The return vector may have invalid contacts. Only use contents of the vector between 0 and **ContactManager::GetContactCount** (p. 284)

Returns

Vector of contact pointers.

10.37.3.8 `unsigned int gazebo::physics::ContactManager::GetFilterCount ()`

Get the number of filters in the contact manager.

return Number of filters

10.37.3.9 `bool gazebo::physics::ContactManager::HasFilter (const std::string & _name)`

Check if a filter with the specified name exists.

param[in] *_name* Name of filter. return True if the filter exists.

10.37.3.10 `void gazebo::physics::ContactManager::Init (WorldPtr _world)`

Initialize the **ContactManager** (p. 282).

This is required in order to publish contact messages via the **ContactManager::PublishContacts** (p. 285) method.

Parameters

<i>in</i>	<i>_world</i>	Pointer to the world that is initializing the contact manager.
-----------	---------------	--

10.37.3.11 `Contact* gazebo::physics::ContactManager::NewContact (Collision * _collision1, Collision * _collision2, const common::Time & _time)`

Add a new contact.

Normally this is only used by a Physics/Collision engine when a new contact is generated. All other users should just make use of the accessor functions.

If no one is listening, then the return value will be NULL. This is a signal to the Physics engine that it can skip the extra processing necessary to get back contact information.

Returns

The new contact. The physics engine should populate the contact's parameters. NULL will be returned if there are no subscribers to the contact topic.

10.37.3.12 `void gazebo::physics::ContactManager::PublishContacts ()`

Publish all contacts in a `msgs::Contacts` message.

10.37.3.13 void gazebo::physics::ContactManager::RemoveFilter (const std::string & *_name*)

Remove a contacts filter and the associated custom publisher param[in] *_name* Filter name.

10.37.3.14 void gazebo::physics::ContactManager::ResetCount ()

Set the contact count to zero.

The documentation for this class was generated from the following file:

- **ContactManager.hh**

10.38 gazebo::rendering::ContactVisualPrivate::ContactPoint Class Reference

A contact point visualization.

```
#include <ContactVisualPrivate.hh>
```

Public Attributes

- **DynamicLines * depth**
- **DynamicLines * normal**
Normal and depth for the contact point.
- **Ogre::SceneNode * sceneNode**
The scene node for the contact visualization.

10.38.1 Detailed Description

A contact point visualization.

10.38.2 Member Data Documentation

10.38.2.1 **DynamicLines * gazebo::rendering::ContactVisualPrivate::ContactPoint::depth**

10.38.2.2 **DynamicLines* gazebo::rendering::ContactVisualPrivate::ContactPoint::normal**

Normal and depth for the contact point.

10.38.2.3 **Ogre::SceneNode* gazebo::rendering::ContactVisualPrivate::ContactPoint::sceneNode**

The scene node for the contact visualization.

The documentation for this class was generated from the following file:

- **ContactVisualPrivate.hh**

10.39 gazebo::physics::ContactPublisher Class Reference

A custom contact publisher created for each contact filter in the **Contact** (p. 279) Manager.

```
#include <ContactManager.hh>
```

Public Attributes

- `std::vector< std::string >` **collisionNames**
- `boost::unordered_set< Collision * >` **collisions**
Pointers of collisions monitored by contact manager for contacts.
- `std::vector< Contact * >` **contacts**
A list of contacts associated to the collisions.
- `transport::PublisherPtr` **publisher**
Contact (p. 279) message publisher.

10.39.1 Detailed Description

A custom contact publisher created for each contact filter in the **Contact** (p. 279) Manager.

10.39.2 Member Data Documentation

10.39.2.1 `std::vector<std::string>` gazebo::physics::ContactPublisher::collisionNames

10.39.2.2 `boost::unordered_set<Collision *>` gazebo::physics::ContactPublisher::collisions

Pointers of collisions monitored by contact manager for contacts.

10.39.2.3 `std::vector<Contact *>` gazebo::physics::ContactPublisher::contacts

A list of contacts associated to the collisions.

10.39.2.4 `transport::PublisherPtr` gazebo::physics::ContactPublisher::publisher

Contact (p. 279) message publisher.

The documentation for this class was generated from the following file:

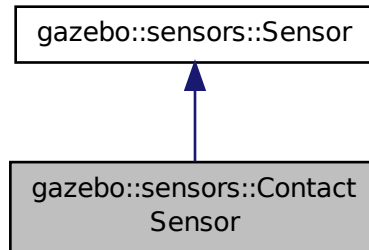
- **ContactManager.hh**

10.40 gazebo::sensors::ContactSensor Class Reference

Contact sensor.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::ContactSensor:



Public Member Functions

- **ContactSensor** ()
Constructor.
- virtual **~ContactSensor** ()
Destructor.
- unsigned int **GetCollisionContactCount** (const std::string &_collisionName) const
Return the number of contacts for an observed collision.
- unsigned int **GetCollisionCount** () const
Get the number of collisions that the sensor is observing.
- std::string **GetCollisionName** (unsigned int _index) const
Get a collision name at index _index.
- msgs::Contacts **GetContacts** () const
*Get all the contacts for the **ContactSensor** (p. 287).*
- std::map< std::string, physics::Contact > **GetContacts** (const std::string &_collisionName)
Gets contacts of a collision.
- virtual void **Init** ()
Initialize the sensor.
- virtual bool **IsActive** ()
Returns true if sensor generation is active.
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)
Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.

Protected Member Functions

- virtual void **Fini** ()
Finalize the sensor.
- virtual bool **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.40.1 Detailed Description

Contact sensor.

This sensor detects and reports contacts between objects

10.40.2 Constructor & Destructor Documentation

10.40.2.1 gazebo::sensors::ContactSensor::ContactSensor ()

Constructor.

10.40.2.2 virtual gazebo::sensors::ContactSensor::~~ContactSensor () [virtual]

Destructor.

10.40.3 Member Function Documentation

10.40.3.1 virtual void gazebo::sensors::ContactSensor::Fini () [protected],[virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 912).

10.40.3.2 unsigned int gazebo::sensors::ContactSensor::GetCollisionContactCount (const std::string & *_collisionName*) const

Return the number of contacts for an observed collision.

Parameters

<i>in</i>	<i>_collisionName</i>	The name of the observed collision.
-----------	-----------------------	-------------------------------------

Returns

The collision contact count.

10.40.3.3 unsigned int gazebo::sensors::ContactSensor::GetCollisionCount () const

Get the number of collisions that the sensor is observing.

Returns

Number of collisions.

10.40.3.4 std::string gazebo::sensors::ContactSensor::GetCollisionName (unsigned int *_index*) const

Get a collision name at index *_index*.

Parameters

in	_index	Index of collision in collection of collisions.
----	--------	---

Returns

name of collision.

10.40.3.5 msgs::Contacts gazebo::sensors::ContactSensor::GetContacts () const

Get all the contacts for the **ContactSensor** (p. 287).

Returns

Message that contains contact information between collision pairs.

During ODEPhysics::UpdateCollisions, all collision pairs in the world are pushed into a buffer within ContactManager. Subsequently, World::Update invokes ContactManager::PublishContacts to publish all contacts generated within a timestep onto Gazebo topic ~/physics/contacts.

Each **ContactSensor** (p. 287) subscribes to the Gazebo ~/physics/contacts topic, retrieves all contact pairs in a time step and filters them within ContactSensor::OnContacts against <collision> body name specified by the **ContactSensor** (p. 287) SDF. All collision pairs between **ContactSensor** (p. 287) <collision> body and other bodies in the world are stored in an array inside contacts.proto.

Within each element of the contact.proto array inside contacts.proto, list of collisions between collision bodies (collision1 and collision 2) are stored in an array of elements, (position, normal, depth, wrench). A timestamp has also been added (time). Details are described below:

- string collision1 name of the first collision object.
- string collision2 name of the second collision object.
- Vector3d position position of the contact joint in inertial frame.
- Vector3d normal normal of the contact joint in inertial frame.
- double depth intersection (penetration) depth of two collision bodies.
- JointWrench wrench Forces and torques acting on both collision bodies. See joint_wrench.proto for details. The forces and torques are applied at the CG of perspective links for each collision body, specified in the inertial frame.
- Time time time at which this contact happened.

10.40.3.6 std::map<std::string, physics::Contact> gazebo::sensors::ContactSensor::GetContacts (const std::string & _collisionName)

Gets contacts of a collision.

Parameters

in	_collisionName	Name of collision
----	----------------	-------------------

Returns

Container of contacts

10.40.3.7 `virtual void gazebo::sensors::ContactSensor::Init () [virtual]`

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.40.3.8 `virtual bool gazebo::sensors::ContactSensor::IsActive () [virtual]`

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.40.3.9 `virtual void gazebo::sensors::ContactSensor::Load (const std::string & _worldName, sdf::ElementPtr _sdf) [virtual]`

Load the sensor with SDF parameters.

Parameters

in	<code>_sdf</code>	SDF Sensor (p. 907) parameters
in	<code>_worldName</code>	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.40.3.10 `virtual void gazebo::sensors::ContactSensor::Load (const std::string & _worldName) [virtual]`

Load the sensor with default parameters.

Parameters

in	<code>_worldName</code>	Name of world to load from.
----	-------------------------	-----------------------------

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.40.3.11 `virtual bool gazebo::sensors::ContactSensor::UpdateImpl (bool) [protected],[virtual]`

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	<code>_force</code>	True if update is forced, false if not
----	---------------------	--

Returns

True if the sensor was updated.

Reimplemented from `gazebo::sensors::Sensor` (p. 917).

The documentation for this class was generated from the following file:

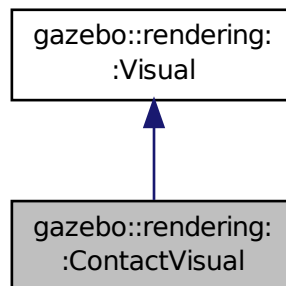
- `ContactSensor.hh`

10.41 gazebo::rendering::ContactVisual Class Reference

Contact visualization.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for `gazebo::rendering::ContactVisual`:



Public Member Functions

- **ContactVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)
Constructor.
- virtual ~**ContactVisual** ()
Destructor.
- void **SetEnabled** (bool _enabled)
Set to true to enable contact visualization.

Additional Inherited Members

10.41.1 Detailed Description

Contact visualization.

This class visualizes contact points by drawing arrows in the 3D environment.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 `gazebo::rendering::ContactVisual::ContactVisual (const std::string & _name, VisualPtr _vis, const std::string & _topicName)`

Constructor.

Parameters

in	<i>_name</i>	Name of the ContactVisual (p. 292)
in	<i>_vis</i>	Pointer the parent Visual (p. 1196)
in	<i>_topicName</i>	Name of the topic which publishes the contact information.

10.41.2.2 `virtual gazebo::rendering::ContactVisual::~~ContactVisual () [virtual]`

Destructor.

10.41.3 Member Function Documentation

10.41.3.1 `void gazebo::rendering::ContactVisual::SetEnabled (bool _enabled)`

Set to true to enable contact visualization.

Parameters

in	<i>_enabled</i>	True to show contacts, false to hide.
----	-----------------	---------------------------------------

The documentation for this class was generated from the following file:

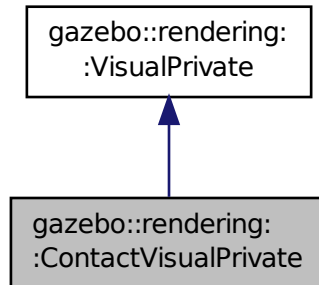
- **ContactVisual.hh**

10.42 gazebo::rendering::ContactVisualPrivate Class Reference

Private data for the Arrow **Visual** (p. 1196) class.

```
#include <ContactVisualPrivate.hh>
```

Inheritance diagram for gazebo::rendering::ContactVisualPrivate:



Classes

- class **ContactPoint**
A contact point visualization.

Public Attributes

- `std::vector< event::ConnectionPtr > connections`
All the event connections.
- `boost::shared_ptr< msgs::Contacts const > contactsMsg`
The current contact message.
- `transport::SubscriberPtr contactsSub`
Subscription to the contact data.
- `bool enabled`
True if this visualization is enabled.
- `boost::mutex mutex`
Mutex to protect the contact message.
- `transport::NodePtr node`
Node for communication.
- `std::vector< ContactVisualPrivate::ContactPoint * > points`
All the contact points.
- `bool receivedMsg`
True if we have received a message.
- `std::string topicName`
Name of the topic contact information is published on.

Additional Inherited Members

10.42.1 Detailed Description

Private data for the Arrow **Visual** (p. 1196) class.

10.42.2 Member Data Documentation

10.42.2.1 `std::vector<event::ConnectionPtr> gazebo::rendering::ContactVisualPrivate::connections`

All the event connections.

10.42.2.2 `boost::shared_ptr<msgs::Contacts const> gazebo::rendering::ContactVisualPrivate::contactsMsg`

The current contact message.

10.42.2.3 `transport::SubscriberPtr gazebo::rendering::ContactVisualPrivate::contactsSub`

Subscription to the contact data.

10.42.2.4 `bool gazebo::rendering::ContactVisualPrivate::enabled`

True if this visualization is enabled.

10.42.2.5 `boost::mutex gazebo::rendering::ContactVisualPrivate::mutex`

Mutex to protect the contact message.

10.42.2.6 `transport::NodePtr gazebo::rendering::ContactVisualPrivate::node`

Node for communication.

10.42.2.7 `std::vector<ContactVisualPrivate::ContactPoint *> gazebo::rendering::ContactVisualPrivate::points`

All the contact points.

10.42.2.8 `bool gazebo::rendering::ContactVisualPrivate::receivedMsg`

True if we have received a message.

10.42.2.9 `std::string gazebo::rendering::ContactVisualPrivate::topicName`

Name of the topic contact information is published on.

The documentation for this class was generated from the following file:

- **ContactVisualPrivate.hh**

10.43 gazebo::rendering::Conversions Class Reference

Conversions (p. 296) **Conversions.hh** (p. 1306) **rendering/Conversions.hh** (p. 1306).

```
#include <Conversions.hh>
```

Static Public Member Functions

- static `Ogre::ColourValue` **Convert** (const `common::Color` &_clr)
Return the equivalent ogre color.
- static `common::Color` **Convert** (const `Ogre::ColourValue` &_clr)
Return the equivalent gazebo color.
- static `Ogre::Vector3` **Convert** (const `math::Vector3` &_v)
*return **Ogre** (p. 137) Vector from Gazebo Vector3*
- static `math::Vector3` **Convert** (const `Ogre::Vector3` &_v)
return gazebo Vector from ogre Vector3
- static `Ogre::Quaternion` **Convert** (const `math::Quaternion` &_v)
*Gazebo quaternion to **Ogre** (p. 137) quaternion.*
- static `math::Quaternion` **Convert** (const `Ogre::Quaternion` &_v)
***Ogre** (p. 137) quaternion to Gazebo quaternion.*

10.43.1 Detailed Description

Conversions (p. 296) **Conversions.hh** (p. 1306) **rendering/Conversions.hh** (p. 1306).

A set of utility function to convert between Gazebo and **Ogre** (p. 137) data types

10.43.2 Member Function Documentation

10.43.2.1 static `Ogre::ColourValue` gazebo::rendering::Conversions::Convert (const `common::Color` &_clr) [static]

Return the equivalent ogre color.

Parameters

in	_clr	Gazebo color to convert
----	------	-------------------------

Returns

Ogre (p. 137) color value

10.43.2.2 static `common::Color` gazebo::rendering::Conversions::Convert (const `Ogre::ColourValue` &_clr) [static]

Return the equivalent gazebo color.

Parameters

in	_clr	Ogre (p. 137) color to convert
----	------	---------------------------------------

Returns

Gazebo color value

10.43.2.3 `static Ogre::Vector3 gazebo::rendering::Conversions::Convert (const math::Vector3 & _v) [static]`

return **Ogre** (p. 137) Vector from Gazebo Vector3

Parameters

<code>in</code>	<code>_v</code>	Gazebo vector
-----------------	-----------------	---------------

Returns

Ogre (p. 137) vector

10.43.2.4 `static math::Vector3 gazebo::rendering::Conversions::Convert (const Ogre::Vector3 & _v) [static]`

return gazebo Vector from ogre Vector3

Parameters

<code>in</code>	<code>_v</code>	Ogre (p. 137) vector
-----------------	-----------------	-----------------------------

Returns

Gazebo vector

10.43.2.5 `static Ogre::Quaternion gazebo::rendering::Conversions::Convert (const math::Quaternion & _v) [static]`

Gazebo quaternion to **Ogre** (p. 137) quaternion.

Parameters

<code>in</code>	<code>_v</code>	Gazebo quaternion
-----------------	-----------------	-------------------

Returns

Ogre (p. 137) quaternion

10.43.2.6 `static math::Quaternion gazebo::rendering::Conversions::Convert (const Ogre::Quaternion & _v) [static]`

Ogre (p. 137) quaternion to Gazebo quaternion.

Parameters

<code>in</code>	<code>_v</code>	Ogre (p. 137) quaternion return Gazebo quaternion
-----------------	-----------------	--

The documentation for this class was generated from the following file:

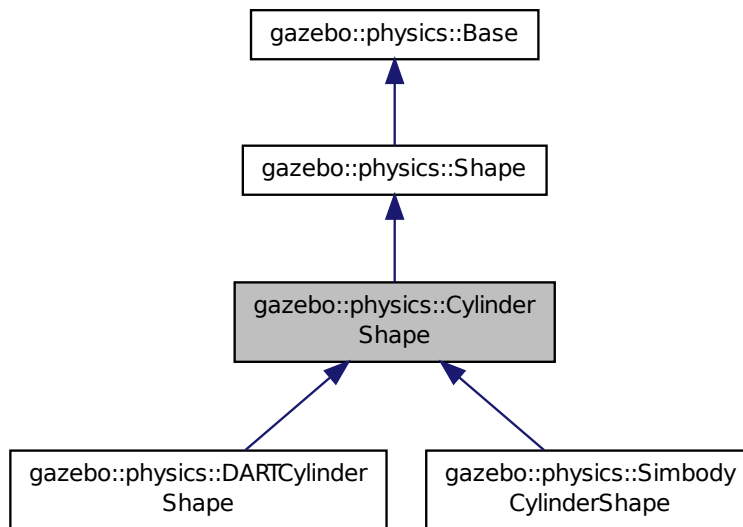
- **Conversions.hh**

10.44 gazebo::physics::CylinderShape Class Reference

Cylinder collision.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::CylinderShape:



Public Member Functions

- **CylinderShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~CylinderShape** ()
Destructor.
- void **FillMsg** (msgs::Geometry &_msg)
Fill in the values for a geometry message.
- double **GetLength** () const
Get length.
- double **GetRadius** () const
Get radius.
- void **Init** ()
Initialize the cylinder.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
Update values based on a message.

- void **SetLength** (double *_length*)
Set length.
- void **SetRadius** (double *_radius*)
Set radius.
- virtual void **SetScale** (const **math::Vector3** &*_scale*)
Set scale of cylinder.
- virtual void **SetSize** (double *_radius*, double *_length*)
Set the size of the cylinder.

Additional Inherited Members

10.44.1 Detailed Description

Cylinder collision.

10.44.2 Constructor & Destructor Documentation

10.44.2.1 gazebo::physics::CylinderShape::CylinderShape (**CollisionPtr** *_parent*) [explicit]

Constructor.

Parameters

<i>in</i>	<i>_parent</i>	Parent of the shape.
-----------	----------------	----------------------

10.44.2.2 virtual gazebo::physics::CylinderShape::~~CylinderShape () [virtual]

Destructor.

10.44.3 Member Function Documentation

10.44.3.1 void gazebo::physics::CylinderShape::FillMsg (**msgs::Geometry** & *_msg*) [virtual]

Fill in the values for a geometry message.

Parameters

<i>out</i>	<i>_msg</i>	The geometry message to fill.
------------	-------------	-------------------------------

Implements **gazebo::physics::Shape** (p. 934).

10.44.3.2 double gazebo::physics::CylinderShape::GetLength () const

Get length.

Returns

The cylinder length.

10.44.3.3 `double gazebo::physics::CylinderShape::GetRadius () const`

Get radius.

Returns

The cylinder radius.

10.44.3.4 `void gazebo::physics::CylinderShape::Init () [virtual]`

Initialize the cylinder.

Implements **`gazebo::physics::Shape`** (p. 935).

10.44.3.5 `virtual void gazebo::physics::CylinderShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]`

Update values based on a message.

Parameters

<code>in</code>	<code>_msg</code>	Message to update from.
-----------------	-------------------	-------------------------

Implements **`gazebo::physics::Shape`** (p. 935).

10.44.3.6 `void gazebo::physics::CylinderShape::SetLength (double _length)`

Set length.

Parameters

<code>in</code>	<code>_length</code>	New length of the cylinder.
-----------------	----------------------	-----------------------------

10.44.3.7 `void gazebo::physics::CylinderShape::SetRadius (double _radius)`

Set radius.

Parameters

<code>in</code>	<code>_radius</code>	New radius of the cylinder.
-----------------	----------------------	-----------------------------

10.44.3.8 `virtual void gazebo::physics::CylinderShape::SetScale (const math::Vector3 & _scale) [virtual]`

Set scale of cylinder.

Parameters

<code>in</code>	<code>_scale</code>	Scale to set the cylinder to.
-----------------	---------------------	-------------------------------

Implements **`gazebo::physics::Shape`** (p. 935).

10.44.3.9 virtual void gazebo::physics::CylinderShape::SetSize (double *_radius*, double *_length*) [virtual]

Set the size of the cylinder.

Parameters

in	<i>_radius</i>	New radius.
in	<i>_length</i>	New length.

Reimplemented in **gazebo::physics::SimbodyCylinderShape** (p. 948), and **gazebo::physics::DARTCylinderShape** (p. 315).

Referenced by gazebo::physics::DARTCylinderShape::SetSize(), and gazebo::physics::SimbodyCylinderShape::SetSize().

The documentation for this class was generated from the following file:

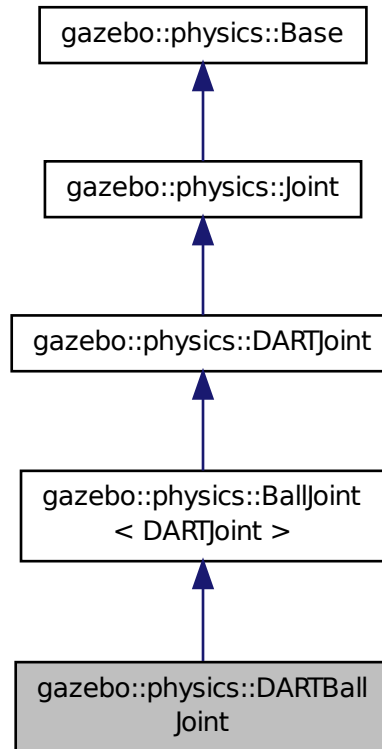
- **CylinderShape.hh**

10.45 gazebo::physics::DARTBallJoint Class Reference

An **DARTBallJoint** (p. 301).

```
#include <DARTBallJoint.hh>
```

Inheritance diagram for gazebo::physics::DARTBallJoint:



Public Member Functions

- **DARTBallJoint** (BasePtr _parent)
Constructor.
- virtual **~DARTBallJoint** ()
Destructor.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const
Get the anchor point.
- virtual **math::Angle GetAngleImpl** (unsigned int _index) const
Get the angle of an axis helper function.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const
Get the axis of rotation in global coordinate frame.
- virtual **math::Angle GetHighStop** (unsigned int _index)
Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (unsigned int _index)
Get the low stop of an axis(index).
- virtual double **GetMaxForce** (unsigned int _index)

- Get the max allowed force of an axis(index).*

 - virtual double **GetVelocity** (unsigned int _index) const

Get the rotation rate of an axis(index)
- virtual void **Init** ()

Initialize joint.
- virtual void **Load** (sdf::ElementPtr _sdf)

*Template to ::Load the **BallJoint** (p. 167).*
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

Set the axis of rotation where axis is specified in local joint frame.
- virtual bool **SetHighStop** (unsigned int _index, const **math::Angle** &_angle)

Set the high stop of an axis(index).
- virtual bool **SetLowStop** (unsigned int _index, const **math::Angle** &_angle)

Set the low stop of an axis(index).
- virtual void **SetMaxForce** (unsigned int _index, double _t)

Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (unsigned int _index, double _angle)

Set the velocity of an axis(index).

Protected Member Functions

- void **SetForceImpl** (unsigned int _index, double _torque)
- Set the force applied to this **physics::Joint** (p. 541).*

Protected Attributes

- dart::dynamics::BallJoint * **dtBallJoint**

Additional Inherited Members

10.45.1 Detailed Description

An **DARTBallJoint** (p. 301).

10.45.2 Constructor & Destructor Documentation

10.45.2.1 gazebo::physics::DARTBallJoint::DARTBallJoint (**BasePtr** _parent)

Constructor.

Parameters

in	_parent	Parent of the Joint (p. 541)
----	---------	-------------------------------------

10.45.2.2 virtual gazebo::physics::DARTBallJoint::~~DARTBallJoint () [virtual]

Destructor.

10.45.3 Member Function Documentation

10.45.3.1 `virtual math::Vector3 gazebo::physics::DARTBallJoint::GetAnchor (unsigned int _index) const [virtual]`

Get the anchor point.

Parameters

<i>in</i>	<i>_index</i>	Index of the axis.
-----------	---------------	--------------------

Returns

Anchor value for the axis.

Implements `gazebo::physics::Joint` (p. 549).

10.45.3.2 `virtual math::Angle gazebo::physics::DARTBallJoint::GetAngleImpl (unsigned int _index) const [virtual]`

Get the angle of an axis helper function.

Parameters

<i>in</i>	<i>_index</i>	Index of the axis.
-----------	---------------	--------------------

Returns

Angle of the axis.

Implements `gazebo::physics::Joint` (p. 550).

10.45.3.3 `virtual math::Vector3 gazebo::physics::DARTBallJoint::GetGlobalAxis (unsigned int _index) const [virtual]`

Get the axis of rotation in global coordinate frame.

Parameters

<i>in</i>	<i>_index</i>	Index of the axis to get.
-----------	---------------	---------------------------

Returns

Axis value for the provided index.

Implements `gazebo::physics::Joint` (p. 553).

10.45.3.4 `virtual math::Angle gazebo::physics::DARTBallJoint::GetHighStop (unsigned int _index) [virtual]`

Get the high stop of an axis(index).

This function is replaced by `GetUpperLimit(unsigned int)`. If you are interested in getting the value of `dParamHiStop*`, use `GetAttribute(hi_stop, _index)`

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

Angle of the high stop value.

Reimplemented from **gazebo::physics::DARTJoint** (p. 332).

10.45.3.5 virtual math::Angle gazebo::physics::DARTBallJoint::GetLowStop (unsigned int _index) [virtual]

Get the low stop of an axis(index).

This function is replaced by GetLowerLimit(unsigned int). If you are interested in getting the value of dParamHiStop*, use GetAttribute(hi_stop, _index)

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

Angle of the low stop value.

Reimplemented from **gazebo::physics::DARTJoint** (p. 334).

10.45.3.6 virtual double gazebo::physics::DARTBallJoint::GetMaxForce (unsigned int _index) [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 556).

10.45.3.7 virtual double gazebo::physics::DARTBallJoint::GetVelocity (unsigned int _index) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

The rotational velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 558).

10.45.3.8 `virtual void gazebo::physics::DARTBallJoint::Init () [virtual]`

Initialize joint.

Reimplemented from **gazebo::physics::BallJoint< DARTJoint >** (p. 168).

10.45.3.9 `virtual void gazebo::physics::DARTBallJoint::Load (sdf::ElementPtr _sdf) [virtual]`

Template to `::Load` the **BallJoint** (p. 167).

Parameters

<code>in</code>	<code>_sdf</code>	SDF to load the joint from.
-----------------	-------------------	-----------------------------

Reimplemented from **gazebo::physics::BallJoint< DARTJoint >** (p. 168).

10.45.3.10 `virtual void gazebo::physics::DARTBallJoint::SetAxis (unsigned int _index, const math::Vector3 & _axis) [virtual]`

Set the axis of rotation where axis is specified in local joint frame.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to set.
<code>in</code>	<code>_axis</code>	Vector in local joint frame of axis direction (must have length greater than zero).

Implements **gazebo::physics::Joint** (p. 561).

10.45.3.11 `void gazebo::physics::DARTBallJoint::SetForceImpl (unsigned int _index, double _force) [protected], [virtual]`

Set the force applied to this **physics::Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to `SetForceImpl` to the same joint in the same time step will accumulate forces on that **Joint** (p. 541)).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_force</code>	Force value.

Implements **gazebo::physics::DARTJoint** (p. 336).

10.45.3.12 `virtual bool gazebo::physics::DARTBallJoint::SetHighStop (unsigned int _index, const math::Angle & _angle)`
`[virtual]`

Set the high stop of an axis(index).

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_angle</i></code>	High stop angle.

Reimplemented from `gazebo::physics::DARTJoint` (p. 336).

10.45.3.13 `virtual bool gazebo::physics::DARTBallJoint::SetLowStop (unsigned int _index, const math::Angle & _angle)`
`[virtual]`

Set the low stop of an axis(index).

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_angle</i></code>	Low stop angle.

Reimplemented from `gazebo::physics::DARTJoint` (p. 336).

10.45.3.14 `virtual void gazebo::physics::DARTBallJoint::SetMaxForce (unsigned int _index, double _force)` `[virtual]`

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_force</i></code>	Maximum force that can be applied to the axis.

Implements `gazebo::physics::Joint` (p. 563).

10.45.3.15 `virtual void gazebo::physics::DARTBallJoint::SetVelocity (unsigned int _index, double _vel)` `[virtual]`

Set the velocity of an axis(index).

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_vel</i></code>	Velocity.

Implements `gazebo::physics::Joint` (p. 565).

10.45.4 Member Data Documentation

10.45.4.1 `dart::dynamics::BallJoint*` `gazebo::physics::DARTBallJoint::dtBallJoint` [protected]

The documentation for this class was generated from the following file:

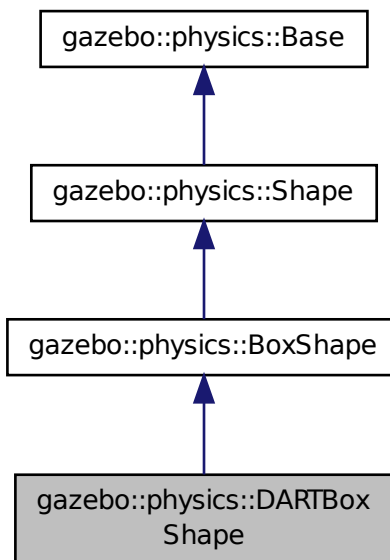
- **DARTBallJoint.hh**

10.46 gazebo::physics::DARTBoxShape Class Reference

DART Box shape.

```
#include <DARTBoxShape.hh>
```

Inheritance diagram for gazebo::physics::DARTBoxShape:



Public Member Functions

- **DARTBoxShape** (`DARTCollisionPtr` _parent)
Constructor.
- virtual `~DARTBoxShape` ()
Destructor.
- virtual void **SetSize** (const `math::Vector3` &_size)
Set the size of the box.

Additional Inherited Members

10.46.1 Detailed Description

DART Box shape.

10.46.2 Constructor & Destructor Documentation

10.46.2.1 gazebo::physics::DARTBoxShape::DARTBoxShape (DARTCollisionPtr *_parent*) [inline],[explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent Collision (p. 235).
----	----------------	-----------------------------------

10.46.2.2 virtual gazebo::physics::DARTBoxShape::~~DARTBoxShape () [inline],[virtual]

Destructor.

10.46.3 Member Function Documentation

10.46.3.1 virtual void gazebo::physics::DARTBoxShape::SetSize (const math::Vector3 & *_size*) [inline],[virtual]

Set the size of the box.

Parameters

in	<i>_size</i>	Size of each side of the box.
----	--------------	-------------------------------

Reimplemented from gazebo::physics::BoxShape (p. 188).

References gazebo::physics::DARTTypes::ConvVec3(), gazebo::math::equal(), gazebo::physics::DARTCollision::GetDARTBodyNode(), gzerr, gzwarn, NULL, gazebo::physics::BoxShape::SetSize(), gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

The documentation for this class was generated from the following file:

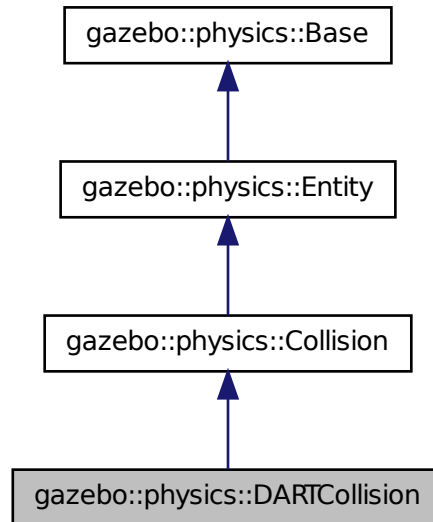
- **DARTBoxShape.hh**

10.47 gazebo::physics::DARTCollision Class Reference

Base (p. 168) class for all DART collisions.

```
#include <DARTCollision.hh>
```

Inheritance diagram for gazebo::physics::DARTCollision:



Public Member Functions

- **DARTCollision** (**LinkPtr** _parent)
Constructor.
- virtual **~DARTCollision** ()
Destructor.
- virtual void **Fini** ()
Finalize the collision.
- virtual **math::Box** **GetBoundingBox** () const
Get the bounding box for this collision.
- virtual unsigned int **GetCategoryBits** () const
Get the category bits, used during collision detection.
- virtual unsigned int **GetCollideBits** () const
Get the collide bits, used during collision detection.
- dart::dynamics::BodyNode * **GetDARTBodyNode** () const
Get DART body node.
- dart::dynamics::Shape * **GetDARTCollisionShape** () const
Get DART collision shape.
- virtual void **Init** ()
Initialize the collision.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the collision.
- virtual void **OnPoseChange** ()

This function is called when the entity's (or one of its parents) pose of the parent has changed.

- virtual void **SetCategoryBits** (unsigned int _bits)
Set the category bits, used during collision detection.
- virtual void **SetCollideBits** (unsigned int _bits)
Set the collide bits, used during collision detection.
- void **SetDARTCollisionShape** (dart::dynamics::Shape *_shape, bool _placeable=true)
Set DART collision shape.

Additional Inherited Members

10.47.1 Detailed Description

Base (p. 168) class for all DART collisions.

10.47.2 Constructor & Destructor Documentation

10.47.2.1 gazebo::physics::DARTCollision::DARTCollision (LinkPtr _parent) [explicit]

Constructor.

Parameters

in	_link	Parent Link (p. 595)
----	-------	-----------------------------

10.47.2.2 virtual gazebo::physics::DARTCollision::~~DARTCollision () [virtual]

Destructor.

10.47.3 Member Function Documentation

10.47.3.1 virtual void gazebo::physics::DARTCollision::Fini () [virtual]

Finalize the collision.

Reimplemented from **gazebo::physics::Collision** (p. 239).

10.47.3.2 virtual math::Box gazebo::physics::DARTCollision::GetBoundingBox () const [virtual]

Get the bounding box for this collision.

Returns

The bounding box.

Implements **gazebo::physics::Collision** (p. 239).

10.47.3.3 virtual unsigned int gazebo::physics::DARTCollision::GetCategoryBits () const [virtual]

Get the category bits, used during collision detection.

Returns

The bits

10.47.3.4 `virtual unsigned int gazebo::physics::DARTCollision::GetCollideBits () const [virtual]`

Get the collide bits, used during collision detection.

Returns

The bits

10.47.3.5 `dart::dynamics::BodyNode* gazebo::physics::DARTCollision::GetDARTBodyNode () const`

Get DART body node.

Returns

Pointer to the dart BodyNode.

Referenced by `gazebo::physics::DARTPlaneShape::CreatePlane()`, `gazebo::physics::DARTSphereShape::SetRadius()`, `gazebo::physics::DARTCylinderShape::SetSize()`, and `gazebo::physics::DARTBoxShape::SetSize()`.

10.47.3.6 `dart::dynamics::Shape* gazebo::physics::DARTCollision::GetDARTCollisionShape () const`

Get DART collision shape.

10.47.3.7 `virtual void gazebo::physics::DARTCollision::Init () [virtual]`

Initialize the collision.

Reimplemented from **`gazebo::physics::Collision`** (p. 242).

10.47.3.8 `virtual void gazebo::physics::DARTCollision::Load (sdf::ElementPtr _sdf) [virtual]`

Load the collision.

Parameters

in	<code>_sdf</code>	SDF to load from.
----	-------------------	-------------------

Reimplemented from **`gazebo::physics::Collision`** (p. 243).

10.47.3.9 `virtual void gazebo::physics::DARTCollision::OnPoseChange () [virtual]`

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implements **`gazebo::physics::Entity`** (p. 412).

10.47.3.10 virtual void gazebo::physics::DARTCollision::SetCategoryBits (unsigned int *_bits*) [virtual]

Set the category bits, used during collision detection.

Parameters

in	<i>_bits</i>	The bits to set.
----	--------------	------------------

Implements **gazebo::physics::Collision** (p. 243).

10.47.3.11 virtual void gazebo::physics::DARTCollision::SetCollideBits (unsigned int *_bits*) [virtual]

Set the collide bits, used during collision detection.

Parameters

in	<i>_bits</i>	The bits to set.
----	--------------	------------------

Implements **gazebo::physics::Collision** (p. 243).

10.47.3.12 void gazebo::physics::DARTCollision::SetDARTCollisionShape (dart::dynamics::Shape * *_shape*, bool *_placeable* = true)

Set DART collision shape.

Parameters

in	<i>_shape</i>	DART Collision (p. 235) shape
in	<i>_placeable</i>	True to make the object movable.

The documentation for this class was generated from the following file:

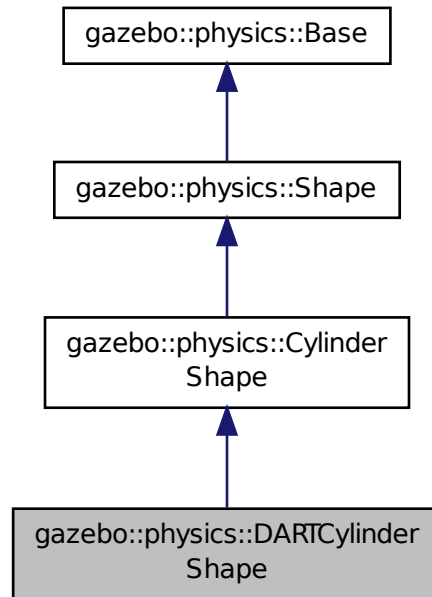
- **DARTCollision.hh**

10.48 gazebo::physics::DARTCylinderShape Class Reference

DART cylinder shape.

```
#include <DARTCylinderShape.hh>
```

Inheritance diagram for gazebo::physics::DARTCylinderShape:



Public Member Functions

- **DARTCylinderShape** (*CollisionPtr* _parent)
Constructor.
- virtual **~DARTCylinderShape** ()
Destructor.
- void **SetSize** (double _radius, double _length)
Set the size of the cylinder.

Additional Inherited Members

10.48.1 Detailed Description

DART cylinder shape.

10.48.2 Constructor & Destructor Documentation

10.48.2.1 gazebo::physics::DARTCylinderShape::DARTCylinderShape (*CollisionPtr* _parent) [inline],[explicit]

Constructor.

Parameters

<code>in</code>	<code>_parent</code>	Collision (p. 235) parent.
-----------------	----------------------	-----------------------------------

10.48.2.2 `virtual gazebo::physics::DARTCylinderShape::~~DARTCylinderShape () [inline],[virtual]`

Destructor.

10.48.3 Member Function Documentation

10.48.3.1 `void gazebo::physics::DARTCylinderShape::SetSize (double _radius, double _length) [inline],[virtual]`

Set the size of the cylinder.

Parameters

<code>in</code>	<code>_radius</code>	New radius.
<code>in</code>	<code>_length</code>	New length.

Reimplemented from `gazebo::physics::CylinderShape` (p. 301).

References `gazebo::math::equal()`, `gazebo::physics::DARTCollision::GetDARTBodyNode()`, `gzerr`, `gzwarn`, `NULL`, and `gazebo::physics::CylinderShape::SetSize()`.

The documentation for this class was generated from the following file:

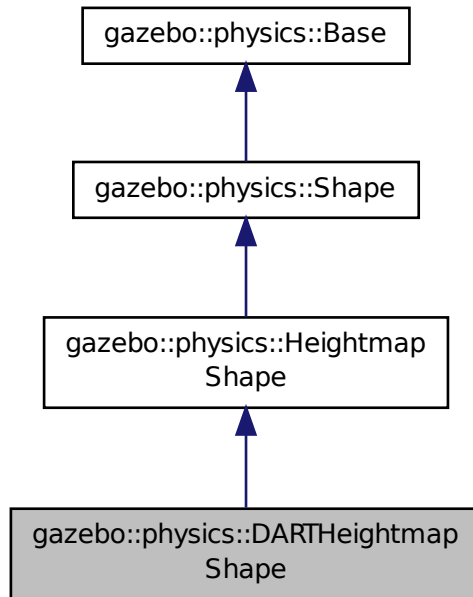
- **DARTCylinderShape.hh**

10.49 gazebo::physics::DARTHeightmapShape Class Reference

DART Height map collision.

```
#include <DARTHeightmapShape.hh>
```

Inheritance diagram for gazebo::physics::DARTHeightmapShape:



Public Member Functions

- **DARTHeightmapShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~DARTHeightmapShape** ()
Destructor.
- virtual void **Init** ()
Initialize the heightmap.

Additional Inherited Members

10.49.1 Detailed Description

DART Height map collision.

10.49.2 Constructor & Destructor Documentation

10.49.2.1 gazebo::physics::DARTHeightmapShape::DARTHeightmapShape (**CollisionPtr** _parent)

Constructor.

Parameters

in	<i>_parent</i>	Collision (p. 235) parent.
----	----------------	-----------------------------------

10.49.2.2 virtual gazebo::physics::DARTHeightmapShape::~~DARTHeightmapShape () [virtual]

Destructor.

10.49.3 Member Function Documentation

10.49.3.1 virtual void gazebo::physics::DARTHeightmapShape::Init () [virtual]

Initialize the heightmap.

Reimplemented from **gazebo::physics::HeightmapShape** (p. 510).

The documentation for this class was generated from the following file:

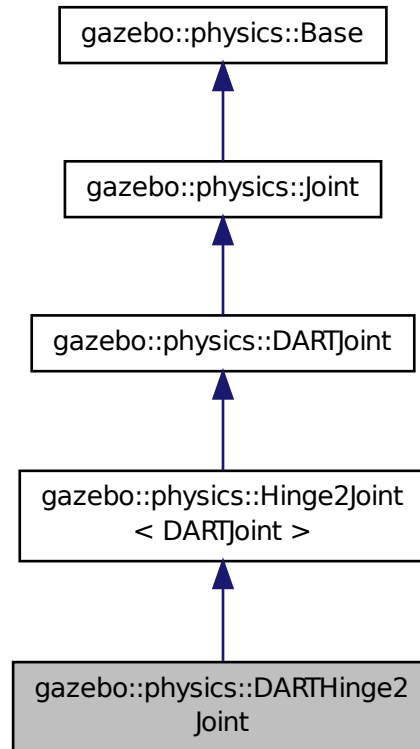
- **DARTHeightmapShape.hh**

10.50 gazebo::physics::DARTHinge2Joint Class Reference

A two axis hinge joint.

```
#include <DARTHinge2Joint.hh>
```

Inheritance diagram for gazebo::physics::DARTHinge2Joint:



Public Member Functions

- **DARTHinge2Joint** (**BasePtr** _parent)
Constructor.
- virtual **~DARTHinge2Joint** ()
Destructor.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const
Get the anchor point.
- virtual **math::Angle GetAngleImpl** (unsigned int _index) const
Get the angle of an axis helper function.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const
Get the axis of rotation in global coordinate frame.
- virtual double **GetMaxForce** (unsigned int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (unsigned int _index) const
Get the rotation rate of an axis(index)
- virtual void **Init** ()

Initialize a joint.

- virtual void **Load** (sdf::ElementPtr _sdf)
Load the joint.
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)
Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int _index, double _force)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (unsigned int _index, double _vel)
Set the velocity of an axis(index).

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _effort)
*Set the force applied to this **physics::Joint** (p. 541).*

Protected Attributes

- dart::dynamics::UniversalJoint * **dtUniversalJoint**
Universal joint of DART.

Additional Inherited Members

10.50.1 Detailed Description

A two axis hinge joint.

10.50.2 Constructor & Destructor Documentation

10.50.2.1 gazebo::physics::DARTHinge2Joint::DARTHinge2Joint (BasePtr _parent)

Constructor.

Parameters

in	_parent	Parent of the Joint (p. 541)
----	---------	-------------------------------------

10.50.2.2 virtual gazebo::physics::DARTHinge2Joint::~~DARTHinge2Joint () [virtual]

Destructor.

10.50.3 Member Function Documentation

10.50.3.1 virtual **math::Vector3** gazebo::physics::DARTHinge2Joint::GetAnchor (unsigned int _index) const [virtual]

Get the anchor point.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 549).

10.50.3.2 `virtual math::Angle gazebo::physics::DARTHinge2Joint::GetAngleImpl (unsigned int _index) const` [virtual]

Get the angle of an axis helper function.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 550).

10.50.3.3 `virtual math::Vector3 gazebo::physics::DARTHinge2Joint::GetGlobalAxis (unsigned int _index) const` [virtual]

Get the axis of rotation in global coordinate frame.

Parameters

in	<i>_index</i>	Index of the axis to get.
----	---------------	---------------------------

Returns

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 553).

10.50.3.4 `virtual double gazebo::physics::DARTHinge2Joint::GetMaxForce (unsigned int _index)` [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 556).

10.50.3.5 virtual double gazebo::physics::DARTHinge2Joint::GetVelocity (unsigned int *_index*) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 558).

10.50.3.6 virtual void gazebo::physics::DARTHinge2Joint::Init () [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::DARTJoint** (p. 334).

10.50.3.7 virtual void gazebo::physics::DARTHinge2Joint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the joint.

Parameters

in	<i>_sdf</i>	SDF values to load from.
----	-------------	--------------------------

Reimplemented from **gazebo::physics::Hinge2Joint< DARTJoint >** (p. 513).

10.50.3.8 virtual void gazebo::physics::DARTHinge2Joint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	<i>_index</i>	Index of the axis to set.
in	<i>_axis</i>	Vector in local joint frame of axis direction (must have length greater than zero).

Implements **gazebo::physics::Joint** (p. 561).

10.50.3.9 virtual void gazebo::physics::DARTHinge2Joint::SetForceImpl (unsigned int *_index*, double *_force*) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to `SetForceImpl` to the same joint in the same time step will accumulate forces on that **Joint** (p. 541)).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_force</code>	Force value.

Implements `gazebo::physics::DARTJoint` (p. 336).

10.50.3.10 `virtual void gazebo::physics::DARTHinge2Joint::SetMaxForce (unsigned int _index, double _force) [virtual]`

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_force</code>	Maximum force that can be applied to the axis.

Implements `gazebo::physics::Joint` (p. 563).

10.50.3.11 `virtual void gazebo::physics::DARTHinge2Joint::SetVelocity (unsigned int _index, double _vel) [virtual]`

Set the velocity of an axis(index).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_vel</code>	Velocity.

Implements `gazebo::physics::Joint` (p. 565).

10.50.4 Member Data Documentation

10.50.4.1 `dart::dynamics::UniversalJoint* gazebo::physics::DARTHinge2Joint::dtUniversalJoint [protected]`

Universal joint of DART.

The documentation for this class was generated from the following file:

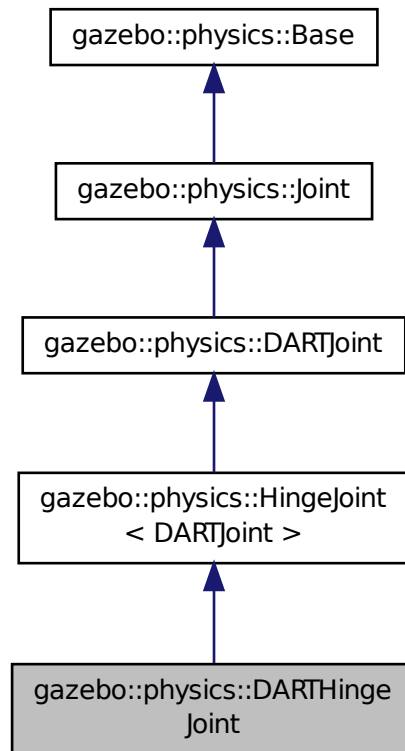
- **DARTHinge2Joint.hh**

10.51 gazebo::physics::DARTHingeJoint Class Reference

A single axis hinge joint.

```
#include <DARTHingeJoint.hh>
```

Inheritance diagram for gazebo::physics::DARTHingeJoint:



Public Member Functions

- **DARTHingeJoint** (**BasePtr** _parent)
Constructor.
- virtual **~DARTHingeJoint** ()
Destructor.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const
Get the anchor point.
- virtual **math::Angle GetAngleImpl** (unsigned int _index) const
Get the angle of an axis helper function.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const
Get the axis of rotation in global coordinate frame.
- virtual double **GetMaxForce** (unsigned int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (unsigned int _index) const
Get the rotation rate of an axis(index)
- virtual void **Init** ()

Initialize joint.

- virtual void **Load** (sdf::ElementPtr _sdf)

Load joint.

- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

Set the axis of rotation where axis is specified in local joint frame.

- virtual void **SetMaxForce** (unsigned int _index, double _force)

Set the max allowed force of an axis(index).

- virtual void **SetVelocity** (unsigned int _index, double _vel)

Set the velocity of an axis(index).

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _effort)

*Set the force applied to this **physics::Joint** (p. 541).*

Protected Attributes

- dart::dynamics::RevoluteJoint * **dtRevoluteJoint**

Revolute joint of DART.

Additional Inherited Members

10.51.1 Detailed Description

A single axis hinge joint.

10.51.2 Constructor & Destructor Documentation

10.51.2.1 gazebo::physics::DARTHingeJoint::DARTHingeJoint (BasePtr _parent)

Constructor.

Parameters

in	_parent	Parent of the Joint (p. 541)
----	---------	-------------------------------------

10.51.2.2 virtual gazebo::physics::DARTHingeJoint::~~DARTHingeJoint () [virtual]

Destructor.

10.51.3 Member Function Documentation

10.51.3.1 virtual **math::Vector3** gazebo::physics::DARTHingeJoint::GetAnchor (unsigned int _index) const [virtual]

Get the anchor point.

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 549).

10.51.3.2 `virtual math::Angle gazebo::physics::DARTHingeJoint::GetAngleImpl (unsigned int _index) const` [virtual]

Get the angle of an axis helper function.

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 550).

10.51.3.3 `virtual math::Vector3 gazebo::physics::DARTHingeJoint::GetGlobalAxis (unsigned int _index) const` [virtual]

Get the axis of rotation in global coordinate frame.

Parameters

in	_index	Index of the axis to get.
----	--------	---------------------------

Returns

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 553).

10.51.3.4 `virtual double gazebo::physics::DARTHingeJoint::GetMaxForce (unsigned int _index)` [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 556).

10.51.3.5 `virtual double gazebo::physics::DARTHingeJoint::GetVelocity (unsigned int _index) const` [virtual]

Get the rotation rate of an axis(index)

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

The rotational velocity of the joint axis.

Implements `gazebo::physics::Joint` (p. 558).

10.51.3.6 `virtual void gazebo::physics::DARTHingeJoint::Init ()` [virtual]

Initialize joint.

Reimplemented from `gazebo::physics::HingeJoint< DARTJoint >` (p. 514).

10.51.3.7 `virtual void gazebo::physics::DARTHingeJoint::Load (sdf::ElementPtr _sdf)` [virtual]

Load joint.

Parameters

<code>in</code>	<code>_sdf</code>	Pointer to SDF element
-----------------	-------------------	------------------------

Reimplemented from `gazebo::physics::HingeJoint< DARTJoint >` (p. 514).

10.51.3.8 `virtual void gazebo::physics::DARTHingeJoint::SetAxis (unsigned int _index, const math::Vector3 & _axis)` [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to set.
<code>in</code>	<code>_axis</code>	Vector in local joint frame of axis direction (must have length greater than zero).

Implements `gazebo::physics::Joint` (p. 561).

10.51.3.9 `virtual void gazebo::physics::DARTHingeJoint::SetForceImpl (unsigned int _index, double _force)` [protected], [virtual]

Set the force applied to this `physics::Joint` (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to `SetForceImpl` to the same joint in the same time step will accumulate forces on that `Joint` (p. 541)).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_force</i>	Force value.

Implements **gazebo::physics::DARTJoint** (p. 336).

10.51.3.10 virtual void gazebo::physics::DARTHingeJoint::SetMaxForce (unsigned int *_index*, double *_force*) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_force</i>	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 563).

10.51.3.11 virtual void gazebo::physics::DARTHingeJoint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_vel</i>	Velocity.

Implements **gazebo::physics::Joint** (p. 565).

10.51.4 Member Data Documentation

10.51.4.1 dart::dynamics::RevoluteJoint* gazebo::physics::DARTHingeJoint::dtRevoluteJoint [protected]

Revolute joint of DART.

The documentation for this class was generated from the following file:

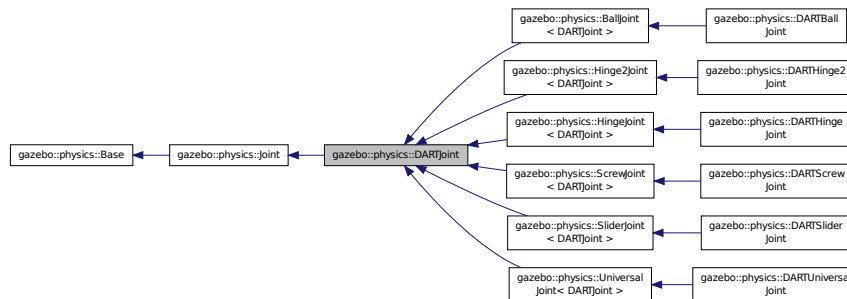
- **DARTHingeJoint.hh**

10.52 gazebo::physics::DARTJoint Class Reference

DART joint interface.

```
#include <DARTJoint.hh>
```

Inheritance diagram for gazebo::physics::DARTJoint:



Public Member Functions

- **DARTJoint** (**BasePtr** _parent)
Constructor.
- virtual **~DARTJoint** ()
Destructor.
- virtual void **ApplyDamping** ()
Callback to apply damping force to joint.
- virtual bool **AreConnected** (**LinkPtr** _one, **LinkPtr** _two) const
Determines if the two bodies are connected by a joint.
- virtual void **Attach** (**LinkPtr** _parent, **LinkPtr** _child)
Attach the two bodies with this joint.
- virtual void **Detach** ()
Detach this joint from all links.
- virtual unsigned int **GetAngleCount** () const
Get the angle count.
- virtual double **GetAttribute** (const std::string &_key, unsigned int _index) **GAZEBO_DEPRECATED(3.0)**
Get a non-generic parameter for the joint.
- dart::dynamics::Joint * **GetDARTJoint** ()
Get DART joint pointer.
- **DARTModelPtr** **GetDARTModel** () const
Get DART model pointer.
- virtual double **GetForce** (unsigned int _index)
- virtual **JointWrench** **GetForceTorque** (unsigned int _index)
get internal force and torque values at a joint.
- virtual **math::Angle** **GetHighStop** (unsigned int _index)
Get the high stop of an axis(index).
- virtual **LinkPtr** **GetJointLink** (unsigned int _index) const
Get the link to which the joint is attached according to the _index.
- virtual **math::Vector3** **GetLinkForce** (unsigned int _index) const
*Get the forces applied to the center of mass of a **physics::Link** (p. 595) due to the existence of this **Joint** (p. 541).*
- virtual **math::Vector3** **GetLinkTorque** (unsigned int _index) const
*Get the torque applied to the center of mass of a **physics::Link** (p. 595) due to the existence of this **Joint** (p. 541).*

- virtual **math::Angle GetLowStop** (unsigned int _index)
Get the low stop of an axis(index).
- virtual double **GetParam** (const std::string &_key, unsigned int _index)
Get a non-generic parameter for the joint.
- virtual void **Init** ()
Initialize a joint.
- virtual void **Load** (sdf::ElementPtr _sdf)
*Load **physics::Joint** (p. 541) from a SDF sdf::Element.*
- virtual void **Reset** ()
Reset the joint.
- virtual void **SetAnchor** (unsigned int, const **gazebo::math::Vector3** &)
Set the anchor point.
- virtual void **SetAttribute** (const std::string &_key, unsigned int _index, const boost::any &_value) **GAZEBO_DEPRECATED(3.0)**
Set a non-generic parameter for the joint.
- virtual void **SetDamping** (unsigned int _index, double _damping)
Set the joint damping.
- virtual void **SetForce** (unsigned int _index, double _force)
*Set the force applied to this **physics::Joint** (p. 541).*
- virtual bool **SetHighStop** (unsigned int _index, const **math::Angle** &_angle)
Set the high stop of an axis(index).
- virtual bool **SetLowStop** (unsigned int _index, const **math::Angle** &_angle)
Set the low stop of an axis(index).
- virtual bool **SetParam** (const std::string &_key, unsigned int _index, const boost::any &_value)
Set a non-generic parameter for the joint.
- virtual void **SetStiffness** (unsigned int _index, const double _stiffness)
Set the joint spring stiffness.
- virtual void **SetStiffnessDamping** (unsigned int _index, double _stiffness, double _damping, double _reference=0)
Set the joint spring stiffness.

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _force)=0
*Set the force applied to this **physics::Joint** (p. 541).*

Protected Attributes

- **DARTPhysicsPtr dartPhysicsEngine**
DARTPhysics (p. 354) engine pointer.
- dart::dynamics::BodyNode * **dtChildBodyNode**
DART child body node pointer.
- dart::dynamics::Joint * **dtJoint**
DART joint pointer.

Additional Inherited Members

10.52.1 Detailed Description

DART joint interface.

10.52.2 Constructor & Destructor Documentation

10.52.2.1 gazebo::physics::DARTJoint::DARTJoint (BasePtr *_parent*)

Constructor.

Parameters

<i>in</i>	<i>_parent</i>	Parent of the Joint (p. 541).
-----------	----------------	--------------------------------------

10.52.2.2 virtual gazebo::physics::DARTJoint::~~DARTJoint () [virtual]

Destructor.

10.52.3 Member Function Documentation

10.52.3.1 virtual void gazebo::physics::DARTJoint::ApplyDamping () [virtual]

Callback to apply damping force to joint.

Deprecated by ApplySpringStiffnessDamping.

Reimplemented from **gazebo::physics::Joint** (p. 547).

10.52.3.2 virtual bool gazebo::physics::DARTJoint::AreConnected (LinkPtr *_one*, LinkPtr *_two*) const [virtual]

Determines if the two bodies are connected by a joint.

Parameters

<i>in</i>	<i>_one</i>	First link.
<i>in</i>	<i>_two</i>	Second link.

Returns

True if the two links are connected by a joint.

Implements **gazebo::physics::Joint** (p. 548).

10.52.3.3 virtual void gazebo::physics::DARTJoint::Attach (LinkPtr *_parent*, LinkPtr *_child*) [virtual]

Attach the two bodies with this joint.

Parameters

in	<code>_parent</code>	Parent link.
in	<code>_child</code>	Child link.

Reimplemented from **gazebo::physics::Joint** (p. 548).

10.52.3.4 virtual void gazebo::physics::DARTJoint::Detach () [virtual]

Detach this joint from all links.

Reimplemented from **gazebo::physics::Joint** (p. 549).

10.52.3.5 virtual unsigned int gazebo::physics::DARTJoint::GetAngleCount () const [virtual]

Get the angle count.

Returns

The number of DOF for the joint.

Implements **gazebo::physics::Joint** (p. 550).

Reimplemented in **gazebo::physics::UniversalJoint**< **DARTJoint** > (p. 1136), **gazebo::physics::BallJoint**< **DARTJoint** > (p. 168), **gazebo::physics::SliderJoint**< **DARTJoint** > (p. 1045), **gazebo::physics::Hinge2Joint**< **DARTJoint** > (p. 513), **gazebo::physics::ScrewJoint**< **DARTJoint** > (p. 898), and **gazebo::physics::HingeJoint**< **DARTJoint** > (p. 514).

10.52.3.6 virtual double gazebo::physics::DARTJoint::GetAttribute (const std::string & *_key*, unsigned int *_index*) [virtual]

Get a non-generic parameter for the joint.

Deprecated by GetParam

Parameters

in	<code>_key</code>	String key.
in	<code>_index</code>	Index of the axis.

Implements **gazebo::physics::Joint** (p. 551).

10.52.3.7 dart::dynamics::Joint* gazebo::physics::DARTJoint::GetDARTJoint ()

Get DART joint pointer.

Returns

A pointer to the DART joint.

10.52.3.8 DARTModelPtr gazebo::physics::DARTJoint::GetDARTModel () const

Get DART model pointer.

Returns

A pointer to the DART model.

10.52.3.9 virtual double gazebo::physics::DARTJoint::GetForce (unsigned int *_index*) [virtual]

Todo : not yet implemented. Get external forces applied at this **Joint** (p. 541). Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<i>in</i>	<i>_index</i>	Index of the axis.
-----------	---------------	--------------------

Returns

The force applied to an axis.

Reimplemented from **gazebo::physics::Joint** (p. 552).

10.52.3.10 virtual **JointWrench** gazebo::physics::DARTJoint::GetForceTorque (unsigned int *_index*) [virtual]

get internal force and torque values at a joint.

The force and torque values are returned in a **JointWrench** (p. 581) data structure. Where **JointWrench.body1Force** (p. 582) contains the force applied by the parent **Link** (p. 595) on the **Joint** (p. 541) specified in the parent **Link** (p. 595) frame, and **JointWrench.body2Force** (p. 583) contains the force applied by the child **Link** (p. 595) on the **Joint** (p. 541) specified in the child **Link** (p. 595) frame. Note that this sign convention is opposite of the reaction forces of the **Joint** (p. 541) on the Links.

FIXME TODO: change name of this function to something like: GetNegatedForceTorqueInLinkFrame and make GetForceTorque call return non-negated reaction forces in perspective **Link** (p. 595) frames.

Note that for ODE you must set <provide_feedback>true</provide_feedback> in the joint sdf to use this.

Parameters

<i>in</i>	<i>_index</i>	Not used right now
-----------	---------------	--------------------

Returns

The force and torque at the joint, see above for details on conventions.

Implements **gazebo::physics::Joint** (p. 553).

10.52.3.11 virtual math::Angle gazebo::physics::DARTJoint::GetHighStop (unsigned int *_index*) [virtual]

Get the high stop of an axis(index).

This function is replaced by GetUpperLimit(unsigned int). If you are interested in getting the value of dParamHiStop*, use GetAttribute(hi_stop, *_index*)

Parameters

<i>in</i>	<i>_index</i>	Index of the axis.
-----------	---------------	--------------------

Returns

Angle of the high stop value.

Implements **gazebo::physics::Joint** (p. 553).

Reimplemented in **gazebo::physics::DARTScrewJoint** (p. 367), and **gazebo::physics::DARTBallJoint** (p. 304).

10.52.3.12 virtual **LinkPtr** gazebo::physics::DARTJoint::GetJointLink (unsigned int *_index*) const [virtual]

Get the link to which the joint is attached according the *_index*.

Parameters

<i>in</i>	<i>_index</i>	Index of the link to retrieve.
-----------	---------------	--------------------------------

Returns

Pointer to the request link. NULL if the index was invalid.

Implements **gazebo::physics::Joint** (p. 554).

10.52.3.13 virtual **math::Vector3** gazebo::physics::DARTJoint::GetLinkForce (unsigned int *_index*) const [virtual]

Get the forces applied to the center of mass of a **physics::Link** (p. 595) due to the existence of this **Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<i>in</i>	<i>index</i>	The index of the link(0 or 1).
-----------	--------------	--------------------------------

Returns

Force applied to the link.

Implements **gazebo::physics::Joint** (p. 555).

10.52.3.14 virtual **math::Vector3** gazebo::physics::DARTJoint::GetLinkTorque (unsigned int *_index*) const [virtual]

Get the torque applied to the center of mass of a **physics::Link** (p. 595) due to the existence of this **Joint** (p. 541).

Note that the unit of torque should be consistent with the rest of the simulation scales.

Parameters

<i>in</i>	<i>index</i>	The index of the link(0 or 1)
-----------	--------------	-------------------------------

Returns

Torque applied to the link.

Implements **gazebo::physics::Joint** (p. 555).

10.52.3.15 `virtual math::Angle gazebo::physics::DARTJoint::GetLowStop (unsigned int _index) [virtual]`

Get the low stop of an axis(index).

This function is replaced by `GetLowerLimit(unsigned int)`. If you are interested in getting the value of `dParamHiStop*`, use `GetAttribute(hi_stop, _index)`

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Angle of the low stop value.

Implements `gazebo::physics::Joint` (p. 556).

Reimplemented in `gazebo::physics::DARTScrewJoint` (p. 367), and `gazebo::physics::DARTBallJoint` (p. 305).

10.52.3.16 `virtual double gazebo::physics::DARTJoint::GetParam (const std::string & _key, unsigned int _index) [virtual]`

Get a non-generic parameter for the joint.

Parameters

<code>in</code>	<code>_key</code>	String key.
<code>in</code>	<code>_index</code>	Index of the axis.

Implements `gazebo::physics::Joint` (p. 557).

10.52.3.17 `virtual void gazebo::physics::DARTJoint::Init () [virtual]`

Initialize a joint.

Reimplemented from `gazebo::physics::Joint` (p. 559).

Reimplemented in `gazebo::physics::ScrewJoint< DARTJoint >` (p. 898), `gazebo::physics::UniversalJoint< DARTJoint >` (p. 1136), `gazebo::physics::BallJoint< DARTJoint >` (p. 168), `gazebo::physics::HingeJoint< DARTJoint >` (p. 514), `gazebo::physics::DARTScrewJoint` (p. 369), `gazebo::physics::DARTHinge2Joint` (p. 321), `gazebo::physics::DARTHingeJoint` (p. 326), `gazebo::physics::DARTBallJoint` (p. 306), `gazebo::physics::DARTSliderJoint` (p. 374), and `gazebo::physics::DARTUniversalJoint` (p. 381).

10.52.3.18 `virtual void gazebo::physics::DARTJoint::Load (sdf::ElementPtr _sdf) [virtual]`

Load `physics::Joint` (p. 541) from a SDF `sdf::Element`.

Parameters

<code>in</code>	<code>_sdf</code>	SDF values to load from.
-----------------	-------------------	--------------------------

Reimplemented from `gazebo::physics::Joint` (p. 560).

Reimplemented in `gazebo::physics::BallJoint< DARTJoint >` (p. 168), `gazebo::physics::UniversalJoint< DARTJoint >` (p. 1136), `gazebo::physics::Hinge2Joint< DARTJoint >` (p. 513), `gazebo::physics::ScrewJoint< DARTJoint >` (p. 898), `gazebo::physics::HingeJoint< DARTJoint >` (p. 514), `gazebo::physics::SliderJoint< DARTJoint >` (p. 374), and `gazebo::physics::UniversalJoint< DARTJoint >` (p. 381).

RTJoint > (p. 1045), **gazebo::physics::DARTHinge2Joint** (p. 321), **gazebo::physics::DARTHingeJoint** (p. 326), **gazebo::physics::DARTBallJoint** (p. 306), **gazebo::physics::DARTScrewJoint** (p. 369), **gazebo::physics::DARTSliderJoint** (p. 374), and **gazebo::physics::DARTUniversalJoint** (p. 381).

10.52.3.19 virtual void gazebo::physics::DARTJoint::Reset () [virtual]

Reset the joint.

Reimplemented from **gazebo::physics::Joint** (p. 560).

10.52.3.20 virtual void gazebo::physics::DARTJoint::SetAnchor (unsigned int, const gazebo::math::Vector3 &) [virtual]

Set the anchor point.

Implements **gazebo::physics::Joint** (p. 561).

Reimplemented in **gazebo::physics::DARTScrewJoint** (p. 369).

10.52.3.21 virtual void gazebo::physics::DARTJoint::SetAttribute (const std::string & _key, unsigned int _index, const boost::any & _value) [virtual]

Set a non-generic parameter for the joint.

replaces SetAttribute(Attribute, int, double) Deprecated by bool SetParam

Parameters

in	<i>_key</i>	String key.
in	<i>_index</i>	Index of the axis.
in	<i>_value</i>	Value of the attribute.

Implements **gazebo::physics::Joint** (p. 561).

10.52.3.22 virtual void gazebo::physics::DARTJoint::SetDamping (unsigned int *_index*, double *_damping*) [virtual]

Set the joint damping.

Parameters

in	<i>_index</i>	Index of the axis to set, currently ignored, to be implemented.
in	<i>_damping</i>	Damping value for the axis.

Implements **gazebo::physics::Joint** (p. 562).

10.52.3.23 virtual void gazebo::physics::DARTJoint::SetForce (unsigned int *_index*, double *_effort*) [virtual]

Set the force applied to this **physics::Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForce to the same joint in the same time step will accumulate forces on that **Joint** (p. 541)). Forces are truncated by effortLimit before applied.

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_effort</code>	Force value.

Implements **gazebo::physics::Joint** (p. 562).

10.52.3.24 `virtual void gazebo::physics::DARTJoint::SetForceImpl (unsigned int _index, double _force) [protected], [pure virtual]`

Set the force applied to this **physics::Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 541)).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_force</code>	Force value.

Implemented in **gazebo::physics::DARTScrewJoint** (p. 369), **gazebo::physics::DARTHinge2Joint** (p. 321), **gazebo::physics::DARTHingeJoint** (p. 326), **gazebo::physics::DARTSliderJoint** (p. 375), **gazebo::physics::DARTUniversalJoint** (p. 382), and **gazebo::physics::DARTBallJoint** (p. 306).

10.52.3.25 `virtual bool gazebo::physics::DARTJoint::SetHighStop (unsigned int _index, const math::Angle & _angle) [virtual]`

Set the high stop of an axis(index).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_angle</code>	High stop angle.

Reimplemented from **gazebo::physics::Joint** (p. 562).

Reimplemented in **gazebo::physics::DARTBallJoint** (p. 307).

10.52.3.26 `virtual bool gazebo::physics::DARTJoint::SetLowStop (unsigned int _index, const math::Angle & _angle) [virtual]`

Set the low stop of an axis(index).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_angle</code>	Low stop angle.

Reimplemented from **gazebo::physics::Joint** (p. 563).

Reimplemented in **gazebo::physics::DARTBallJoint** (p. 307).

10.52.3.27 `virtual bool gazebo::physics::DARTJoint::SetParam (const std::string & _key, unsigned int _index, const boost::any & _value) [virtual]`

Set a non-generic parameter for the joint.

replaces SetAttribute(Attribute, int, double)

Parameters

in	<code>_key</code>	String key.
in	<code>_index</code>	Index of the axis.
in	<code>_value</code>	Value of the attribute.

Implements `gazebo::physics::Joint` (p. 564).

10.52.3.28 `virtual void gazebo::physics::DARTJoint::SetStiffness (unsigned int _index, const double _stiffness) [virtual]`

Set the joint spring stiffness.

Parameters

in	<code>_index</code>	Index of the axis to set, currently ignored, to be implemented.
in	<code>_stiffness</code>	Spring stiffness value for the axis. : rename to SetSpringStiffness()

Implements `gazebo::physics::Joint` (p. 564).

10.52.3.29 `virtual void gazebo::physics::DARTJoint::SetStiffnessDamping (unsigned int _index, double _stiffness, double _damping, double _reference = 0) [virtual]`

Set the joint spring stiffness.

Parameters

in	<code>_index</code>	Index of the axis to set, currently ignored, to be implemented.
in	<code>_stiffness</code>	Stiffness value for the axis.
in	<code>_reference</code>	Spring zero load reference position. : rename to SetSpringStiffnessDamping()

Implements `gazebo::physics::Joint` (p. 565).

10.52.4 Member Data Documentation

10.52.4.1 `DARTPhysicsPtr gazebo::physics::DARTJoint::dartPhysicsEngine` [protected]

DARTPhysics (p. 354) engine pointer.

10.52.4.2 `dart::dynamics::BodyNode* gazebo::physics::DARTJoint::dtChildBodyNode` [protected]

DART child body node pointer.

10.52.4.3 dart::dynamics::Joint* gazebo::physics::DARTJoint::dtJoint [protected]

DART joint pointer.

The documentation for this class was generated from the following file:

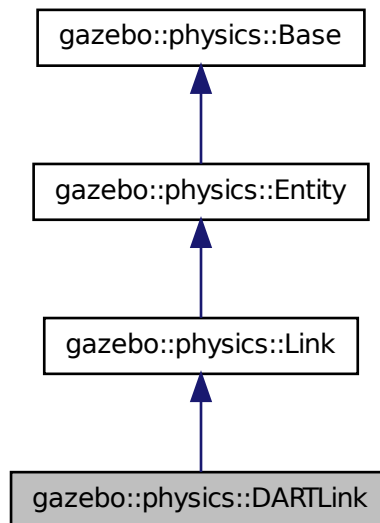
- **DARTJoint.hh**

10.53 gazebo::physics::DARTLink Class Reference

DART **Link** (p. 595) class.

```
#include <DARTLink.hh>
```

Inheritance diagram for gazebo::physics::DARTLink:



Public Member Functions

- **DARTLink** (**EntityPtr** _parent)
Constructor.
- virtual **~DARTLink** ()
Destructor.
- void **AddDARTChildJoint** (**DARTJointPtr** _dartChildJoint)
Set child joint of this link.
- virtual void **AddForce** (const **math::Vector3** &_force)
Add a force to the body.
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relpos)

- Add a force to the body at position expressed to the body's own frame of reference.*

 - virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)

Add a force to the body using a global position.

 - virtual void **AddRelativeForce** (const **math::Vector3** &_force)

Add a force to the body, components are relative to the body's own frame of reference.

 - virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)

Add a torque to the body, components are relative to the body's own frame of reference.

 - virtual void **AddTorque** (const **math::Vector3** &_torque)

Add a torque to the body.

 - virtual void **Fini** ()

Finalize the body.

 - dart::dynamics::BodyNode * **GetDARTBodyNode** () const

Get pointer to DART BodyNode associated with this link.

 - **DARTModelPtr GetDARTModel** () const

*Get pointer to DART **Model** (p. 678) associated with this link.*

 - **DARTPhysicsPtr GetDARTPhysics** (void) const

Get pointer to DART Physics engine associated with this link.

 - dart::simulation::World * **GetDARTWorld** (void) const

*Get pointer to DART **World** (p. 1239) associated with this link.*

 - virtual bool **GetEnabled** () const

Get whether this body is enabled in the physics engine.

 - virtual bool **GetGravityMode** () const

Get the gravity mode.

 - virtual bool **GetKinematic** () const

Implement this function.

 - virtual **math::Vector3 GetWorldAngularVel** () const

Get the angular velocity of the entity in the world frame.

 - virtual **math::Vector3 GetWorldCoGLinearVel** () const

Get the linear velocity at the body's center of gravity in the world frame.

 - virtual **math::Vector3 GetWorldForce** () const

Get the force applied to the body in the world frame.

 - virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset=**math::Vector3**(0, 0, 0)) const

Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame.

 - virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset, const **math::Quaternion** &_q) const

Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.

 - virtual **math::Vector3 GetWorldTorque** () const

Get the torque applied to the body in the world frame.

 - virtual void **Init** ()

Initialize the body.

 - virtual void **Load** (sdf::ElementPtr _ptr)

Load the body based on an SDF element.

 - virtual void **OnPoseChange** ()

This function is called when the entity's (or one of its parents) pose of the parent has changed.

 - virtual void **SetAngularDamping** (double _damping)

Set the angular damping factor.

 - virtual void **SetAngularVel** (const **math::Vector3** &_vel)

Set the angular velocity of the body.

- virtual void **SetAutoDisable** (bool _disable)
Allow the link to auto disable.
- void **SetDARTParentJoint** (DARTJointPtr _dartParentJoint)
Set parent joint of this link.
- virtual void **SetEnabled** (bool _enable) const
Set whether this body is enabled.
- virtual void **SetForce** (const math::Vector3 &_force)
Set the force applied to the body.
- virtual void **SetGravityMode** (bool _mode)
Set whether gravity affects this body.
- virtual void **SetKinematic** (const bool &_state)
Implement this function.
- virtual void **SetLinearDamping** (double _damping)
Set the linear damping factor.
- virtual void **SetLinearVel** (const math::Vector3 &_vel)
Set the linear velocity of the body.
- virtual void **SetLinkStatic** (bool _static)
Freeze link to ground (inertial frame).
- virtual void **SetSelfCollide** (bool _collide)
Set whether this body will collide with others in the model.
- virtual void **SetTorque** (const math::Vector3 &_torque)
Set the torque applied to the body.
- void **updateDirtyPoseFromDARTTransformation** ()
*Store DART Transformation to **Entity::dirtyPose** (p. 415) and add this link to **World::dirtyPoses** (p. 1251) so that **World::Update()** trigger **Entity::SetWorldPose()** (p. 414) for this link.*

Additional Inherited Members

10.53.1 Detailed Description

DART **Link** (p. 595) class.

10.53.2 Constructor & Destructor Documentation

10.53.2.1 gazebo::physics::DARTLink::DARTLink (EntityPtr _parent) [explicit]

Constructor.

10.53.2.2 virtual gazebo::physics::DARTLink::~DARTLink () [virtual]

Destructor.

10.53.3 Member Function Documentation

10.53.3.1 void gazebo::physics::DARTLink::AddDARTChildJoint (DARTJointPtr *_dartChildJoint*)

Set child joint of this link.

Parameters

in	<i>_dartChildJoint</i>	Pointer to the child joint.
----	------------------------	-----------------------------

10.53.3.2 virtual void gazebo::physics::DARTLink::AddForce (const math::Vector3 & *_force*) [virtual]

Add a force to the body.

Parameters

in	<i>_force</i>	Force to add.
----	---------------	---------------

Implements **gazebo::physics::Link** (p. 601).

10.53.3.3 virtual void gazebo::physics::DARTLink::AddForceAtRelativePosition (const math::Vector3 & *_force*, const math::Vector3 & *_relPos*) [virtual]

Add a force to the body at position expressed to the body's own frame of reference.

Parameters

in	<i>_force</i>	Force to add.
in	<i>_relPos</i>	Position on the link to add the force.

Implements **gazebo::physics::Link** (p. 601).

10.53.3.4 virtual void gazebo::physics::DARTLink::AddForceAtWorldPosition (const math::Vector3 & *_force*, const math::Vector3 & *_pos*) [virtual]

Add a force to the body using a global position.

Parameters

in	<i>_force</i>	Force to add.
in	<i>_pos</i>	Position in global coord frame to add the force.

Implements **gazebo::physics::Link** (p. 602).

10.53.3.5 virtual void gazebo::physics::DARTLink::AddRelativeForce (const math::Vector3 & *_force*) [virtual]

Add a force to the body, components are relative to the body's own frame of reference.

Parameters

in	<i>_force</i>	Force to add.
----	---------------	---------------

Implements **gazebo::physics::Link** (p. 602).

10.53.3.6 `virtual void gazebo::physics::DARTLink::AddRelativeTorque (const math::Vector3 & _torque) [virtual]`

Add a torque to the body, components are relative to the body's own frame of reference.

Parameters

<code>in</code>	<code>_torque</code>	Torque value to add.
-----------------	----------------------	----------------------

Implements **gazebo::physics::Link** (p. 602).

10.53.3.7 `virtual void gazebo::physics::DARTLink::AddTorque (const math::Vector3 & _torque) [virtual]`

Add a torque to the body.

Parameters

<code>in</code>	<code>_torque</code>	Torque value to add to the link.
-----------------	----------------------	----------------------------------

Implements **gazebo::physics::Link** (p. 602).

10.53.3.8 `virtual void gazebo::physics::DARTLink::Fini () [virtual]`

Finalize the body.

Reimplemented from **gazebo::physics::Link** (p. 604).

10.53.3.9 `dart::dynamics::BodyNode* gazebo::physics::DARTLink::GetDARTBodyNode () const`

Get pointer to DART BodyNode associated with this link.

Returns

Pointer to DART BodyNode.

10.53.3.10 `DARTModelPtr gazebo::physics::DARTLink::GetDARTModel () const`

Get pointer to DART **Model** (p. 678) associated with this link.

Returns

Pointer to the DART **Model** (p. 678).

10.53.3.11 `DARTPhysicsPtr gazebo::physics::DARTLink::GetDARTPhysics (void) const`

Get pointer to DART Physics engine associated with this link.

Returns

Pointer to the DART Physics engine.

10.53.3.12 `dart::simulation::World*` gazebo::physics::DARTLink::GetDARTWorld (void) const

Get pointer to DART **World** (p. 1239) associated with this link.

Returns

Pointer to the DART **World** (p. 1239).

10.53.3.13 `virtual bool` gazebo::physics::DARTLink::GetEnabled () const [virtual]

Get whether this body is enabled in the physics engine.

Returns

True if the link is enabled.

Implements **gazebo::physics::Link** (p. 605).

10.53.3.14 `virtual bool` gazebo::physics::DARTLink::GetGravityMode () const [virtual]

Get the gravity mode.

Returns

True if gravity is enabled.

Implements **gazebo::physics::Link** (p. 606).

10.53.3.15 `virtual bool` gazebo::physics::DARTLink::GetKinematic () const [virtual]

Implement this function.

Get whether this body is in the kinematic state.

Returns

True if the link is kinematic only.

Reimplemented from **gazebo::physics::Link** (p. 606).

10.53.3.16 `virtual math::Vector3` gazebo::physics::DARTLink::GetWorldAngularVel () const [virtual]

Get the angular velocity of the entity in the world frame.

Returns

A **math::Vector3** (p. 1165) for the velocity.

Reimplemented from **gazebo::physics::Entity** (p. 410).

10.53.3.17 `virtual math::Vector3 gazebo::physics::DARTLink::GetWorldCoGLinearVel () const [virtual]`

Get the linear velocity at the body's center of gravity in the world frame.

Returns

Linear velocity at the body's center of gravity in the world frame.

Implements `gazebo::physics::Link` (p. 609).

10.53.3.18 `virtual math::Vector3 gazebo::physics::DARTLink::GetWorldForce () const [virtual]`

Get the force applied to the body in the world frame.

Returns

Force applied to the body in the world frame.

Implements `gazebo::physics::Link` (p. 610).

10.53.3.19 `virtual math::Vector3 gazebo::physics::DARTLink::GetWorldLinearVel (const math::Vector3 & _offset = math::Vector3(0, 0, 0)) const [virtual]`

Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame.

If no offset is given, the velocity at the origin of the `Link` (p. 595) frame will be returned.

Parameters

<code>in</code>	<code>_offset</code>	Offset of the point from the origin of the <code>Link</code> (p. 595) frame, expressed in the body-fixed frame.
-----------------	----------------------	---

Returns

Linear velocity of the point on the body

Implements `gazebo::physics::Link` (p. 611).

10.53.3.20 `virtual math::Vector3 gazebo::physics::DARTLink::GetWorldLinearVel (const math::Vector3 & _offset, const math::Quaternion & _q) const [virtual]`

Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.

Parameters

<code>in</code>	<code>_offset</code>	Offset from the origin of the link frame expressed in a frame defined by <code>_q</code> .
<code>in</code>	<code>_q</code>	Describes the rotation of a reference frame relative to the world reference frame.

Returns

Linear velocity of the point on the body in the world frame.

Implements `gazebo::physics::Link` (p. 611).

10.53.3.21 `virtual math::Vector3 gazebo::physics::DARTLink::GetWorldTorque () const [virtual]`

Get the torque applied to the body in the world frame.

Returns

Torque applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 611).

10.53.3.22 `virtual void gazebo::physics::DARTLink::Init () [virtual]`

Initialize the body.

Reimplemented from **gazebo::physics::Link** (p. 611).

10.53.3.23 `virtual void gazebo::physics::DARTLink::Load (sdf::ElementPtr _sdf) [virtual]`

Load the body based on an SDF element.

Parameters

<code>in</code>	<code>_sdf</code>	SDF parameters.
-----------------	-------------------	-----------------

Reimplemented from **gazebo::physics::Link** (p. 612).

10.53.3.24 `virtual void gazebo::physics::DARTLink::OnPoseChange () [virtual]`

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Reimplemented from **gazebo::physics::Link** (p. 612).

10.53.3.25 `virtual void gazebo::physics::DARTLink::SetAngularDamping (double _damping) [virtual]`

Set the angular damping factor.

Parameters

<code>in</code>	<code>_damping</code>	Angular damping factor.
-----------------	-----------------------	-------------------------

Implements **gazebo::physics::Link** (p. 613).

10.53.3.26 `virtual void gazebo::physics::DARTLink::SetAngularVel (const math::Vector3 & _vel) [virtual]`

Set the angular velocity of the body.

Parameters

<code>in</code>	<code>_vel</code>	Angular velocity.
-----------------	-------------------	-------------------

Implements **gazebo::physics::Link** (p. 613).

10.53.3.27 `virtual void gazebo::physics::DARTLink::SetAutoDisable (bool _disable) [virtual]`

Allow the link to auto disable.

Parameters

<code>in</code>	<code>_disable</code>	If true, the link is allowed to auto disable.
-----------------	-----------------------	---

Implements `gazebo::physics::Link` (p. 613).

10.53.3.28 `void gazebo::physics::DARTLink::SetDARTParentJoint (DARTJointPtr _dartParentJoint)`

Set parent joint of this link.

Parameters

<code>in</code>	<code>_dartParentJoint</code>	Pointer to the parent joint.
-----------------	-------------------------------	------------------------------

10.53.3.29 `virtual void gazebo::physics::DARTLink::SetEnabled (bool _enable) const [virtual]`

Set whether this body is enabled.

Parameters

<code>in</code>	<code>_enable</code>	True to enable the link in the physics engine.
-----------------	----------------------	--

Implements `gazebo::physics::Link` (p. 614).

10.53.3.30 `virtual void gazebo::physics::DARTLink::SetForce (const math::Vector3 & _force) [virtual]`

Set the force applied to the body.

Parameters

<code>in</code>	<code>_force</code>	Force value.
-----------------	---------------------	--------------

Implements `gazebo::physics::Link` (p. 614).

10.53.3.31 `virtual void gazebo::physics::DARTLink::SetGravityMode (bool _mode) [virtual]`

Set whether gravity affects this body.

Parameters

<code>in</code>	<code>_mode</code>	True to enable gravity.
-----------------	--------------------	-------------------------

Implements `gazebo::physics::Link` (p. 614).

10.53.3.32 `virtual void gazebo::physics::DARTLink::SetKinematic (const bool & _kinematic) [virtual]`

Implement this function.

Set whether this body is in the kinematic state.

Parameters

<code>in</code>	<code>_kinematic</code>	True to make the link kinematic only.
-----------------	-------------------------	---------------------------------------

Reimplemented from **gazebo::physics::Link** (p. 615).

10.53.3.33 virtual void gazebo::physics::DARTLink::SetLinearDamping (double *_damping*) [virtual]

Set the linear damping factor.

Parameters

<code>in</code>	<code>_damping</code>	Linear damping factor.
-----------------	-----------------------	------------------------

Implements **gazebo::physics::Link** (p. 615).

10.53.3.34 virtual void gazebo::physics::DARTLink::SetLinearVel (const math::Vector3 & *_vel*) [virtual]

Set the linear velocity of the body.

Parameters

<code>in</code>	<code>_vel</code>	Linear velocity.
-----------------	-------------------	------------------

Implements **gazebo::physics::Link** (p. 615).

10.53.3.35 virtual void gazebo::physics::DARTLink::SetLinkStatic (bool *_static*) [virtual]

Freeze link to ground (inertial frame).

Parameters

<code>in</code>	<code>_static</code>	if true, freeze link to ground. Otherwise unfreeze link.
-----------------	----------------------	--

Implements **gazebo::physics::Link** (p. 615).

10.53.3.36 virtual void gazebo::physics::DARTLink::SetSelfCollide (bool *_collide*) [virtual]

Set whether this body will collide with others in the model.

Parameters

<code>in</code>	<code>_collid</code>	True to enable collisions.
-----------------	----------------------	----------------------------

Implements **gazebo::physics::Link** (p. 616).

10.53.3.37 virtual void gazebo::physics::DARTLink::SetTorque (const math::Vector3 & *_torque*) [virtual]

Set the torque applied to the body.

Parameters

in	<code>_torque</code>	Torque value.
----	----------------------	---------------

Implements **gazebo::physics::Link** (p. 617).

10.53.3.38 void gazebo::physics::DARTLink::updateDirtyPoseFromDARTTransformation ()

Store DART Transformation to **Entity::dirtyPose** (p. 415) and add this link to **World::dirtyPoses** (p. 1251) so that World::Update() trigger **Entity::SetWorldPose()** (p. 414) for this link.

The documentation for this class was generated from the following file:

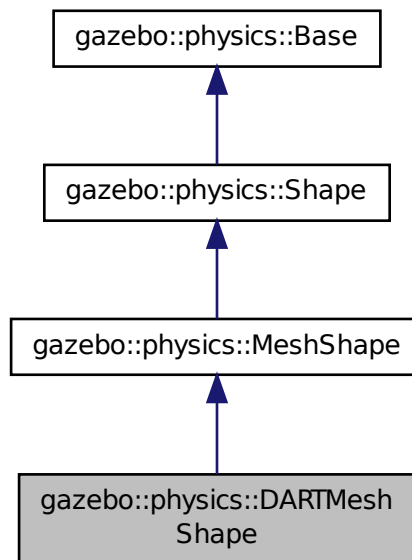
- **DARTLink.hh**

10.54 gazebo::physics::DARTMeshShape Class Reference

Triangle mesh collision.

```
#include <DARTMeshShape.hh>
```

Inheritance diagram for gazebo::physics::DARTMeshShape:



Public Member Functions

- **DARTMeshShape (CollisionPtr _parent)**
Constructor.

- virtual `~DARTMeshShape ()`
Destructor.
- virtual void `Init ()`
Initialize the shape.
- virtual void `Load (sdf::ElementPtr _sdf)`
Load.
- virtual void `Update ()`
Update the tri mesh.

Additional Inherited Members

10.54.1 Detailed Description

Triangle mesh collision.

10.54.2 Constructor & Destructor Documentation

10.54.2.1 gazebo::physics::DARTMeshShape::DARTMeshShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

<code>in</code>	<code>_parent</code>	Parent collision object.
-----------------	----------------------	--------------------------

10.54.2.2 virtual gazebo::physics::DARTMeshShape::~~DARTMeshShape () [virtual]

Destructor.

10.54.3 Member Function Documentation

10.54.3.1 virtual void gazebo::physics::DARTMeshShape::Init () [virtual]

Initialize the shape.

Reimplemented from `gazebo::physics::MeshShape` (p. 677).

10.54.3.2 virtual void gazebo::physics::DARTMeshShape::Load (sdf::ElementPtr *_sdf*) [virtual]

Load.

Parameters

<code>in</code>	<code>node</code>	Pointer to an SDF parameters
-----------------	-------------------	------------------------------

Reimplemented from `gazebo::physics::Base` (p. 177).

10.54.3.3 `virtual void gazebo::physics::DARTMeshShape::Update () [virtual]`

Update the tri mesh.

Reimplemented from `gazebo::physics::MeshShape` (p. 678).

The documentation for this class was generated from the following file:

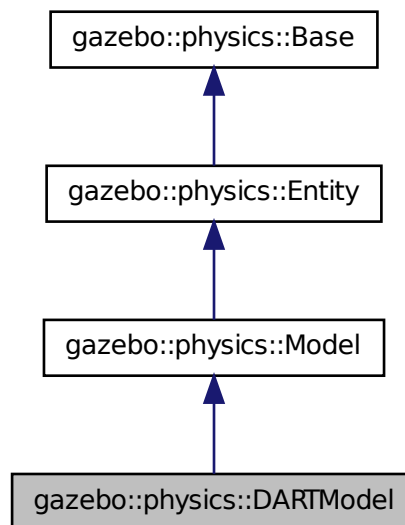
- `DARTMeshShape.hh`

10.55 gazebo::physics::DARTModel Class Reference

DART model class.

```
#include <DARTModel.hh>
```

Inheritance diagram for `gazebo::physics::DARTModel`:



Public Member Functions

- **DARTModel** (`BasePtr _parent`)
Constructor.
- `virtual ~DARTModel ()`
Destructor.
- `void BackupState ()`
- `virtual void Fini ()`
Finalize the model.
- `DARTPhysicsPtr GetDARTPhysics (void) const`

- dart::dynamics::Skeleton * **GetDARTSkeleton** ()
- dart::simulation::World * **GetDARTWorld** (void) const
- virtual void **Init** ()
Initialize the model.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the model.
- void **RestoreState** ()
- virtual void **Update** ()
Update the model.

Protected Attributes

- Eigen::VectorXd **dtConfig**
- dart::dynamics::Skeleton * **dtSkeleton**
- Eigen::VectorXd **dtVelocity**

Additional Inherited Members

10.55.1 Detailed Description

DART model class.

10.55.2 Constructor & Destructor Documentation

10.55.2.1 gazebo::physics::DARTModel::DARTModel (BasePtr *_parent*) [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent object.
----	----------------	----------------

10.55.2.2 virtual gazebo::physics::DARTModel::~~DARTModel () [virtual]

Destructor.

10.55.3 Member Function Documentation

10.55.3.1 void gazebo::physics::DARTModel::BackupState ()

10.55.3.2 virtual void gazebo::physics::DARTModel::Fini () [virtual]

Finalize the model.

Reimplemented from **gazebo::physics::Model** (p. 683).

10.55.3.3 DARTPhysicsPtr gazebo::physics::DARTModel::GetDARTPhysics (void) const

10.55.3.4 `dart::dynamics::Skeleton*` gazebo::physics::DARTModel::GetDARTSkeleton ()

10.55.3.5 `dart::simulation::World*` gazebo::physics::DARTModel::GetDARTWorld (void) const

10.55.3.6 `virtual void` gazebo::physics::DARTModel::Init () [virtual]

Initialize the model.

Reimplemented from `gazebo::physics::Model` (p. 688).

10.55.3.7 `virtual void` gazebo::physics::DARTModel::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the model.

Parameters

<code>in</code>	<code>_sdf</code>	SDF parameters to load from.
-----------------	-------------------	------------------------------

Reimplemented from `gazebo::physics::Model` (p. 688).

10.55.3.8 `void` gazebo::physics::DARTModel::RestoreState ()

10.55.3.9 `virtual void` gazebo::physics::DARTModel::Update () [virtual]

Update the model.

Reimplemented from `gazebo::physics::Model` (p. 692).

10.55.4 Member Data Documentation

10.55.4.1 `Eigen::VectorXd` gazebo::physics::DARTModel::dtConfig [protected]

10.55.4.2 `dart::dynamics::Skeleton*` gazebo::physics::DARTModel::dtSkeleton [protected]

10.55.4.3 `Eigen::VectorXd` gazebo::physics::DARTModel::dtVelocity [protected]

The documentation for this class was generated from the following file:

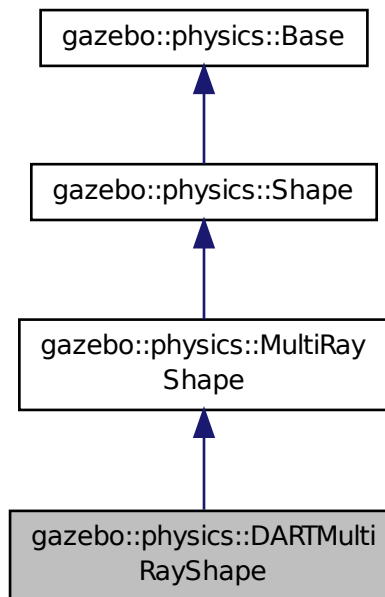
- **DARTModel.hh**

10.56 gazebo::physics::DARTMultiRayShape Class Reference

DART specific version of `MultiRayShape` (p. 723).

```
#include <DARTMultiRayShape.hh>
```

Inheritance diagram for gazebo::physics::DARTMultiRayShape:



Public Member Functions

- **DARTMultiRayShape** (*CollisionPtr* _parent)
Constructor.
- virtual \sim **DARTMultiRayShape** ()
Destructor.
- virtual void **UpdateRays** ()
Physics engine specific method for updating the rays.

Protected Member Functions

- void **AddRay** (const **math::Vector3** &_start, const **math::Vector3** &_end)
Add a ray to the collision.

Additional Inherited Members

10.56.1 Detailed Description

DART specific version of **MultiRayShape** (p. 723).

The documentation for this class was generated from the following file:

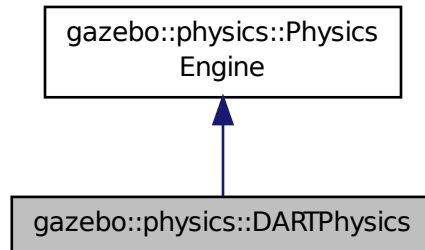
- **DARTMultiRayShape.hh**

10.57 gazebo::physics::DARTPhysics Class Reference

DART physics engine.

```
#include <DARTPhysics.hh>
```

Inheritance diagram for gazebo::physics::DARTPhysics:



Public Types

- enum **DARTParam** { **MAX_CONTACTS**, **MIN_STEP_SIZE** }
DART physics parameter types.

Public Member Functions

- **DARTPhysics** (**WorldPtr** _world)
Constructor.
- virtual **~DARTPhysics** ()
Destructor.
- virtual **CollisionPtr** **CreateCollision** (const std::string &_type, **LinkPtr** _body)
Create a collision.
- virtual **JointPtr** **CreateJoint** (const std::string &_type, **ModelPtr** _parent)
Create a new joint.
- virtual **LinkPtr** **CreateLink** (**ModelPtr** _parent)
Create a new body.
- virtual **ModelPtr** **CreateModel** (**BasePtr** _parent)
Create a new model.
- virtual **ShapePtr** **CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)
*Create a **physics::Shape** (p. 932) object.*
- virtual void **DebugPrint** () const
Debug print out of the physic engine state.

- virtual void **Fini** ()
Finilize the physics engine.
- dart::simulation::World * **GetDARTWorld** ()
*Get pointer to DART **World** (p. 1239) associated with this DART Physics.*
- virtual boost::any **GetParam** (const std::string &_key) const
Get an parameter of the physics engine.
- virtual boost::any **GetParam** (DARTParam _param) const **GAZEBO_DEPRECATED(3.0)**
Get an parameter of the physics engine.
- virtual std::string **GetType** () const
Return the physics engine type (ode|bullet|dart|simbody).
- virtual void **Init** ()
Initialize the physics engine.
- virtual void **InitForThread** ()
Init the engine for threads.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the physics engine.
- virtual void **Reset** ()
Rest the physics engine.
- virtual void **SetGravity** (const gazebo::math::Vector3 &_gravity)
Set the gavity vector.
- virtual bool **SetParam** (const std::string &_key, const boost::any &_value)
Set a parameter of the physics engine.
- virtual void **SetSeed** (uint32_t _seed)
Set the random number seed for the physics engine.
- virtual void **UpdateCollision** ()
Update the physics engine collision.
- virtual void **UpdatePhysics** ()
Update the physics engine.

Protected Member Functions

- virtual void **OnPhysicsMsg** (ConstPhysicsPtr &_msg)
virtual callback for gztopic "~/physics".
- virtual void **OnRequest** (ConstRequestPtr &_msg)
virtual callback for gztopic "~/request".

Additional Inherited Members

10.57.1 Detailed Description

DART physics engine.

10.57.2 Member Enumeration Documentation

10.57.2.1 enum gazebo::physics::DARTPhysics::DARTParam

DART physics parameter types.

Enumerator:

MAX_CONTACTS Maximum number of contacts.

MIN_STEP_SIZE Minimum step size.

10.57.3 Constructor & Destructor Documentation

10.57.3.1 gazebo::physics::DARTPhysics::DARTPhysics (WorldPtr *_world*)

Constructor.

10.57.3.2 virtual gazebo::physics::DARTPhysics::~~DARTPhysics () [virtual]

Destructor.

10.57.4 Member Function Documentation

10.57.4.1 virtual CollisionPtr gazebo::physics::DARTPhysics::CreateCollision (const std::string & *_shapeType*, LinkPtr *_link*) [virtual]

Create a collision.

Parameters

in	<i>_shapeType</i>	Type of collision to create.
in	<i>_link</i>	Parent link.

Implements **gazebo::physics::PhysicsEngine** (p. 770).

10.57.4.2 virtual JointPtr gazebo::physics::DARTPhysics::CreateJoint (const std::string & *_type*, ModelPtr *_parent*) [virtual]

Create a new joint.

Parameters

in	<i>_type</i>	Type of joint to create.
in	<i>_parent</i>	Model (p. 678) parent.

Implements **gazebo::physics::PhysicsEngine** (p. 771).

10.57.4.3 virtual LinkPtr gazebo::physics::DARTPhysics::CreateLink (ModelPtr *_parent*) [virtual]

Create a new body.

Parameters

in	<code>_parent</code>	Parent model for the link.
----	----------------------	----------------------------

Implements **gazebo::physics::PhysicsEngine** (p. 771).

10.57.4.4 `virtual ModelPtr gazebo::physics::DARTPhysics::CreateModel (BasePtr _base) [virtual]`

Create a new model.

Parameters

in	<code>_base</code>	Boost shared pointer to a new model.
----	--------------------	--------------------------------------

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 771).

10.57.4.5 `virtual ShapePtr gazebo::physics::DARTPhysics::CreateShape (const std::string & _shapeType, CollisionPtr _collision) [virtual]`

Create a **physics::Shape** (p. 932) object.

Parameters

in	<code>_shapeType</code>	Type of shape to create.
in	<code>_collision</code>	Collision (p. 235) parent.

Implements **gazebo::physics::PhysicsEngine** (p. 771).

10.57.4.6 `virtual void gazebo::physics::DARTPhysics::DebugPrint () const [virtual]`

Debug print out of the physic engine state.

Implements **gazebo::physics::PhysicsEngine** (p. 772).

10.57.4.7 `virtual void gazebo::physics::DARTPhysics::Fini () [virtual]`

Finilize the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 772).

10.57.4.8 `dart::simulation::World* gazebo::physics::DARTPhysics::GetDARTWorld ()`

Get pointer to DART **World** (p. 1239) associated with this DART Physics.

Returns

The pointer to DART **World** (p. 1239).

10.57.4.9 `virtual boost::any gazebo::physics::DARTPhysics::GetParam (const std::string & _key) const [virtual]`

Get an parameter of the physics engine.

Parameters

in	<code>_attr</code>	String key
----	--------------------	------------

See Also

SetParam (p. 360)

Returns

The value of the parameter

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 773).

10.57.4.10 virtual boost::any gazebo::physics::DARTPhysics::GetParam (DARTParam `_param`) const [virtual]

Get an parameter of the physics engine.

See Also

boost::any **GetParam(const std::string &`_key`) const** (p. 357)

Parameters

in	<code>_param</code>	A parameter listed in the ODEParam enum
----	---------------------	---

Returns

The value of the parameter

10.57.4.11 virtual std::string gazebo::physics::DARTPhysics::GetType () const [virtual]

Return the physics engine type (ode|bullet|dart|simbody).

Returns

Type of the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 774).

10.57.4.12 virtual void gazebo::physics::DARTPhysics::Init () [virtual]

Initialize the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 775).

10.57.4.13 virtual void gazebo::physics::DARTPhysics::InitForThread () [virtual]

Init the engine for threads.

Implements **gazebo::physics::PhysicsEngine** (p. 775).

10.57.4.14 `virtual void gazebo::physics::DARTPhysics::Load (sdf::ElementPtr _sdf) [virtual]`

Load the physics engine.

Parameters

in	_sdf	Pointer to the SDF parameters.
----	------	--------------------------------

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 775).

10.57.4.15 `virtual void gazebo::physics::DARTPhysics::OnPhysicsMsg (ConstPhysicsPtr & _msg) [protected], [virtual]`

virtual callback for gztopic "~/physics".

Parameters

in	_msg	Physics message.
----	------	------------------

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 776).

10.57.4.16 `virtual void gazebo::physics::DARTPhysics::OnRequest (ConstRequestPtr & _msg) [protected], [virtual]`

virtual callback for gztopic "~/request".

Parameters

in	_msg	Request message.
----	------	------------------

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 776).

10.57.4.17 `virtual void gazebo::physics::DARTPhysics::Reset () [virtual]`

Rest the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 776).

10.57.4.18 `virtual void gazebo::physics::DARTPhysics::SetGravity (const gazebo::math::Vector3 & _gravity) [virtual]`

Set the gravity vector.

Parameters

in	_gravity	New gravity vector.
----	----------	---------------------

Implements **gazebo::physics::PhysicsEngine** (p. 777).

10.57.4.19 `virtual bool gazebo::physics::DARTPhysics::SetParam (const std::string & _key, const boost::any & _value)`
`[virtual]`

Set a parameter of the physics engine.

See SetParam documentation for descriptions of duplicate parameters.

Parameters

<code>in</code>	<code>_key</code>	String key Below is a list of <code>_key</code> parameter definitions: <ol style="list-style-type: none"> 1. "solver_type" (string) - returns solver used by engine, e.g. "sequential_impulse" for Bullet, "quick" for ODE "Featherstone and Lemkes" for DART and "Spatial Algebra and Elastic Foundation" for Simbody. -# "type" (string) - deprecated, use keyword "solver_type". -# "cfm" (double) - global CFM -# "erp" (double) - global ERP -# "precon_iters" (bool) - precondition iterations (experimental). 2. "iters" (int) - number of LCP PGS iterations. If <code>sor_lcp_tolerance</code> is negative, full iteration count is executed. Otherwise, PGS may stop iteration early if <code>sor_lcp_tolerance</code> is satisfied by the total RMS residual. 3. "sor" (double) - relaxation parameter for Projected Gauss-Seidel (PGS) updates. 4. "contact_max_correcting_vel" (double) - truncates correction impulses from ERP by this value. 5. "contact_surface_layer" (double) - ERP is 0 for interpenetration depths below this value. 6. "max_contacts" (int) - max number of contact constraints between any pair of collision bodies. 7. "min_step_size" (double) - minimum internal step size. (defined but not used in ode). 8. "max_step_size" (double) - maximum physics step size when physics update step must return.
<code>in</code>	<code>_value</code>	The value to set to

Returns

true if SetParam is successful, false if operation fails.

Reimplemented from `gazebo::physics::PhysicsEngine` (p. 777).

10.57.4.20 `virtual void gazebo::physics::DARTPhysics::SetSeed (uint32_t _seed)` `[virtual]`

Set the random number seed for the physics engine.

Parameters

<code>in</code>	<code>_seed</code>	The random number seed.
-----------------	--------------------	-------------------------

Implements `gazebo::physics::PhysicsEngine` (p. 778).

10.57.4.21 virtual void gazebo::physics::DARTPhysics::UpdateCollision () [virtual]

Update the physics engine collision.

Implements **gazebo::physics::PhysicsEngine** (p. 780).

10.57.4.22 virtual void gazebo::physics::DARTPhysics::UpdatePhysics () [virtual]

Update the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 780).

The documentation for this class was generated from the following file:

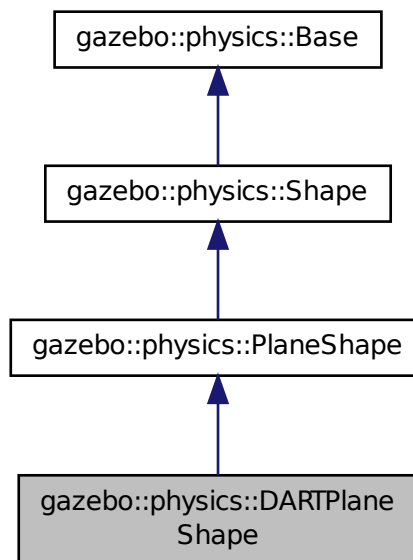
- **DARTPhysics.hh**

10.58 gazebo::physics::DARTPlaneShape Class Reference

An DART Plane shape.

```
#include <DARTPlaneShape.hh>
```

Inheritance diagram for gazebo::physics::DARTPlaneShape:



Public Member Functions

- **DARTPlaneShape** (**CollisionPtr** _parent)

Constructor.

- virtual `~DARTPlaneShape ()`

Destructor.

- virtual void `CreatePlane ()`

Create the plane.

- virtual void `SetAltitude (const math::Vector3 &_pos)`

Set the altitude of the plane.

Additional Inherited Members

10.58.1 Detailed Description

An DART Plane shape.

10.58.2 Constructor & Destructor Documentation

10.58.2.1 `gazebo::physics::DARTPlaneShape::DARTPlaneShape (CollisionPtr _parent) [inline],[explicit]`

Constructor.

Parameters

<code>in</code>	<code>_parent</code>	Parent Collision (p. 235).
-----------------	----------------------	-----------------------------------

10.58.2.2 `virtual gazebo::physics::DARTPlaneShape::~~DARTPlaneShape () [inline],[virtual]`

Destructor.

10.58.3 Member Function Documentation

10.58.3.1 `virtual void gazebo::physics::DARTPlaneShape::CreatePlane () [inline],[virtual]`

Create the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 792).

References `gazebo::physics::PlaneShape::CreatePlane()`, and `gazebo::physics::DARTCollision::GetDARTBodyNode()`.

10.58.3.2 `virtual void gazebo::physics::DARTPlaneShape::SetAltitude (const math::Vector3 &_pos) [inline],[virtual]`

Set the altitude of the plane.

Parameters

<code>in</code>	<code>_pos</code>	Position of the plane.
-----------------	-------------------	------------------------

Reimplemented from **gazebo::physics::PlaneShape** (p. 793).

References `gazebo::physics::PlaneShape::SetAltitude()`.

The documentation for this class was generated from the following file:

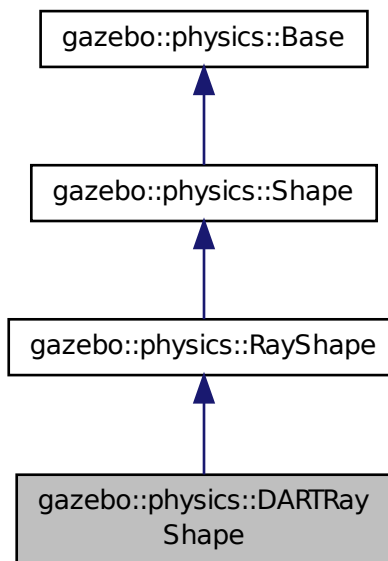
- **DARTPlaneShape.hh**

10.59 gazebo::physics::DARTRayShape Class Reference

Ray collision.

```
#include <DARTRayShape.hh>
```

Inheritance diagram for gazebo::physics::DARTRayShape:



Public Member Functions

- **DARTRayShape (PhysicsEnginePtr _physicsEngine)**
Constructor for a global ray.
- **DARTRayShape (CollisionPtr _collision)**
Constructor.
- virtual **~DARTRayShape ()**
Destructor.
- virtual void **GetIntersection** (double &_dist, std::string &_entity)
Get the nearest intersection.
- virtual void **SetPoints** (const **math::Vector3** &_posStart, const **math::Vector3** &_posEnd)
Set the ray based on starting and ending points relative to the body.
- virtual void **Update ()**
Update the ray collision.

Additional Inherited Members

10.59.1 Detailed Description

Ray collision.

The documentation for this class was generated from the following file:

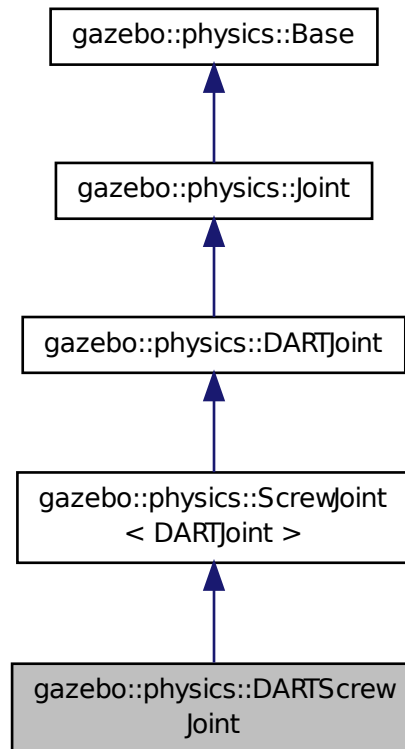
- **DARTRayShape.hh**

10.60 gazebo::physics::DARTScrewJoint Class Reference

A screw joint.

```
#include <DARTScrewJoint.hh>
```

Inheritance diagram for gazebo::physics::DARTScrewJoint:



Public Member Functions

- **DARTScrewJoint** (**BasePtr** _parent)

Constructor.

- virtual `~DARTScrewJoint ()`

Destructor.

- virtual `math::Vector3 GetAnchor (unsigned int _index) const`
Get the anchor point.
- virtual `math::Angle GetAngleImpl (unsigned int _index) const`
Get the angle of an axis helper function.
- virtual `math::Vector3 GetGlobalAxis (unsigned int _index) const`
Get the axis of rotation in global coordinate frame.
- virtual `math::Angle GetHighStop (unsigned int _index)`
Get the high stop of an axis(index).
- virtual `math::Angle GetLowStop (unsigned int _index)`
Get the low stop of an axis(index).
- virtual double `GetMaxForce (unsigned int _index)`
Get the max allowed force of an axis(index).
- virtual double `GetThreadPitch (unsigned int _index)`
Get screw joint thread pitch.
- virtual double `GetThreadPitch ()`
Get screw joint thread pitch.
- virtual double `GetVelocity (unsigned int _index) const`
Get the rotation rate of an axis(index)
- virtual void `Init ()`
Initialize joint.
- virtual void `Load (sdf::ElementPtr _sdf)`
*Load a **ScrewJoint** (p. 896).*
- virtual void `SetAnchor (unsigned int _index, const math::Vector3 &_anchor)`
Set the anchor point.
- virtual void `SetAxis (unsigned int _index, const math::Vector3 &_axis)`
Set the axis of rotation where axis is specified in local joint frame.
- virtual void `SetMaxForce (unsigned int _index, double _force)`
Set the max allowed force of an axis(index).
- virtual void `SetThreadPitch (unsigned int _index, double _threadPitch)`
Set screw joint thread pitch.
- virtual void `SetThreadPitch (double _threadPitch)`
Set screw joint thread pitch.
- virtual void `SetVelocity (unsigned int _index, double _vel)`
Set the velocity of an axis(index).

Protected Member Functions

- virtual void `SetForceImpl (unsigned int _index, double _effort)`
*Set the force applied to this **physics::Joint** (p. 541).*

Protected Attributes

- `dart::dynamics::ScrewJoint * dartScrewJoint`
Universal joint of DART.

Additional Inherited Members

10.60.1 Detailed Description

A screw joint.

10.60.2 Constructor & Destructor Documentation

10.60.2.1 gazebo::physics::DARTScrewJoint::DARTScrewJoint (BasePtr *_parent*)

Constructor.

Parameters

in	<i>_parent</i>	Pointer to the Link (p. 595) that is the joint' parent
----	----------------	---

10.60.2.2 virtual gazebo::physics::DARTScrewJoint::~~DARTScrewJoint () [virtual]

Destructor.

10.60.3 Member Function Documentation

10.60.3.1 virtual math::Vector3 gazebo::physics::DARTScrewJoint::GetAnchor (unsigned int *_index*) const [virtual]

Get the anchor point.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 549).

10.60.3.2 virtual math::Angle gazebo::physics::DARTScrewJoint::GetAngleImpl (unsigned int *_index*) const [virtual]

Get the angle of an axis helper function.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 550).

10.60.3.3 `virtual math::Vector3 gazebo::physics::DARTScrewJoint::GetGlobalAxis (unsigned int _index) const` [virtual]

Get the axis of rotation in global coordinate frame.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis to get.
-----------------	----------------------------	---------------------------

Returns

Axis value for the provided index.

Implements `gazebo::physics::Joint` (p. 553).

10.60.3.4 `virtual math::Angle gazebo::physics::DARTScrewJoint::GetHighStop (unsigned int _index)` [virtual]

Get the high stop of an axis(index).

This function is replaced by `GetUpperLimit(unsigned int)`. If you are interested in getting the value of `dParamHiStop*`, use `GetAttribute(hi_stop, _index)`

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Angle of the high stop value.

Reimplemented from `gazebo::physics::DARTJoint` (p. 332).

10.60.3.5 `virtual math::Angle gazebo::physics::DARTScrewJoint::GetLowStop (unsigned int _index)` [virtual]

Get the low stop of an axis(index).

This function is replaced by `GetLowerLimit(unsigned int)`. If you are interested in getting the value of `dParamHiStop*`, use `GetAttribute(hi_stop, _index)`

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Angle of the low stop value.

Reimplemented from `gazebo::physics::DARTJoint` (p. 334).

10.60.3.6 `virtual double gazebo::physics::DARTScrewJoint::GetMaxForce (unsigned int _index)` [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 556).

10.60.3.7 `virtual double gazebo::physics::DARTScrewJoint::GetThreadPitch (unsigned int _index) [virtual]`

Get screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

_threadPitch Thread pitch value.

Implements **gazebo::physics::ScrewJoint**< **DARTJoint** > (p. 898).

10.60.3.8 `virtual double gazebo::physics::DARTScrewJoint::GetThreadPitch () [virtual]`

Get screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

_threadPitch Thread pitch value.

Implements **gazebo::physics::ScrewJoint**< **DARTJoint** > (p. 898).

10.60.3.9 `virtual double gazebo::physics::DARTScrewJoint::GetVelocity (unsigned int _index) const [virtual]`

Get the rotation rate of an axis(index)

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 558).

10.60.3.10 virtual void gazebo::physics::DARTScrewJoint::Init () [virtual]

Initialize joint.

Reimplemented from **gazebo::physics::ScrewJoint< DARTJoint >** (p. 898).

10.60.3.11 virtual void gazebo::physics::DARTScrewJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load a **ScrewJoint** (p. 896).

Parameters

in	<i>_sdf</i>	SDF value to load from
----	-------------	------------------------

Reimplemented from **gazebo::physics::ScrewJoint< DARTJoint >** (p. 898).

10.60.3.12 virtual void gazebo::physics::DARTScrewJoint::SetAnchor (unsigned int *int*, const math::Vector3 &) [virtual]

Set the anchor point.

Reimplemented from **gazebo::physics::DARTJoint** (p. 335).

10.60.3.13 virtual void gazebo::physics::DARTScrewJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	<i>_index</i>	Index of the axis to set.
in	<i>_axis</i>	Vector in local joint frame of axis direction (must have length greater than zero).

Implements **gazebo::physics::Joint** (p. 561).

10.60.3.14 virtual void gazebo::physics::DARTScrewJoint::SetForceImpl (unsigned int *_index*, double *_force*) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 541)).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_force</i>	Force value.

Implements **gazebo::physics::DARTJoint** (p. 336).

10.60.3.15 `virtual void gazebo::physics::DARTScrewJoint::SetMaxForce (unsigned int _index, double _force) [virtual]`

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_force</i>	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 563).

10.60.3.16 `virtual void gazebo::physics::DARTScrewJoint::SetThreadPitch (unsigned int _index, double _threadPitch) [virtual]`

Set screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class Deprecated, please use the index-less version in the future: `virtual void SetThreadPitch(double _threadPitch)` (p. 370) = 0;

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_threadPitch</i>	Thread pitch value.

Implements **gazebo::physics::ScrewJoint< DARTJoint >** (p. 899).

10.60.3.17 `virtual void gazebo::physics::DARTScrewJoint::SetThreadPitch (double _threadPitch) [virtual]`

Set screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class Deprecated, please use the index-less version in the future: `virtual void SetThreadPitch(double _threadPitch)` (p. 899) = 0;

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_threadPitch</i>	Thread pitch value.

Implements **gazebo::physics::ScrewJoint< DARTJoint >** (p. 899).

10.60.3.18 `virtual void gazebo::physics::DARTScrewJoint::SetVelocity (unsigned int _index, double _vel) [virtual]`

Set the velocity of an axis(index).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_vel</i>	Velocity.

Implements `gazebo::physics::Joint` (p. 565).

10.60.4 Member Data Documentation

10.60.4.1 `dart::dynamics::ScrewJoint*` `gazebo::physics::DARTScrewJoint::dartScrewJoint` [protected]

Universal joint of DART.

The documentation for this class was generated from the following file:

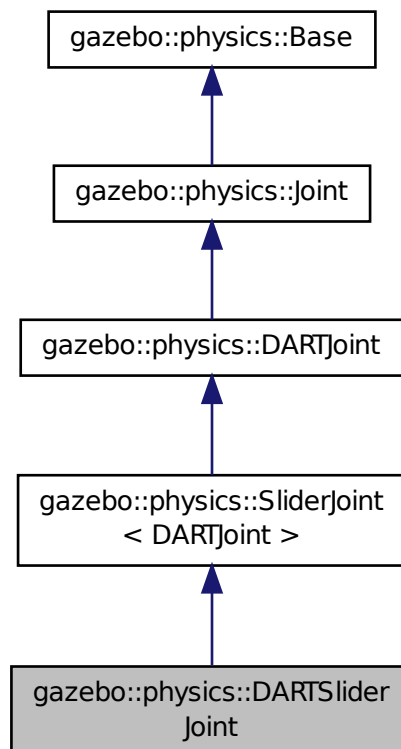
- `DARTScrewJoint.hh`

10.61 gazebo::physics::DARTSliderJoint Class Reference

A slider joint.

```
#include <DARTSliderJoint.hh>
```

Inheritance diagram for `gazebo::physics::DARTSliderJoint`:



Public Member Functions

- **DARTSliderJoint** (**BasePtr** _parent)
Constructor.
- virtual \sim **DARTSliderJoint** ()
Destructor.
- virtual **math::Vector3** **GetAnchor** (unsigned int _index) const
Get the anchor point.
- virtual **math::Angle** **GetAngleImpl** (unsigned int _index) const
Get the angle of an axis helper function.
- virtual **math::Vector3** **GetGlobalAxis** (unsigned int _index) const
Get the axis of rotation in global coordinate frame.
- virtual double **GetMaxForce** (unsigned int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (unsigned int _index) const
Get the rotation rate of an axis(index)
- virtual void **Init** ()
Initialize a joint.
- virtual void **Load** (sdf::ElementPtr _sdf)
*Load a **SliderJoint** (p. 1044).*
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)
Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int _index, double _force)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (unsigned int _index, double _vel)
Set the velocity of an axis(index).

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _effort)
*Set the force applied to this **physics::Joint** (p. 541).*

Protected Attributes

- dart::dynamics::PrismaticJoint * **dtPrismaticJoint**
Prismatic joint of DART.

Additional Inherited Members

10.61.1 Detailed Description

A slider joint.

10.61.2 Constructor & Destructor Documentation

10.61.2.1 gazebo::physics::DARTSliderJoint::DARTSliderJoint (BasePtr *_parent*)

Constructor.

Parameters

in	<i>_parent</i>	Pointer to the Link (p. 595) that is the joint' parent
----	----------------	---

10.61.2.2 virtual gazebo::physics::DARTSliderJoint::~~DARTSliderJoint () [virtual]

Destructor.

10.61.3 Member Function Documentation

10.61.3.1 virtual math::Vector3 gazebo::physics::DARTSliderJoint::GetAnchor (unsigned int *_index*) const [virtual]

Get the anchor point.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 549).

10.61.3.2 virtual math::Angle gazebo::physics::DARTSliderJoint::GetAngleImpl (unsigned int *_index*) const [virtual]

Get the angle of an axis helper function.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 550).

10.61.3.3 virtual math::Vector3 gazebo::physics::DARTSliderJoint::GetGlobalAxis (unsigned int *_index*) const [virtual]

Get the axis of rotation in global coordinate frame.

Parameters

in	<i>_index</i>	Index of the axis to get.
----	---------------	---------------------------

Returns

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 553).

10.61.3.4 `virtual double gazebo::physics::DARTSliderJoint::GetMaxForce (unsigned int _index) [virtual]`

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 556).

10.61.3.5 `virtual double gazebo::physics::DARTSliderJoint::GetVelocity (unsigned int _index) const [virtual]`

Get the rotation rate of an axis(index)

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 558).

10.61.3.6 `virtual void gazebo::physics::DARTSliderJoint::Init () [virtual]`

Initialize a joint.

Reimplemented from **gazebo::physics::DARTJoint** (p. 334).

10.61.3.7 `virtual void gazebo::physics::DARTSliderJoint::Load (sdf::ElementPtr _sdf) [virtual]`

Load a **SliderJoint** (p. 1044).

Parameters

<code>in</code>	<code><i>_sdf</i></code>	SDF values to load from
-----------------	--------------------------	-------------------------

Reimplemented from **gazebo::physics::SliderJoint< DARTJoint >** (p. 1045).

10.61.3.8 virtual void gazebo::physics::DARTSliderJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*)
[virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	<i>_index</i>	Index of the axis to set.
in	<i>_axis</i>	Vector in local joint frame of axis direction (must have length greater than zero).

Implements **gazebo::physics::Joint** (p. 561).

10.61.3.9 virtual void gazebo::physics::DARTSliderJoint::SetForceImpl (unsigned int *_index*, double *_force*) [protected],
[virtual]

Set the force applied to this **physics::Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 541)).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_force</i>	Force value.

Implements **gazebo::physics::DARTJoint** (p. 336).

10.61.3.10 virtual void gazebo::physics::DARTSliderJoint::SetMaxForce (unsigned int *_index*, double *_force*) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_force</i>	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 563).

10.61.3.11 virtual void gazebo::physics::DARTSliderJoint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_vel</i>	Velocity.

Implements **gazebo::physics::Joint** (p. 565).

10.61.4 Member Data Documentation

10.61.4.1 `dart::dynamics::PrismaticJoint*` `gazebo::physics::DARTSliderJoint::dtPrismaticJoint` [protected]

Prismatic joint of DART.

The documentation for this class was generated from the following file:

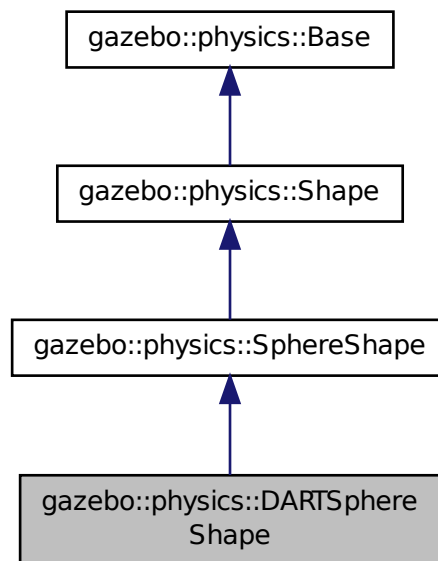
- **DARTSliderJoint.hh**

10.62 gazebo::physics::DARTSphereShape Class Reference

A DART sphere shape.

```
#include <DARTSphereShape.hh>
```

Inheritance diagram for gazebo::physics::DARTSphereShape:



Public Member Functions

- **DARTSphereShape** (`DARTCollisionPtr` _parent)
Constructor.
- virtual `~DARTSphereShape` ()
Destructor.
- virtual void **SetRadius** (double _radius)
Set the size.

Additional Inherited Members

10.62.1 Detailed Description

A DART sphere shape.

10.62.2 Constructor & Destructor Documentation

10.62.2.1 `gazebo::physics::DARTSphereShape::DARTSphereShape (DARTCollisionPtr _parent) [inline], [explicit]`

Constructor.

Parameters

<code>in</code>	<code><i>_parent</i></code>	Parent Collision (p. 235).
-----------------	-----------------------------	-----------------------------------

10.62.2.2 `virtual gazebo::physics::DARTSphereShape::~DARTSphereShape () [inline], [virtual]`

Destructor.

10.62.3 Member Function Documentation

10.62.3.1 `virtual void gazebo::physics::DARTSphereShape::SetRadius (double _radius) [inline], [virtual]`

Set the size.

Parameters

<code>in</code>	<code><i>_radius</i></code>	Radius of the sphere.
-----------------	-----------------------------	-----------------------

Reimplemented from `gazebo::physics::SphereShape` (p. 1057).

References `gazebo::math::equal()`, `gazebo::physics::DARTCollision::GetDARTBodyNode()`, `gzerr`, `gzwarn`, `NULL`, and `gazebo::physics::SphereShape::SetRadius()`.

The documentation for this class was generated from the following file:

- **DARTSphereShape.hh**

10.63 gazebo::physics::DARTTypes Class Reference

A set of functions for converting between the math types used by gazebo and dart.

```
#include <DARTTypes.hh>
```

Static Public Member Functions

- static Eigen::Isometry3d **ConvPose** (const **math::Pose** &*_pose*)
- static **math::Pose ConvPose** (const Eigen::Isometry3d &*_T*)

- static Eigen::Quaterniond **ConvQuat** (const **math::Quaternion** &_quat)
- static **math::Quaternion ConvQuat** (const Eigen::Quaterniond &_quat)
- static Eigen::Vector3d **ConvVec3** (const **math::Vector3** &_vec3)
- static **math::Vector3 ConvVec3** (const Eigen::Vector3d &_vec3)

10.63.1 Detailed Description

A set of functions for converting between the math types used by gazebo and dart.

The documentation for this class was generated from the following file:

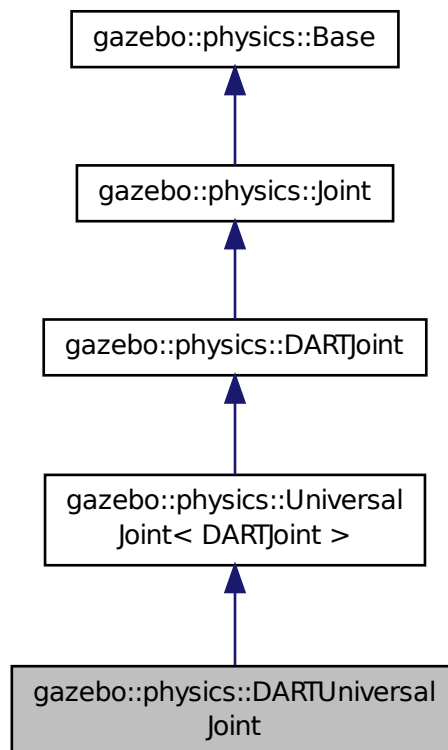
- **DARTTypes.hh**

10.64 gazebo::physics::DARTUniversalJoint Class Reference

A universal joint.

```
#include <DARTUniversalJoint.hh>
```

Inheritance diagram for gazebo::physics::DARTUniversalJoint:



Public Member Functions

- **DARTUniversalJoint** (**BasePtr** _parent)
Constructor.
- virtual **~DARTUniversalJoint** ()
Destructor.
- virtual **math::Vector3** **GetAnchor** (unsigned int _index) const
Get the anchor point.
- virtual **math::Angle** **GetAngleImpl** (unsigned int _index) const
Get the angle of an axis helper function.
- virtual **math::Vector3** **GetGlobalAxis** (unsigned int _index) const
Get the axis of rotation in global coordinate frame.
- virtual double **GetMaxForce** (unsigned int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (unsigned int _index) const
Get the rotation rate of an axis(index)
- virtual void **Init** ()
Initialize joint.
- virtual void **Load** (sdf::ElementPtr _sdf)
*Load a **UniversalJoint** (p. 1135).*
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)
Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int _index, double _force)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (unsigned int _index, double _vel)
Set the velocity of an axis(index).

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _effort)
*Set the force applied to this **physics::Joint** (p. 541).*

Protected Attributes

- dart::dynamics::UniversalJoint * **dtUniversalJoint**
Universal joint of DART.

Additional Inherited Members

10.64.1 Detailed Description

A universal joint.

10.64.2 Constructor & Destructor Documentation

10.64.2.1 gazebo::physics::DARTUniversalJoint::DARTUniversalJoint (BasePtr _parent)

Constructor.

Parameters

in	_parent	Pointer to the Link (p. 595) that is the joint' parent
----	---------	---

10.64.2.2 virtual gazebo::physics::DARTUniversalJoint::~~DARTUniversalJoint () [virtual]

Destuctor.

10.64.3 Member Function Documentation

10.64.3.1 virtual math::Vector3 gazebo::physics::DARTUniversalJoint::GetAnchor (unsigned int _index) const [virtual]

Get the anchor point.

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 549).

10.64.3.2 virtual math::Angle gazebo::physics::DARTUniversalJoint::GetAngleImpl (unsigned int _index) const [virtual]

Get the angle of an axis helper function.

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 550).

10.64.3.3 virtual math::Vector3 gazebo::physics::DARTUniversalJoint::GetGlobalAxis (unsigned int _index) const [virtual]

Get the axis of rotation in global coordinate frame.

Parameters

in	<i>_index</i>	Index of the axis to get.
----	---------------	---------------------------

Returns

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 553).

10.64.3.4 virtual double gazebo::physics::DARTUniversalJoint::GetMaxForce (unsigned int *_index*) [virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 556).

10.64.3.5 virtual double gazebo::physics::DARTUniversalJoint::GetVelocity (unsigned int *_index*) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

The rotational velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 558).

10.64.3.6 virtual void gazebo::physics::DARTUniversalJoint::Init () [virtual]

Initialize joint.

Reimplemented from **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1136).

10.64.3.7 virtual void gazebo::physics::DARTUniversalJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load a **UniversalJoint** (p. 1135).

Parameters

in	<i>_sdf</i>	SDF values to load from.
----	-------------	--------------------------

Reimplemented from `gazebo::physics::UniversalJoint< DARTJoint >` (p. 1136).

10.64.3.8 `virtual void gazebo::physics::DARTUniversalJoint::SetAxis (unsigned int _index, const math::Vector3 & _axis)`
`[virtual]`

Set the axis of rotation where axis is specified in local joint frame.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to set.
<code>in</code>	<code>_axis</code>	Vector in local joint frame of axis direction (must have length greater than zero).

Implements `gazebo::physics::Joint` (p. 561).

10.64.3.9 `virtual void gazebo::physics::DARTUniversalJoint::SetForceImpl (unsigned int _index, double _force)`
`[protected], [virtual]`

Set the force applied to this `physics::Joint` (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to `SetForceImpl` to the same joint in the same time step will accumulate forces on that `Joint` (p. 541)).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_force</code>	Force value.

Implements `gazebo::physics::DARTJoint` (p. 336).

10.64.3.10 `virtual void gazebo::physics::DARTUniversalJoint::SetMaxForce (unsigned int _index, double _force)` `[virtual]`

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_force</code>	Maximum force that can be applied to the axis.

Implements `gazebo::physics::Joint` (p. 563).

10.64.3.11 `virtual void gazebo::physics::DARTUniversalJoint::SetVelocity (unsigned int _index, double _vel)` `[virtual]`

Set the velocity of an axis(index).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_vel</code>	Velocity.

Implements `gazebo::physics::Joint` (p. 565).

10.64.4 Member Data Documentation

10.64.4.1 `dart::dynamics::UniversalJoint*` `gazebo::physics::DARTUniversalJoint::dtUniversalJoint` [protected]

Universal joint of DART.

The documentation for this class was generated from the following file:

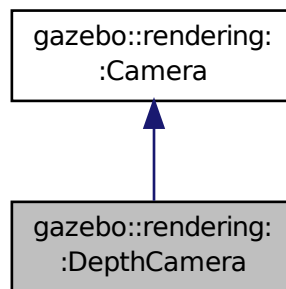
- **DARTUniversalJoint.hh**

10.65 gazebo::rendering::DepthCamera Class Reference

Depth camera used to render depth data into an image buffer.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::DepthCamera:



Public Member Functions

- **DepthCamera** (const std::string &_namePrefix, **ScenePtr** _scene, bool _autoRender=true)
Constructor.
- virtual **~DepthCamera** ()
Destructor.
- template<typename T >
event::ConnectionPtr ConnectNewDepthFrame (T _subscriber)
Connect a to the new depth image signal.
- template<typename T >
event::ConnectionPtr ConnectNewRGBPointCloud (T _subscriber)
Connect a to the new rgb point cloud signal.
- void **CreateDepthTexture** (const std::string &_textureName)
Create a texture which will hold the depth data.
- void **DisconnectNewDepthFrame** (**event::ConnectionPtr** &_c)
Disconnect from an depth image singal.

- void **DisconnectNewRGBPointCloud** (event::ConnectionPtr &c)
Disconnect from an rgb point cloud singal.
- void **Fini** ()
Finalize the camera.
- virtual const float * **GetDepthData** ()
All things needed to get back z buffer for depth data.
- void **Init** ()
Initialize the camera.
- void **Load** (sdf::ElementPtr _sdf)
Load the camera with a set of parmeters.
- void **Load** ()
Load the camera with default parmeters.
- virtual void **PostRender** ()
Render the camera.
- virtual void **SetDepthTarget** (Ogre::RenderTarget * _target)
Set the render target, which renders the depth data.

Protected Attributes

- Ogre::RenderTarget * **depthTarget**
Pointer to the depth target.
- Ogre::Texture * **depthTexture**
Pointer to the depth texture.
- Ogre::Viewport * **depthViewport**
Pointer to the depth viewport.

Additional Inherited Members

10.65.1 Detailed Description

Depth camera used to render depth data into an image buffer.

10.65.2 Constructor & Destructor Documentation

10.65.2.1 gazebo::rendering::DepthCamera::DepthCamera (const std::string & *_namePrefix*, ScenePtr *_scene*, bool *_autoRender* = true)

Constructor.

Parameters

in	<i>_namePrefix</i>	Unique prefix name for the camera.
in	<i>_scene</i>	Scene (p. 879) that will contain the camera
in	<i>_autoRender</i>	Almost everyone should leave this as true.

10.65.2.2 `virtual gazebo::rendering::DepthCamera::~DepthCamera () [virtual]`

Destructor.

10.65.3 Member Function Documentation

10.65.3.1 `template<typename T> event::ConnectionPtr gazebo::rendering::DepthCamera::ConnectNewDepthFrame (T _subscriber) [inline]`

Connect a to the new depth image signal.

Parameters

<code>in</code>	<code>_subscriber</code>	Subscriber callback function
-----------------	--------------------------	------------------------------

Returns

Pointer to the new Connection. This must be kept in scope

10.65.3.2 `template<typename T> event::ConnectionPtr gazebo::rendering::DepthCamera::ConnectNewRGBPointCloud (T _subscriber) [inline]`

Connect a to the new rgb point cloud signal.

Parameters

<code>in</code>	<code>_subscriber</code>	Subscriber callback function
-----------------	--------------------------	------------------------------

Returns

Pointer to the new Connection. This must be kept in scope

10.65.3.3 `void gazebo::rendering::DepthCamera::CreateDepthTexture (const std::string & _textureName)`

Create a texture which will hold the depth data.

Parameters

<code>in</code>	<code>_textureName</code>	Name of the texture to create
-----------------	---------------------------	-------------------------------

10.65.3.4 `void gazebo::rendering::DepthCamera::DisconnectNewDepthFrame (event::ConnectionPtr & _c) [inline]`

Disconnect from an depth image signal.

Parameters

<code>in</code>	<code>_c</code>	The connection to disconnect
-----------------	-----------------	------------------------------

10.65.3.5 `void gazebo::rendering::DepthCamera::DisconnectNewRGBPointCloud (event::ConnectionPtr & c) [inline]`

Disconnect from an rgb point cloud singal.

Parameters

in	_c	The connection to disconnect
----	----	------------------------------

10.65.3.6 `void gazebo::rendering::DepthCamera::Fini () [virtual]`

Finalize the camera.

Reimplemented from `gazebo::rendering::Camera` (p.207).

10.65.3.7 `virtual const float* gazebo::rendering::DepthCamera::GetDepthData () [virtual]`

All things needed to get back z buffer for depth data.

Returns

The z-buffer as a float array

10.65.3.8 `void gazebo::rendering::DepthCamera::Init () [virtual]`

Initialize the camera.

Reimplemented from `gazebo::rendering::Camera` (p.214).

10.65.3.9 `void gazebo::rendering::DepthCamera::Load (sdf::ElementPtr _sdf) [virtual]`

Load the camera with a set of parmeters.

Parameters

in	_sdf	The SDF camera info
----	------	---------------------

Reimplemented from `gazebo::rendering::Camera` (p.215).

10.65.3.10 `void gazebo::rendering::DepthCamera::Load () [virtual]`

Load the camera with default parmeters.

Reimplemented from `gazebo::rendering::Camera` (p.215).

10.65.3.11 `virtual void gazebo::rendering::DepthCamera::PostRender () [virtual]`

Render the camera.

Reimplemented from `gazebo::rendering::Camera` (p.216).

10.65.3.12 virtual void gazebo::rendering::DepthCamera::SetDepthTarget (Ogre::RenderTarget * *_target*) [virtual]

Set the render target, which renders the depth data.

Parameters

in	<i>_target</i>	Pointer to the render target
----	----------------	------------------------------

10.65.4 Member Data Documentation

10.65.4.1 Ogre::RenderTarget* gazebo::rendering::DepthCamera::depthTarget [protected]

Pointer to the depth target.

10.65.4.2 Ogre::Texture* gazebo::rendering::DepthCamera::depthTexture [protected]

Pointer to the depth texture.

10.65.4.3 Ogre::Viewport* gazebo::rendering::DepthCamera::depthViewport [protected]

Pointer to the depth viewport.

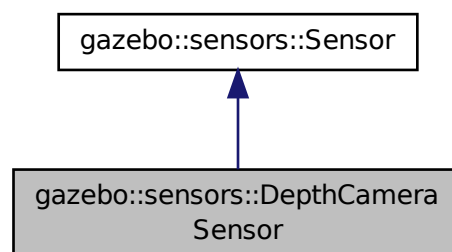
The documentation for this class was generated from the following file:

- **DepthCamera.hh**

10.66 gazebo::sensors::DepthCameraSensor Class Reference

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::DepthCameraSensor:



Public Member Functions

- **DepthCameraSensor** ()
Constructor.
- virtual **~DepthCameraSensor** ()
Destructor.
- **rendering::DepthCameraPtr GetDepthCamera** () const
*Returns a pointer to the **rendering::DepthCamera** (p. 383).*
- bool **SaveFrame** (const std::string &_filename)
Saves an image frame of depth camera sensor to file.
- virtual void **SetActive** (bool _value)
Set whether the sensor is active or not.

Protected Member Functions

- virtual void **Fini** ()
Finalize the camera.
- virtual void **Init** ()
Initialize the camera.
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)
Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.
- virtual bool **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.66.1 Constructor & Destructor Documentation

10.66.1.1 gazebo::sensors::DepthCameraSensor::DepthCameraSensor ()

Constructor.

10.66.1.2 virtual gazebo::sensors::DepthCameraSensor::~~DepthCameraSensor () [virtual]

Destructor.

10.66.2 Member Function Documentation

10.66.2.1 virtual void gazebo::sensors::DepthCameraSensor::Fini () [protected],[virtual]

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 912).

10.66.2.2 `rendering::DepthCameraPtr gazebo::sensors::DepthCameraSensor::GetDepthCamera () const [inline]`

Returns a pointer to the `rendering::DepthCamera` (p. 383).

Returns

Depth Camera pointer

10.66.2.3 `virtual void gazebo::sensors::DepthCameraSensor::Init () [protected],[virtual]`

Initialize the camera.

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.66.2.4 `virtual void gazebo::sensors::DepthCameraSensor::Load (const std::string & _worldName, sdf::ElementPtr _sdf) [protected],[virtual]`

Load the sensor with SDF parameters.

Parameters

<code>in</code>	<code>_sdf</code>	SDF Sensor (p. 907) parameters
<code>in</code>	<code>_worldName</code>	Name of world to load from

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.66.2.5 `virtual void gazebo::sensors::DepthCameraSensor::Load (const std::string & _worldName) [protected],[virtual]`

Load the sensor with default parameters.

Parameters

<code>in</code>	<code>_worldName</code>	Name of world to load from
-----------------	-------------------------	----------------------------

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.66.2.6 `bool gazebo::sensors::DepthCameraSensor::SaveFrame (const std::string & _filename)`

Saves an image frame of depth camera sensor to file.

Parameters

<code>in</code>	<code>Name</code>	of file to save as
-----------------	-------------------	--------------------

Returns

True if saved, false if not

10.66.2.7 `virtual void gazebo::sensors::DepthCameraSensor::SetActive (bool _value) [virtual]`

Set whether the sensor is active or not.

Parameters

<code>in</code>	<code><i>_value</i></code>	True if active, false if not
-----------------	----------------------------	------------------------------

Reimplemented from `gazebo::sensors::Sensor` (p. 916).

10.66.2.8 `virtual bool gazebo::sensors::DepthCameraSensor::UpdateImpl (bool) [protected],[virtual]`

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

<code>in</code>	<code><i>_force</i></code>	True if update is forced, false if not
-----------------	----------------------------	--

Returns

True if the sensor was updated.

Reimplemented from `gazebo::sensors::Sensor` (p. 917).

The documentation for this class was generated from the following file:

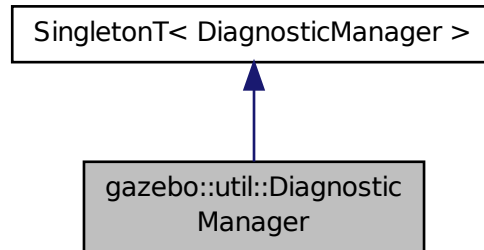
- `DepthCameraSensor.hh`

10.67 gazebo::util::DiagnosticManager Class Reference

A diagnostic manager class.

```
#include <util/util.hh>
```

Inheritance diagram for gazebo::util::DiagnosticManager:



Public Member Functions

- `std::string GetLabel (int _index) const`
Get a label for a timer.
- `boost::filesystem::path GetLogPath () const`
Get the path in which logs are stored.
- `common::Time GetTime (int _index) const`
Get the time of a timer instance.
- `common::Time GetTime (const std::string &_label) const`
Get a time based on a label.
- `int GetTimerCount () const`
Get the number of timers.
- `void Init (const std::string &_worldName)`
Initialize to report diagnostics about a world.
- `void Lap (const std::string &_name, const std::string &_prefix)`
Output the current elapsed time of an active timer with a prefix string.
- `void StartTimer (const std::string &_name)`
Start a new timer instance.
- `void StopTimer (const std::string &_name)`
Stop a currently running timer.

Additional Inherited Members

10.67.1 Detailed Description

A diagnostic manager class.

10.67.2 Member Function Documentation

10.67.2.1 `std::string gazebo::util::DiagnosticManager::GetLabel (int _index) const`

Get a label for a timer.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of a timer instance
-----------------	----------------------------	---------------------------

Returns

Label of the specified timer

10.67.2.2 `boost::filesystem::path gazebo::util::DiagnosticManager::GetLogPath () const`

Get the path in which logs are stored.

Returns

The path in which logs are stored.

10.67.2.3 `common::Time gazebo::util::DiagnosticManager::GetTime (int _index) const`

Get the time of a timer instance.

Parameters

<code>in</code>	<code><i>_index</i></code>	The index of a timer instance
-----------------	----------------------------	-------------------------------

Returns

Time of the specified timer

10.67.2.4 `common::Time gazebo::util::DiagnosticManager::GetTime (const std::string & _label) const`

Get a time based on a label.

Parameters

<code>in</code>	<code><i>_label</i></code>	Name of the timer instance
-----------------	----------------------------	----------------------------

Returns

Time of the specified timer

10.67.2.5 `int gazebo::util::DiagnosticManager::GetTimerCount () const`

Get the number of timers.

Returns

The number of timers

10.67.2.6 void gazebo::util::DiagnosticManager::Init (const std::string & *_worldName*)

Initialize to report diagnostics about a world.

Parameters

<i>in</i>	<i>_worldName</i>	Name of the world.
-----------	-------------------	--------------------

10.67.2.7 void gazebo::util::DiagnosticManager::Lap (const std::string & *_name*, const std::string & *_prefix*)

Output the current elapsed time of an active timer with a prefix string.

This also resets the timer and keeps it running.

Parameters

<i>in</i>	<i>_name</i>	Name of the timer to access.
<i>in</i>	<i>_prefix</i>	Informational string that is output with the elapsed time.

10.67.2.8 void gazebo::util::DiagnosticManager::StartTimer (const std::string & *_name*)

Start a new timer instance.

Parameters

<i>in</i>	<i>_name</i>	Name of the timer.
-----------	--------------	--------------------

Returns

A pointer to the new diagnostic timer

10.67.2.9 void gazebo::util::DiagnosticManager::StopTimer (const std::string & *_name*)

Stop a currently running timer.

Parameters

<i>in</i>	<i>_name</i>	Name of the timer to stop.
-----------	--------------	----------------------------

The documentation for this class was generated from the following file:

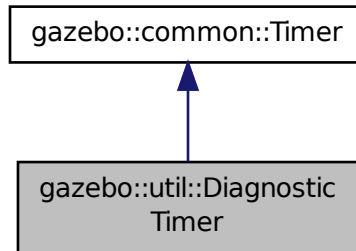
- **Diagnostics.hh**

10.68 gazebo::util::DiagnosticTimer Class Reference

A timer designed for diagnostics.

```
#include <util/util.hh>
```

Inheritance diagram for gazebo::util::DiagnosticTimer:



Public Member Functions

- **DiagnosticTimer** (const std::string &_name)
Constructor.
- virtual ~**DiagnosticTimer** ()
Destructor.
- const std::string **GetName** () const
Get the name of the timer.
- void **Lap** (const std::string &_prefix)
Output a lap time.
- virtual void **Start** ()
Start the timer.
- virtual void **Stop** ()
Stop the timer.

10.68.1 Detailed Description

A timer designed for diagnostics.

10.68.2 Constructor & Destructor Documentation

10.68.2.1 gazebo::util::DiagnosticTimer::DiagnosticTimer (const std::string & _name)

Constructor.

Parameters

in	<i>_name</i>	Name of the timer
----	--------------	-------------------

10.68.2.2 virtual gazebo::util::DiagnosticTimer::~~DiagnosticTimer () [virtual]

Destructor.

10.68.3 Member Function Documentation

10.68.3.1 const std::string gazebo::util::DiagnosticTimer::GetName () const [inline]

Get the name of the timer.

Returns

The name of timer

10.68.3.2 void gazebo::util::DiagnosticTimer::Lap (const std::string & *_prefix*)

Output a lap time.

Parameters

in	<i>_prefix</i>	Annotation to output with the elapsed time.
----	----------------	---

10.68.3.3 virtual void gazebo::util::DiagnosticTimer::Start () [virtual]

Start the timer.

Reimplemented from **gazebo::common::Timer** (p. 1122).

10.68.3.4 virtual void gazebo::util::DiagnosticTimer::Stop () [virtual]

Stop the timer.

Reimplemented from **gazebo::common::Timer** (p. 1122).

The documentation for this class was generated from the following file:

- **Diagnostics.hh**

10.69 gazebo::rendering::DummyPageProvider Class Reference

Pretends to provide procedural page content to avoid page loading.

```
#include <rendering/rendering.hh>
```

Public Member Functions

- bool **loadProceduralPage** (Ogre::Page *, Ogre::PagedWorldSection *)
Give a provider the opportunity to load page content procedurally.
- bool **prepareProceduralPage** (Ogre::Page *, Ogre::PagedWorldSection *)
Give a provider the opportunity to prepare page content procedurally.
- bool **unloadProceduralPage** (Ogre::Page *, Ogre::PagedWorldSection *)
Give a provider the opportunity to unload page content procedurally.
- bool **unprepareProceduralPage** (Ogre::Page *, Ogre::PagedWorldSection *)
Give a provider the opportunity to unprepare page content procedurally.

10.69.1 Detailed Description

Pretends to provide procedural page content to avoid page loading.

10.69.2 Member Function Documentation

10.69.2.1 `bool gazebo::rendering::DummyPageProvider::loadProceduralPage (Ogre::Page *, Ogre::PagedWorldSection *)`
[inline]

Give a provider the opportunity to load page content procedurally.

The parameters are not used.

10.69.2.2 `bool gazebo::rendering::DummyPageProvider::prepareProceduralPage (Ogre::Page *, Ogre::PagedWorldSection *)`
[inline]

Give a provider the opportunity to prepare page content procedurally.

The parameters are not used.

10.69.2.3 `bool gazebo::rendering::DummyPageProvider::unloadProceduralPage (Ogre::Page *, Ogre::PagedWorldSection *)`
[inline]

Give a provider the opportunity to unload page content procedurally.

The parameters are not used.

10.69.2.4 `bool gazebo::rendering::DummyPageProvider::unprepareProceduralPage (Ogre::Page *, Ogre::PagedWorldSection *)`
[inline]

Give a provider the opportunity to unprepare page content procedurally.

The parameters are not used.

The documentation for this class was generated from the following file:

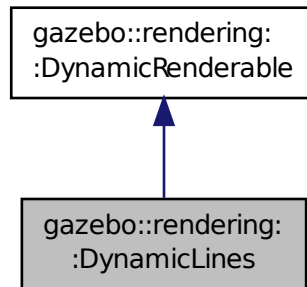
- **Heightmap.hh**

10.70 gazebo::rendering::DynamicLines Class Reference

Class for drawing lines that can change.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::DynamicLines:



Public Member Functions

- **DynamicLines** (**RenderOpType** _opType=**RENDERING_LINE_STRIP**)
Constructor.
- virtual **~DynamicLines** ()
Destructor.
- void **AddPoint** (const **math::Vector3** &_pt, const **common::Color** &_color=**common::Color::White**)
Add a point to the point list.
- void **AddPoint** (double _x, double _y, double _z, const **common::Color** &_color=**common::Color::White**)
Add a point to the point list.
- void **Clear** ()
Remove all points from the point list.
- virtual const **Ogre::String** & **getMovableType** () const
*Overridden function from **Ogre** (p. 137)'s base class.*
- const **math::Vector3** & **GetPoint** (unsigned int _index) const
Return the location of an existing point in the point list.
- unsigned int **GetPointCount** () const
Return the total number of points in the point list.
- void **SetColor** (unsigned int _index, const **common::Color** &_color)
Change the color of an existing point in the point list.
- void **SetPoint** (unsigned int _index, const **math::Vector3** &_value)
Change the location of an existing point in the point list.
- void **Update** ()
Call this to update the hardware buffer after making changes.

Static Public Member Functions

- static std::string **GetMovableType** ()
Get type of movable.

Additional Inherited Members

10.70.1 Detailed Description

Class for drawing lines that can change.

10.70.2 Constructor & Destructor Documentation

10.70.2.1 gazebo::rendering::DynamicLines::DynamicLines (RenderOpType *_opType* = RENDERING_LINE_STRIP)

Constructor.

Parameters

in	<i>_opType</i>	The type of Line
----	----------------	------------------

10.70.2.2 virtual gazebo::rendering::DynamicLines::~DynamicLines () [virtual]

Destructor.

10.70.3 Member Function Documentation

10.70.3.1 void gazebo::rendering::DynamicLines::AddPoint (const math::Vector3 & *_pt*, const common::Color & *_color* = common::Color::White)

Add a point to the point list.

Parameters

in	<i>_pt</i>	math::Vector3 (p. 1165) point
in	<i>_color</i>	common::Color (p. 249) Point color

10.70.3.2 void gazebo::rendering::DynamicLines::AddPoint (double *_x*, double *_y*, double *_z*, const common::Color & *_color* = common::Color::White)

Add a point to the point list.

Parameters

in	<i>_x</i>	X position
in	<i>_y</i>	Y position
in	<i>_z</i>	Z position
in	<i>_color</i>	common::Color (p. 249) Point color

10.70.3.3 void gazebo::rendering::DynamicLines::Clear ()

Remove all points from the point list.

10.70.3.4 static std::string gazebo::rendering::DynamicLines::GetMovableType () [static]

Get type of movable.

Returns

This returns "gazebo::dynamiclines"

10.70.3.5 virtual const Ogre::String& gazebo::rendering::DynamicLines::getMovableType () const [virtual]

Overridden function from **Ogre** (p. 137)'s base class.

Returns

Returns "gazebo::ogredynamicslines"

10.70.3.6 const math::Vector3& gazebo::rendering::DynamicLines::GetPoint (unsigned int *_index*) const

Return the location of an existing point in the point list.

Parameters

<i>in</i>	<i>_index</i>	Number of the point to return
-----------	---------------	-------------------------------

Returns

math::Vector3 (p. 1165) value of the point

10.70.3.7 unsigned int gazebo::rendering::DynamicLines::GetPointCount () const

Return the total number of points in the point list.

Returns

Number of points

10.70.3.8 void gazebo::rendering::DynamicLines::SetColor (unsigned int *_index*, const common::Color & *_color*)

Change the color of an existing point in the point list.

Parameters

<i>in</i>	<i>_index</i>	Index of the point to set
<i>in</i>	<i>_color</i>	common::Color (p. 249) Pixelcolor color to set the point to

10.70.3.9 void gazebo::rendering::DynamicLines::SetPoint (unsigned int *_index*, const math::Vector3 & *_value*)

Change the location of an existing point in the point list.

Parameters

in	<i>_index</i>	Index of the point to set
in	<i>_value</i>	math::Vector3 (p. 1165) value to set the point to

10.70.3.10 void gazebo::rendering::DynamicLines::Update ()

Call this to update the hardware buffer after making changes.

The documentation for this class was generated from the following file:

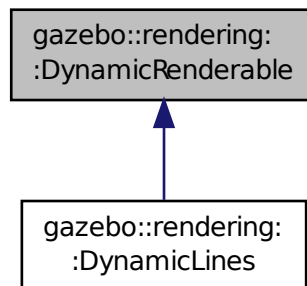
- **DynamicLines.hh**

10.71 gazebo::rendering::DynamicRenderable Class Reference

Abstract base class providing mechanisms for dynamically growing hardware buffers.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::DynamicRenderable:



Public Member Functions

- **DynamicRenderable** ()
Constructor.
- virtual **~DynamicRenderable** ()
Virtual destructor.
- virtual Ogre::Real **getBoundingRadius** () const
Implementation of Ogre::SimpleRenderable.

- `std::string GetMovableType () const`
Get type of movable.
- `RenderOpType GetOperationType () const`
Get the render operation type.
- `virtual Ogre::Real getSquaredViewDepth (const Ogre::Camera *_cam) const`
Implementation of Ogre::SimpleRenderable.
- `void Init (RenderOpType _opType, bool _useIndices=false)`
Initializes the dynamic renderable.
- `void SetOperationType (RenderOpType _opType)`
Set the render operation type.

Protected Member Functions

- `virtual void CreateVertexDeclaration ()=0`
Creates the vertex declaration.
- `virtual void FillHardwareBuffers ()=0`
Fills the hardware vertex and index buffers with data.
- `void PrepareHardwareBuffers (size_t _vertexCount, size_t _indexCount)`
Prepares the hardware buffers for the requested vertex and index counts.

Protected Attributes

- `size_t indexBufferCapacity`
Maximum capacity of the currently allocated index buffer.
- `size_t vertexBufferCapacity`
Maximum capacity of the currently allocated vertex buffer.

10.71.1 Detailed Description

Abstract base class providing mechanisms for dynamically growing hardware buffers.

10.71.2 Constructor & Destructor Documentation

10.71.2.1 gazebo::rendering::DynamicRenderable::DynamicRenderable ()

Constructor.

10.71.2.2 virtual gazebo::rendering::DynamicRenderable::~~DynamicRenderable () [virtual]

Virtual destructor.

10.71.3 Member Function Documentation

10.71.3.1 `virtual void gazebo::rendering::DynamicRenderable::CreateVertexDeclaration () [protected], [pure virtual]`

Creates the vertex declaration.

Remarks

Override and set `mRenderOp.vertexData->vertexDeclaration` here. `mRenderOp.vertexData` will be created for you before this method is called.

10.71.3.2 `virtual void gazebo::rendering::DynamicRenderable::FillHardwareBuffers () [protected], [pure virtual]`

Fills the hardware vertex and index buffers with data.

Remarks

This function must call `prepareHardwareBuffers()` before locking the buffers to ensure they are large enough for the data to be written. Afterwards the vertex and index buffers (if using indices) can be locked, and data can be written to them.

10.71.3.3 `virtual Ogre::Real gazebo::rendering::DynamicRenderable::getBoundingRadius () const [virtual]`

Implementation of `Ogre::SimpleRenderable`.

Returns

The bounding radius

10.71.3.4 `std::string gazebo::rendering::DynamicRenderable::GetMovableType () const`

Get type of movable.

Returns

This returns "gazebo::DynamicRenderable"

10.71.3.5 `RenderOpType gazebo::rendering::DynamicRenderable::GetOperationType () const`

Get the render operation type.

Returns

The render operation type.

10.71.3.6 `virtual Ogre::Real gazebo::rendering::DynamicRenderable::getSquaredViewDepth (const Ogre::Camera * _cam) const`
`[virtual]`

Implementation of `Ogre::SimpleRenderable`.

Parameters

<code>in</code>	<code>_cam</code>	Pointer to the Ogre (p. 137) camera that views the renderable.
-----------------	-------------------	---

Returns

The squared depth in the **Camera** (p. 197)'s view

10.71.3.7 `void gazebo::rendering::DynamicRenderable::Init (RenderOpType _opType, bool _useIndices = false)`

Initializes the dynamic renderable.

Remarks

This function should only be called once. It initializes the render operation, and calls the abstract function **CreateVertexDeclaration()** (p. 402).

Parameters

<code>in</code>	<code>_opType</code>	The type of render operation to perform.
<code>in</code>	<code>_useIndices</code>	Specifies whether to use indices to determine the vertices to use as input.

10.71.3.8 `void gazebo::rendering::DynamicRenderable::PrepareHardwareBuffers (size_t _vertexCount, size_t _indexCount)`
`[protected]`

Prepares the hardware buffers for the requested vertex and index counts.

Remarks

This function must be called before locking the buffers in `fillHardwareBuffers()`. It guarantees that the hardware buffers are large enough to hold at least the requested number of vertices and indices (if using indices). The buffers are possibly reallocated to achieve this.

The vertex and index count in the render operation are set to

the values of `vertexCount` and `indexCount` respectively.

Parameters

<code>in</code>	<code>_vertexCount</code>	The number of vertices the buffer must hold.
<code>in</code>	<code>_indexCount</code>	The number of indices the buffer must hold. This parameter is ignored if not using indices.

10.71.3.9 void gazebo::rendering::DynamicRenderable::SetOperationType (RenderOpType _opType)

Set the render operation type.

Parameters

in	<code>_opType</code>	The type of render operation to perform.
----	----------------------	--

10.71.4 Member Data Documentation

10.71.4.1 size_t gazebo::rendering::DynamicRenderable::indexBufferCapacity [protected]

Maximum capacity of the currently allocated index buffer.

10.71.4.2 size_t gazebo::rendering::DynamicRenderable::vertexBufferCapacity [protected]

Maximum capacity of the currently allocated vertex buffer.

The documentation for this class was generated from the following file:

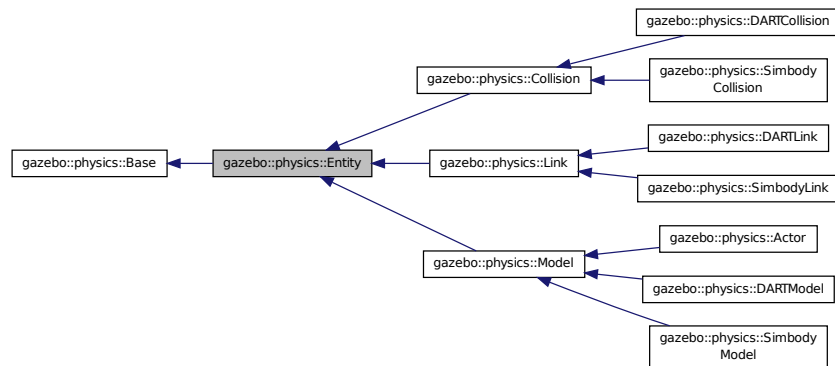
- **DynamicRenderable.hh**

10.72 gazebo::physics::Entity Class Reference

Base (p. 168) class for all physics objects in Gazebo.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::Entity:



Public Member Functions

- **Entity** (BasePtr _parent)
Constructor.
- virtual ~**Entity** ()

- Destructor.*

 - virtual void **Fini** ()

Finalize the entity.
- virtual **math::Box GetBoundingBox** () const

Return the bounding box for the entity.
- **CollisionPtr GetChildCollision** (const std::string &_name)

Get a child collision entity, if one exists.
- **LinkPtr GetChildLink** (const std::string &_name)

Get a child linke entity, if one exists.
- **math::Box GetCollisionBoundingBox** () const

Returns collision bounding box.
- const **math::Pose & GetDirtyPose** () const

*Returns **Entity::dirtyPose** (p. 415).*
- **math::Pose GetInitialRelativePose** () const

Get the initial relative pose.
- void **GetNearestEntityBelow** (double &_distBelow, std::string &_entityName)

Get the distance to the nearest entity below (along the Z-axis) this entity.
- **ModelPtr GetParentModel** ()

Get the parent model, if one exists.
- virtual **math::Vector3 GetRelativeAngularAccel** () const

Get the angular acceleration of the entity.
- virtual **math::Vector3 GetRelativeAngularVel** () const

Get the angular velocity of the entity.
- virtual **math::Vector3 GetRelativeLinearAccel** () const

Get the linear acceleration of the entity.
- virtual **math::Vector3 GetRelativeLinearVel** () const

Get the linear velocity of the entity.
- **math::Pose GetRelativePose** () const

Get the pose of the entity relative to its parent.
- virtual **math::Vector3 GetWorldAngularAccel** () const

Get the angular acceleration of the entity in the world frame.
- virtual **math::Vector3 GetWorldAngularVel** () const

Get the angular velocity of the entity in the world frame.
- virtual **math::Vector3 GetWorldLinearAccel** () const

Get the linear acceleration of the entity in the world frame.
- virtual **math::Vector3 GetWorldLinearVel** () const

Get the linear velocity of the entity in the world frame.
- const **math::Pose & GetWorldPose** () const

Get the absolute pose of the entity.
- bool **IsCanonicalLink** () const

A helper function that checks if this is a canonical body.
- bool **IsStatic** () const

Return whether this entity is static.
- virtual void **Load** (sdf::ElementPtr _sdf)

Load the entity.
- void **PlaceOnEntity** (const std::string &_entityName)

Move this entity to be ontop of another entity by name.

- void **PlaceOnNearestEntityBelow** ()
Move this entity to be ontop of the nearest entity below.
- virtual void **Reset** ()
Reset the entity.
- void **SetAnimation** (const **common::PoseAnimationPtr** &_anim, boost::function< void()> _onComplete)
Set an animation for this entity.
- void **SetAnimation** (**common::PoseAnimationPtr** _anim)
Set an animation for this entity.
- void **SetCanonicalLink** (bool _value)
Set to true if this entity is a canonical link for a model.
- void **SetInitialRelativePose** (const **math::Pose** &_pose)
Set the initial pose.
- virtual void **SetName** (const std::string &_name)
Set the name of the entity.
- void **SetRelativePose** (const **math::Pose** &_pose, bool _notify=true, bool _publish=true)
Set the pose of the entity relative to its parent.
- void **SetStatic** (const bool &_static)
Set whether this entity is static: immovable.
- void **SetWorldPose** (const **math::Pose** &_pose, bool _notify=true, bool _publish=true)
Set the world pose of the entity.
- void **SetWorldTwist** (const **math::Vector3** &_linear, const **math::Vector3** &_angular, bool _update-Children=true)
*Set angular and linear rates of an **physics::Entity** (p. 404).*
- virtual void **StopAnimation** ()
Stop the current animation, if any.
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)
Update the parameters using new sdf values.

Protected Member Functions

- virtual void **OnPoseChange** ()=0
This function is called when the entity's (or one of its parents) pose of the parent has changed.

Protected Attributes

- **common::PoseAnimationPtr** **animation**
Current pose animation.
- **event::ConnectionPtr** **animationConnection**
Connection used to update an animation.
- **math::Pose** **animationStartPose**
Start pose of an animation.
- std::vector< **event::ConnectionPtr** > **connections**
All our event connections.
- **math::Pose** **dirtyPose**
The pose set by a physics engine.
- **transport::NodePtr** **node**
Communication node.

- **EntityPtr parentEntity**
A helper that prevents numerous dynamic_casts.
- **common::Time prevAnimationTime**
Previous time an animation was updated.
- **transport::PublisherPtr requestPub**
Request publisher.
- **math::Vector3 scale**
Scale of the entity.
- **transport::PublisherPtr visPub**
Visual publisher.
- **msgs::Visual * visualMsg**
Visual message container.

Additional Inherited Members

10.72.1 Detailed Description

Base (p. 168) class for all physics objects in Gazebo.

10.72.2 Constructor & Destructor Documentation

10.72.2.1 `gazebo::physics::Entity::Entity (BasePtr _parent) [explicit]`

Constructor.

Parameters

in	_parent	Parent of the entity.
----	---------	-----------------------

10.72.2.2 `virtual gazebo::physics::Entity::~Entity () [virtual]`

Destructor.

10.72.3 Member Function Documentation

10.72.3.1 `virtual void gazebo::physics::Entity::Fini () [virtual]`

Finalize the entity.

Reimplemented from **gazebo::physics::Base** (p. 174).

Reimplemented in **gazebo::physics::Actor** (p. 142), **gazebo::physics::Link** (p. 604), **gazebo::physics::Model** (p. 683), **gazebo::physics::DARTModel** (p. 351), **gazebo::physics::Collision** (p. 239), **gazebo::physics::DART-Link** (p. 342), **gazebo::physics::SimbodyLink** (p. 976), and **gazebo::physics::DARTCollision** (p. 311).

10.72.3.2 `virtual math::Box gazebo::physics::Entity::GetBoundingBox () const [virtual]`

Return the bounding box for the entity.

Returns

The bounding box.

Reimplemented in **gazebo::physics::Link** (p. 604), **gazebo::physics::Model** (p. 683), **gazebo::physics::Collision** (p. 239), **gazebo::physics::DARTCollision** (p. 311), and **gazebo::physics::SimbodyCollision** (p. 945).

10.72.3.3 CollisionPtr gazebo::physics::Entity::GetChildCollision (const std::string & *_name*)

Get a child collision entity, if one exists.

Parameters

<i>in</i>	<i>_name</i>	Name of the child collision object.
-----------	--------------	-------------------------------------

Returns

Pointer to the **Collision** (p. 235) object, or NULL if not found.

10.72.3.4 LinkPtr gazebo::physics::Entity::GetChildLink (const std::string & *_name*)

Get a child linke entity, if one exists.

Parameters

<i>in</i>	<i>_name</i>	Name of the child Link (p. 595) object.
-----------	--------------	--

Returns

Pointer to the **Link** (p. 595) object, or NULL if not found.

10.72.3.5 math::Box gazebo::physics::Entity::GetCollisionBoundingBox () const

Returns collision bounding box.

Returns

Collsiion boundin box.

10.72.3.6 const math::Pose& gazebo::physics::Entity::GetDirtyPose () const

Returns **Entity::dirtyPose** (p. 415).

The dirty pose is the pose set by the physics engine before it's value is propagated to the rest of the simulator.

Returns

The dirty pose of the entity.

10.72.3.7 `math::Pose gazebo::physics::Entity::GetInitialRelativePose () const`

Get the initial relative pose.

Returns

The initial relative pose.

10.72.3.8 `void gazebo::physics::Entity::GetNearestEntityBelow (double & _distBelow, std::string & _entityName)`

Get the distance to the nearest entity below (along the Z-axis) this entity.

Parameters

out	<code>_distBelow</code>	The distance to the nearest entity below.
out	<code>_entityName</code>	The name of the nearest entity below.

10.72.3.9 `ModelPtr gazebo::physics::Entity::GetParentModel ()`

Get the parent model, if one exists.

Returns

Pointer to a model, or NULL if no parent model exists.

10.72.3.10 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularAccel () const [inline],[virtual]`

Get the angular acceleration of the entity.

Returns

A `math::Vector3` (p. 1165) for the acceleration.

Reimplemented in `gazebo::physics::Link` (p. 607), `gazebo::physics::Collision` (p. 240), and `gazebo::physics::Model` (p. 685).

10.72.3.11 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularVel () const [inline],[virtual]`

Get the angular velocity of the entity.

Returns

A `math::Vector3` (p. 1165) for the velocity.

Reimplemented in `gazebo::physics::Link` (p. 607), `gazebo::physics::Collision` (p. 240), and `gazebo::physics::Model` (p. 686).

10.72.3.12 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearAccel () const [inline],[virtual]`

Get the linear acceleration of the entity.

Returns

A **math::Vector3** (p. 1165) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 607), **gazebo::physics::Collision** (p. 240), and **gazebo::physics::Model** (p. 686).

10.72.3.13 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearVel() const [inline],[virtual]`

Get the linear velocity of the entity.

Returns

A **math::Vector3** (p. 1165) for the linear velocity.

Reimplemented in **gazebo::physics::Link** (p. 607), **gazebo::physics::Collision** (p. 241), and **gazebo::physics::Model** (p. 686).

10.72.3.14 `math::Pose gazebo::physics::Entity::GetRelativePose() const`

Get the pose of the entity relative to its parent.

Returns

The pose of the entity relative to its parent.

10.72.3.15 `virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularAccel() const [inline],[virtual]`

Get the angular acceleration of the entity in the world frame.

Returns

A **math::Vector3** (p. 1165) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 608), **gazebo::physics::Collision** (p. 242), and **gazebo::physics::Model** (p. 687).

10.72.3.16 `virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularVel() const [inline],[virtual]`

Get the angular velocity of the entity in the world frame.

Returns

A **math::Vector3** (p. 1165) for the velocity.

Reimplemented in **gazebo::physics::Collision** (p. 242), **gazebo::physics::Model** (p. 687), **gazebo::physics::DARTLink** (p. 343), and **gazebo::physics::SimbodyLink** (p. 976).

10.72.3.17 `virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearAccel () const [inline],[virtual]`

Get the linear acceleration of the entity in the world frame.

Returns

A `math::Vector3` (p. 1165) for the acceleration.

Reimplemented in `gazebo::physics::Link` (p. 610), `gazebo::physics::Collision` (p. 242), and `gazebo::physics::Model` (p. 688).

10.72.3.18 `virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearVel () const [inline],[virtual]`

Get the linear velocity of the entity in the world frame.

Returns

A `math::Vector3` (p. 1165) for the linear velocity.

Reimplemented in `gazebo::physics::Link` (p. 610), `gazebo::physics::Collision` (p. 242), and `gazebo::physics::Model` (p. 688).

10.72.3.19 `const math::Pose& gazebo::physics::Entity::GetWorldPose () const [inline]`

Get the absolute pose of the entity.

Returns

The absolute pose of the entity.

10.72.3.20 `bool gazebo::physics::Entity::IsCanonicalLink () const [inline]`

A helper function that checks if this is a canonical body.

Returns

True if the link is canonical.

10.72.3.21 `bool gazebo::physics::Entity::IsStatic () const`

Return whether this entity is static.

Returns

True if static.

10.72.3.22 `virtual void gazebo::physics::Entity::Load (sdf::ElementPtr _sdf) [virtual]`

Load the entity.

Parameters

in	_sdf	Pointer to an SDF element.
----	------	----------------------------

Reimplemented from **gazebo::physics::Base** (p. 177).

Reimplemented in **gazebo::physics::Actor** (p. 142), **gazebo::physics::Link** (p. 612), **gazebo::physics::Model** (p. 688), **gazebo::physics::Collision** (p. 243), **gazebo::physics::SimbodyCollision** (p. 945), **gazebo::physics::DARTLink** (p. 345), **gazebo::physics::SimbodyLink** (p. 978), **gazebo::physics::DARTModel** (p. 352), **gazebo::physics::DARTCollision** (p. 312), and **gazebo::physics::SimbodyModel** (p. 984).

10.72.3.23 virtual void gazebo::physics::Entity::OnPoseChange () [protected],[pure virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implemented in **gazebo::physics::Link** (p. 612), **gazebo::physics::Model** (p. 689), **gazebo::physics::DART-Link** (p. 345), **gazebo::physics::SimbodyLink** (p. 978), **gazebo::physics::DARTCollision** (p. 312), and **gazebo::physics::SimbodyCollision** (p. 945).

10.72.3.24 void gazebo::physics::Entity::PlaceOnEntity (const std::string & _entityName)

Move this entity to be ontop of another entity by name.

Parameters

in	_entityName	Name of the Entity (p. 404) this Entity (p. 404) should be ontop of.
----	-------------	--

10.72.3.25 void gazebo::physics::Entity::PlaceOnNearestEntityBelow ()

Move this entity to be ontop of the nearest entity below.

10.72.3.26 virtual void gazebo::physics::Entity::Reset () [virtual]

Reset the entity.

Reimplemented from **gazebo::physics::Base** (p. 179).

Reimplemented in **gazebo::physics::Model** (p. 689), and **gazebo::physics::Link** (p. 613).

10.72.3.27 void gazebo::physics::Entity::SetAnimation (const common::PoseAnimationPtr & _anim, boost::function< void()> _onComplete)

Set an animation for this entity.

Parameters

in	_anim	Pose animation.
in	_onComplete	Callback for when the animation completes.

10.72.3.28 void gazebo::physics::Entity::SetAnimation (common::PoseAnimationPtr _anim)

Set an animation for this entity.

Parameters

in	<i>_anim</i>	Pose animation.
----	--------------	-----------------

10.72.3.29 void gazebo::physics::Entity::SetCanonicalLink (bool *_value*)

Set to true if this entity is a canonical link for a model.

Parameters

in	<i>_value</i>	True if the link is canonical.
----	---------------	--------------------------------

10.72.3.30 void gazebo::physics::Entity::SetInitialRelativePose (const math::Pose & *_pose*)

Set the initial pose.

Parameters

in	<i>_pose</i>	The initial pose.
----	--------------	-------------------

10.72.3.31 virtual void gazebo::physics::Entity::SetName (const std::string & *_name*) [virtual]

Set the name of the entity.

Parameters

in	<i>_name</i>	The new name.
----	--------------	---------------

Reimplemented from **gazebo::physics::Base** (p. 179).

10.72.3.32 void gazebo::physics::Entity::SetRelativePose (const math::Pose & *_pose*, bool *_notify* = true, bool *_publish* = true)

Set the pose of the entity relative to its parent.

Parameters

in	<i>_pose</i>	The new pose.
in	<i>_notify</i>	True = tell children of the pose change.
in	<i>_publish</i>	True to publish the pose.

10.72.3.33 void gazebo::physics::Entity::SetStatic (const bool & *_static*)

Set whether this entity is static: immovable.

Parameters

in	<i>_static</i>	True = static.
----	----------------	----------------

10.72.3.34 `void gazebo::physics::Entity::SetWorldPose (const math::Pose & _pose, bool _notify = true, bool _publish = true)`

Set the world pose of the entity.

Parameters

in	<code>_pose</code>	The new world pose.
in	<code>_notify</code>	True = tell children of the pose change.
in	<code>_publish</code>	True to publish the pose.

10.72.3.35 `void gazebo::physics::Entity::SetWorldTwist (const math::Vector3 & _linear, const math::Vector3 & _angular, bool _updateChildren = true)`

Set angular and linear rates of an **physics::Entity** (p. 404).

Parameters

in	<code>_linear</code>	Linear twist.
in	<code>_angular</code>	Angular twist.
in	<code>_updateChildren</code>	True to pass this update to child entities.

10.72.3.36 `virtual void gazebo::physics::Entity::StopAnimation () [virtual]`

Stop the current animation, if any.

Reimplemented in **gazebo::physics::Model** (p. 692).

10.72.3.37 `virtual void gazebo::physics::Entity::UpdateParameters (sdf::ElementPtr _sdf) [virtual]`

Update the parameters using new sdf values.

Parameters

in	<code>_sdf</code>	SDF to update from.
----	-------------------	---------------------

Reimplemented from **gazebo::physics::Base** (p. 180).

Reimplemented in **gazebo::physics::Actor** (p. 143), **gazebo::physics::Link** (p. 617), **gazebo::physics::Model** (p. 692), and **gazebo::physics::Collision** (p. 245).

10.72.4 Member Data Documentation

10.72.4.1 `common::PoseAnimationPtr gazebo::physics::Entity::animation [protected]`

Current pose animation.

10.72.4.2 `event::ConnectionPtr gazebo::physics::Entity::animationConnection [protected]`

Connection used to update an animation.

10.72.4.3 `math::Pose gazebo::physics::Entity::animationStartPose` [protected]

Start pose of an animation.

10.72.4.4 `std::vector<event::ConnectionPtr> gazebo::physics::Entity::connections` [protected]

All our event connections.

10.72.4.5 `math::Pose gazebo::physics::Entity::dirtyPose` [protected]

The pose set by a physics engine.

10.72.4.6 `transport::NodePtr gazebo::physics::Entity::node` [protected]

Communication node.

10.72.4.7 `EntityPtr gazebo::physics::Entity::parentEntity` [protected]

A helper that prevents numerous `dynamic_casts`.

10.72.4.8 `common::Time gazebo::physics::Entity::prevAnimationTime` [protected]

Previous time an animation was updated.

10.72.4.9 `transport::PublisherPtr gazebo::physics::Entity::requestPub` [protected]

Request publisher.

10.72.4.10 `math::Vector3 gazebo::physics::Entity::scale` [protected]

Scale of the entity.

10.72.4.11 `transport::PublisherPtr gazebo::physics::Entity::visPub` [protected]

Visual publisher.

10.72.4.12 `msgs::Visual* gazebo::physics::Entity::visualMsg` [protected]

Visual message container.

The documentation for this class was generated from the following file:

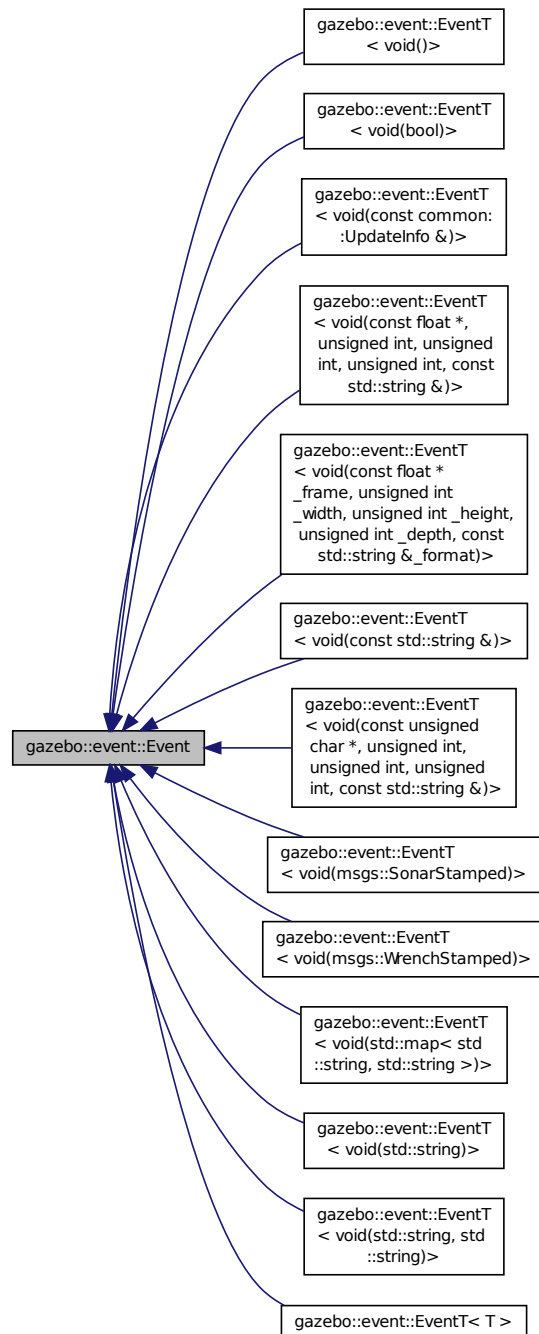
- **Entity.hh**

10.73 gazebo::event::Event Class Reference

Base class for all events.

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::event::Event:



Public Member Functions

- **Event** ()
Constructor.
- virtual **~Event** ()
Destructor.
- virtual void **Disconnect** (ConnectionPtr _c)=0
Disconnect.
- virtual void **Disconnect** (int _id)=0
Disconnect.
- bool **GetSignaled** () const
Get whether this event has been signaled.

Protected Member Functions

- **Event** (EventPrivate &_d)
Allow subclasses to initialize their own data pointer.

Protected Attributes

- **EventPrivate * dataPtr**
Data pointer.

10.73.1 Detailed Description

Base class for all events.

10.73.2 Constructor & Destructor Documentation

10.73.2.1 gazebo::event::Event::Event ()

Constructor.

10.73.2.2 virtual gazebo::event::Event::~~Event () [virtual]

Destructor.

10.73.2.3 gazebo::event::Event::Event (EventPrivate &_d) [protected]

Allow subclasses to initialize their own data pointer.

Parameters

in	_d	Reference to data pointer.
----	----	----------------------------

10.73.3 Member Function Documentation

10.73.3.1 `virtual void gazebo::event::Event::Disconnect (ConnectionPtr _c) [pure virtual]`

Disconnect.

Parameters

in	_c	A pointer to a connection
----	----	---------------------------

Implemented in `gazebo::event::EventT< T >` (p. 48), `gazebo::event::EventT< void(msgs::SonarStamped)>` (p. 48), `gazebo::event::EventT< void(std::string)>` (p. 48), `gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>` (p. 48), `gazebo::event::EventT< void(msgs::WrenchStamped)>` (p. 48), `gazebo::event::EventT< void(const float *_frame, unsigned int _width, unsigned int _height, unsigned int _depth, const std::string &_format)>` (p. 48), `gazebo::event::EventT< void(const std::string &)>` (p. 48), `gazebo::event::EventT< void(std::map< std::string, std::string >)>` (p. 48), `gazebo::event::EventT< void()>` (p. 48), `gazebo::event::EventT< void(const common::UpdateInfo &)>` (p. 48), `gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>` (p. 48), `gazebo::event::EventT< void(std::string, std::string)>` (p. 48), and `gazebo::event::EventT< void(bool)>` (p. 48).

10.73.3.2 `virtual void gazebo::event::Event::Disconnect (int _id) [pure virtual]`

Disconnect.

Parameters

in	_id	Integer ID of a connection
----	-----	----------------------------

Implemented in `gazebo::event::EventT< T >` (p. 48), `gazebo::event::EventT< void(msgs::SonarStamped)>` (p. 48), `gazebo::event::EventT< void(std::string)>` (p. 48), `gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>` (p. 48), `gazebo::event::EventT< void(msgs::WrenchStamped)>` (p. 48), `gazebo::event::EventT< void(const float *_frame, unsigned int _width, unsigned int _height, unsigned int _depth, const std::string &_format)>` (p. 48), `gazebo::event::EventT< void(const std::string &)>` (p. 48), `gazebo::event::EventT< void(std::map< std::string, std::string >)>` (p. 48), `gazebo::event::EventT< void()>` (p. 48), `gazebo::event::EventT< void(const common::UpdateInfo &)>` (p. 48), `gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>` (p. 48), `gazebo::event::EventT< void(std::string, std::string)>` (p. 48), and `gazebo::event::EventT< void(bool)>` (p. 48).

10.73.3.3 `bool gazebo::event::Event::GetSignaled () const`

Get whether this event has been signaled.

Returns

True if the event has been signaled.

10.73.4 Member Data Documentation

10.73.4.1 `EventPrivate* gazebo::event::Event::dataPtr [protected]`

Data pointer.

Referenced by `gazebo::event::EventT< T >::EventT()`.

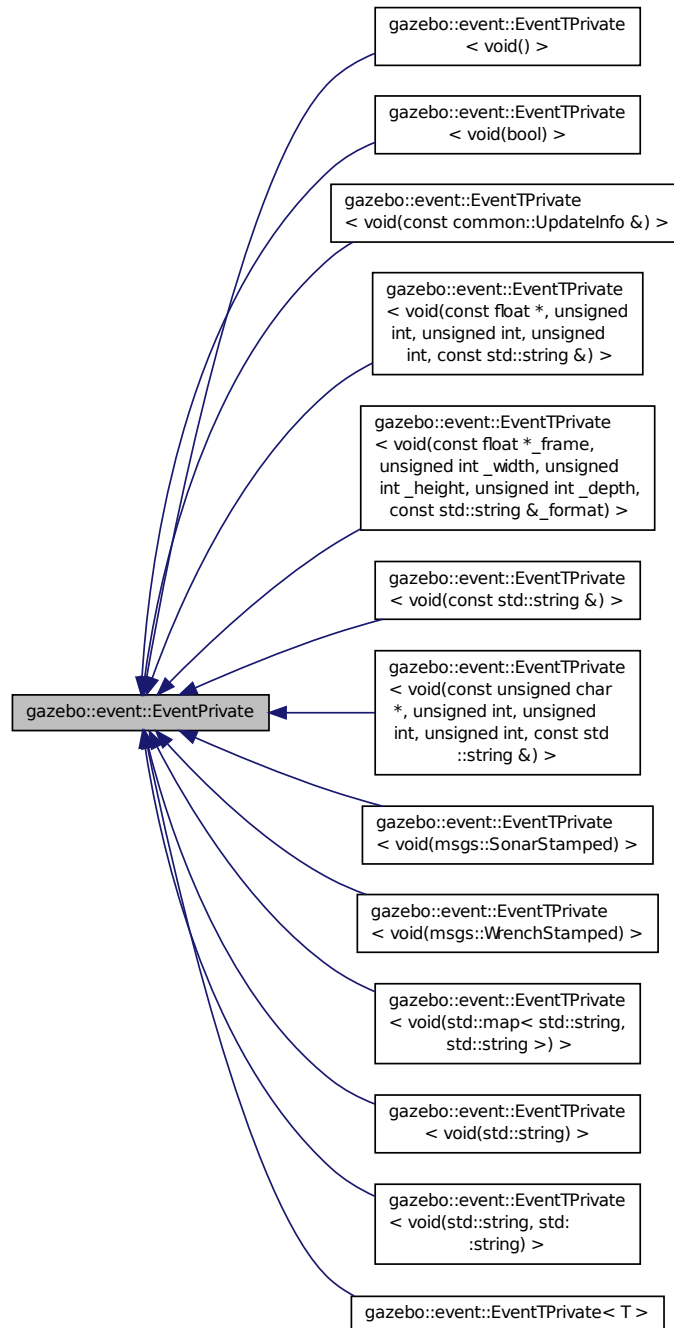
The documentation for this class was generated from the following file:

- [Event.hh](#)

10.74 gazebo::event::EventPrivate Class Reference

```
#include <Event.hh>
```

Inheritance diagram for gazebo::event::EventPrivate:



Public Member Functions

- **EventPrivate** ()

Public Attributes

- bool **signaled**

True if the event has been signaled.

10.74.1 Constructor & Destructor Documentation

10.74.1.1 gazebo::event::EventPrivate::EventPrivate ()

10.74.2 Member Data Documentation

10.74.2.1 bool gazebo::event::EventPrivate::signaled

True if the event has been signaled.

The documentation for this class was generated from the following file:

- **Event.hh**

10.75 gazebo::event::Events Class Reference

An **Event** (p. 416) class to get notifications for simulator events.

```
#include <common/common.hh>
```

Static Public Member Functions

- template<typename T >
static **ConnectionPtr ConnectAddEntity** (T _subscriber)
Connect a boost::slot the the add entity signal.
- template<typename T >
static **ConnectionPtr ConnectCreateEntity** (T _subscriber)
Connect a boost::slot the the add entity signal.
- template<typename T >
static **ConnectionPtr ConnectDeleteEntity** (T _subscriber)
Connect a boost::slot the delete entity.
- template<typename T >
static **ConnectionPtr ConnectDiagTimerStart** (T _subscriber)
Connect a boost::slot the diagnostic timer start signal.
- template<typename T >
static **ConnectionPtr ConnectDiagTimerStop** (T _subscriber)
Connect a boost::slot the diagnostic timer stop signal.
- template<typename T >
static **ConnectionPtr ConnectPause** (T _subscriber)
Connect a boost::slot the the pause signal.
- template<typename T >
static **ConnectionPtr ConnectPostRender** (T _subscriber)
Connect a boost::slot the post render update signal.

- `template<typename T >`
static `ConnectionPtr ConnectPreRender` (`T _subscriber`)
Render start signal.
- `template<typename T >`
static `ConnectionPtr ConnectRender` (`T _subscriber`)
Connect a boost::slot the render update signal.
- `template<typename T >`
static `ConnectionPtr ConnectSetSelectedEntity` (`T _subscriber`)
Connect a boost::slot the set selected entity.
- `template<typename T >`
static `ConnectionPtr ConnectSigInt` (`T _subscriber`)
Connect a boost::slot to the sigint event.
- `template<typename T >`
static `ConnectionPtr ConnectStep` (`T _subscriber`)
Connect a boost::slot the the step signal.
- `template<typename T >`
static `ConnectionPtr ConnectStop` (`T _subscriber`)
Connect a boost::slot the the stop signal.
- `template<typename T >`
static `ConnectionPtr ConnectWorldCreated` (`T _subscriber`)
Connect a boost::slot the the world created signal.
- `template<typename T >`
static `ConnectionPtr ConnectWorldUpdateBegin` (`T _subscriber`)
Connect a boost::slot the the world update start signal.
- `template<typename T >`
static `ConnectionPtr ConnectWorldUpdateEnd` (`T _subscriber`)
Connect a boost::slot the the world update end signal.
- **static void `DisconnectAddEntity` (`ConnectionPtr _subscriber`)**
Disconnect a boost::slot the the add entity signal.
- **static void `DisconnectCreateEntity` (`ConnectionPtr _subscriber`)**
Disconnect a boost::slot the the add entity signal.
- **static void `DisconnectDeleteEntity` (`ConnectionPtr _subscriber`)**
Disconnect a boost::slot the delete entity.
- **static void `DisconnectDiagTimerStart` (`ConnectionPtr _subscriber`)**
Disconnect a boost::slot the diagnostic timer start signal.
- **static void `DisconnectDiagTimerStop` (`ConnectionPtr _subscriber`)**
Disconnect a boost::slot the diagnostic timer stop signal.
- **static void `DisconnectPause` (`ConnectionPtr _subscriber`)**
Disconnect a boost::slot the the pause signal.
- **static void `DisconnectPostRender` (`ConnectionPtr _subscriber`)**
Disconnect a boost::slot the post render update signal.
- **static void `DisconnectPreRender` (`ConnectionPtr _subscriber`)**
Disconnect a render start signal.
- **static void `DisconnectRender` (`ConnectionPtr _subscriber`)**
Disconnect a boost::slot the render update signal.
- **static void `DisconnectSetSelectedEntity` (`ConnectionPtr _subscriber`)**
Disconnect a boost::slot the set selected entity.
- **static void `DisconnectSigInt` (`ConnectionPtr _subscriber`)**

- Disconnect a boost::slot to the sigint event.*

 - static void **DisconnectStep** (**ConnectionPtr** _subscriber)

Disconnect a boost::slot the the step signal.
- static void **DisconnectStop** (**ConnectionPtr** _subscriber)

Disconnect a boost::slot the the stop signal.
- static void **DisconnectWorldCreated** (**ConnectionPtr** _subscriber)

Disconnect a boost::slot the the world created signal.
- static void **DisconnectWorldUpdateBegin** (**ConnectionPtr** _subscriber)

Disconnect a boost::slot the the world update start signal.
- static void **DisconnectWorldUpdateEnd** (**ConnectionPtr** _subscriber)

Disconnect a boost::slot the the world update end signal.

Static Public Attributes

- static **EventT**< void(std::string)> **addEntity**

An entity has been added.
- static **EventT**< void(std::string)> **deleteEntity**

An entity has been deleted.
- static **EventT**< void(std::string)> **diagTimerStart**

Diagnostic timer start.
- static **EventT**< void(std::string)> **diagTimerStop**

Diagnostic timer stop.
- static **EventT**< void(std::string)> **entityCreated**

An entity has been created.
- static **EventT**< void(bool)> **pause**

Pause signal.
- static **EventT**< void()> **postRender**

Post-Render.
- static **EventT**< void()> **preRender**

Pre-render.
- static **EventT**< void()> **render**

Render.
- static **EventT**< void(std::string, std::string)> **setSelectedEntity**

An entity has been selected.
- static **EventT**< void()> **sigInt**

Simulation stop signal.
- static **EventT**< void()> **step**

Step the simulation once signal.
- static **EventT**< void()> **stop**

Simulation stop signal.
- static **EventT**< void(std::string)> **worldCreated**

A world has been created.
- static **EventT**< void(const **common::UpdateInfo** &)> **worldUpdateBegin**

World update has started.
- static **EventT**< void()> **worldUpdateEnd**

World update has ended.

10.75.1 Detailed Description

An **Event** (p. 416) class to get notifications for simulator events.

10.75.2 Member Function Documentation

10.75.2.1 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectAddEntity (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the the add entity signal.

Parameters

in	<code><i>_subscriber</i></code>	the subscriber to this event
----	---------------------------------	------------------------------

Returns

a connection

10.75.2.2 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectCreateEntity (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the the add entity signal.

Parameters

in	<code><i>_subscriber</i></code>	the subscriber to this event
----	---------------------------------	------------------------------

Returns

a connection

10.75.2.3 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDeleteEntity (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the delete entity.

Parameters

in	<code><i>_subscriber</i></code>	the subscriber to this event
----	---------------------------------	------------------------------

Returns

a connection

10.75.2.4 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStart (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the diagnostic timer start signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

10.75.2.5 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStop (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the diagnostic timer stop signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

10.75.2.6 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPause (T _subscriber)` `[inline],`
`[static]`

Connect a boost::slot the the pause signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

10.75.2.7 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPostRender (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the post render update signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

10.75.2.8 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPreRender (T _subscriber)`
`[inline],[static]`

Render start signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

Returns

a connection

10.75.2.9 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectRender (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the render update signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

Returns

a connection

10.75.2.10 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectSetSelectedEntity (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the set selected entity.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

Returns

a connection

10.75.2.11 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectSigInt (T _subscriber)`
`[inline],[static]`

Connect a boost::slot to the sigint event.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

10.75.2.12 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStep (T _subscriber) [inline], [static]`

Connect a boost::slot the the step signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

10.75.2.13 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStop (T _subscriber) [inline], [static]`

Connect a boost::slot the the stop signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

References gazebo::stop().

10.75.2.14 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldCreated (T _subscriber) [inline], [static]`

Connect a boost::slot the the world created signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

10.75.2.15 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldUpdateBegin (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the the world update start signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

Returns

a connection

10.75.2.16 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldUpdateEnd (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the the world update end signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

Returns

a connection

10.75.2.17 `static void gazebo::event::Events::DisconnectAddEntity (ConnectionPtr _subscriber)` `[inline],[static]`

Disconnect a boost::slot the the add entity signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

10.75.2.18 `static void gazebo::event::Events::DisconnectCreateEntity (ConnectionPtr _subscriber)` `[inline],[static]`

Disconnect a boost::slot the the add entity signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

10.75.2.19 `static void gazebo::event::Events::DisconnectDeleteEntity (ConnectionPtr _subscriber)` `[inline],[static]`

Disconnect a boost::slot the delete entity.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

10.75.2.20 static void gazebo::event::Events::DisconnectDiagTimerStart (ConnectionPtr *_subscriber*) [inline],
[static]

Disconnect a boost::slot the diagnostic timer start signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

10.75.2.21 static void gazebo::event::Events::DisconnectDiagTimerStop (ConnectionPtr *_subscriber*) [inline],
[static]

Disconnect a boost::slot the diagnostic timer stop signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

10.75.2.22 static void gazebo::event::Events::DisconnectPause (ConnectionPtr *_subscriber*) [inline],[static]

Disconnect a boost::slot the the pause signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

10.75.2.23 static void gazebo::event::Events::DisconnectPostRender (ConnectionPtr *_subscriber*) [inline],
[static]

Disconnect a boost::slot the post render update signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

10.75.2.24 static void gazebo::event::Events::DisconnectPreRender (ConnectionPtr *_subscriber*) [inline],[static]

Disconnect a render start signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

10.75.2.25 `static void gazebo::event::Events::DisconnectRender (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the render update signal.

Parameters

in	_subscriber	the subscriber to this event
----	-------------	------------------------------

10.75.2.26 `static void gazebo::event::Events::DisconnectSetSelectedEntity (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the set selected entity.

Parameters

in	_subscriber	the subscriber to this event
----	-------------	------------------------------

10.75.2.27 `static void gazebo::event::Events::DisconnectSigInt (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot to the sigint event.

Parameters

in	_subscriber	the subscriber to this event
----	-------------	------------------------------

10.75.2.28 `static void gazebo::event::Events::DisconnectStep (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the the step signal.

Parameters

in	_subscriber	the subscriber to this event
----	-------------	------------------------------

10.75.2.29 `static void gazebo::event::Events::DisconnectStop (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the the stop signal.

Parameters

in	_subscriber	the subscriber to this event
----	-------------	------------------------------

References gazebo::stop().

10.75.2.30 `static void gazebo::event::Events::DisconnectWorldCreated (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the the world created signal.

10.75.2.31 `static void gazebo::event::Events::DisconnectWorldUpdateBegin (ConnectionPtr _subscriber) [static]`

Disconnect a boost::slot the the world update start signal.

Parameters

in	<code><i>_subscriber</i></code>	the subscriber to this event
----	---------------------------------	------------------------------

10.75.2.32 `static void gazebo::event::Events::DisconnectWorldUpdateEnd (ConnectionPtr _subscriber) [inline], [static]`

Disconnect a boost::slot the the world update end signal.

Parameters

in	<code><i>_subscriber</i></code>	the subscriber to this event
----	---------------------------------	------------------------------

10.75.3 Member Data Documentation

10.75.3.1 `EventT<void (std::string)> gazebo::event::Events::addEntity [static]`

An entity has been added.

10.75.3.2 `EventT<void (std::string)> gazebo::event::Events::deleteEntity [static]`

An entity has been deleted.

10.75.3.3 `EventT<void (std::string)> gazebo::event::Events::diagTimerStart [static]`

Diagnostic timer start.

10.75.3.4 `EventT<void (std::string)> gazebo::event::Events::diagTimerStop [static]`

Diagnostic timer stop.

10.75.3.5 `EventT<void (std::string)> gazebo::event::Events::entityCreated [static]`

An entity has been created.

10.75.3.6 `EventT<void (bool)> gazebo::event::Events::pause [static]`

Pause signal.

10.75.3.7 `EventT<void ()> gazebo::event::Events::postRender [static]`

Post-Render.

10.75.3.8 **EventT<void ()> gazebo::event::Events::preRender** [static]

Pre-render.

10.75.3.9 **EventT<void ()> gazebo::event::Events::render** [static]

Render.

10.75.3.10 **EventT<void (std::string, std::string)> gazebo::event::Events::setSelectedEntity** [static]

An entity has been selected.

10.75.3.11 **EventT<void ()> gazebo::event::Events::sigInt** [static]

Simulation stop signal.

10.75.3.12 **EventT<void ()> gazebo::event::Events::step** [static]

Step the simulation once signal.

10.75.3.13 **EventT<void ()> gazebo::event::Events::stop** [static]

Simulation stop signal.

10.75.3.14 **EventT<void (std::string)> gazebo::event::Events::worldCreated** [static]

A world has been created.

10.75.3.15 **EventT<void (const common::UpdateInfo &)> gazebo::event::Events::worldUpdateBegin** [static]

World update has started.

10.75.3.16 **EventT<void ()> gazebo::event::Events::worldUpdateEnd** [static]

World update has ended.

The documentation for this class was generated from the following file:

- **Events.hh**

10.76 gazebo::rendering::Events Class Reference

Base class for rendering events.

```
#include <rendering/rendering.hh>
```

Static Public Member Functions

- `template<typename T >`
`static event::ConnectionPtr ConnectCreateScene (T _subscriber)`
Connect to a scene created event.
- `template<typename T >`
`static event::ConnectionPtr ConnectRemoveScene (T _subscriber)`
Connect to a scene removed event.
- `static void DisconnectCreateScene (event::ConnectionPtr _connection)`
Disconnect from a scene created event.
- `static void DisconnectRemoveScene (event::ConnectionPtr _connection)`
Disconnect from a scene removed event.

Static Public Attributes

- `static event::EventT < void(const std::string &)> createScene`
The event used to trigger a create scene event.
- `static event::EventT < void(const std::string &)> removeScene`
The event used to trigger a remove scene event.

10.76.1 Detailed Description

Base class for rendering events.

10.76.2 Member Function Documentation

10.76.2.1 `template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectCreateScene (T _subscriber) [inline],[static]`

Connect to a scene created event.

Parameters

in	<code>_subscriber</code>	Callback to trigger when event occurs.
----	--------------------------	--

Returns

Pointer the connection. This must stay in scope.

10.76.2.2 `template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectRemoveScene (T _subscriber) [inline],[static]`

Connect to a scene removed event.

Parameters

in	<code>_subscriber</code>	Callback to trigger when event occurs.
----	--------------------------	--

Returns

Pointer the connection. This must stay in scope.

10.76.2.3 `static void gazebo::rendering::Events::DisconnectCreateScene (event::ConnectionPtr _connection)` `[inline]`,
`[static]`

Disconnect from a scene created event.

Parameters

<code>in</code>	<code>_connection</code>	The connection to disconnect.
-----------------	--------------------------	-------------------------------

10.76.2.4 `static void gazebo::rendering::Events::DisconnectRemoveScene (event::ConnectionPtr _connection)`
`[inline]`, `[static]`

Disconnect from a scene removed event.

Parameters

<code>in</code>	<code>_connection</code>	The connection to disconnect.
-----------------	--------------------------	-------------------------------

10.76.3 Member Data Documentation

10.76.3.1 `event::EventT<void (const std::string &)> gazebo::rendering::Events::createScene` `[static]`

The event used to trigger a create scene event.

10.76.3.2 `event::EventT<void (const std::string &)> gazebo::rendering::Events::removeScene` `[static]`

The event used to trigger a remove scene event.

The documentation for this class was generated from the following file:

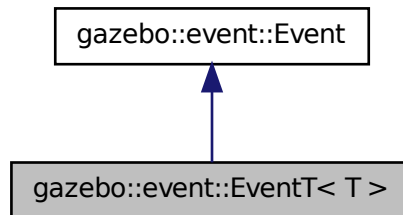
- **RenderEvents.hh**

10.77 gazebo::event::EventT< T > Class Template Reference

A class for event processing.

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::event::EventT< T >:



Public Member Functions

- **EventT** ()
Constructor.
- virtual **~EventT** ()
Destructor.
- **ConnectionPtr Connect** (const boost::function< T > &_subscriber)
Connect a callback to this event.
- unsigned int **ConnectionCount** () const
Get the number of connections.
- virtual void **Disconnect** (**ConnectionPtr** _c)
Disconnect a callback to this event.
- virtual void **Disconnect** (int _id)
Disconnect a callback to this event.
- void **operator**() ()
Access the signal.
- template<typename P >
void **operator**() (const P &_p)
Signal the event with one parameter.
- template<typename P1 , typename P2 >
void **operator**() (const P1 &_p1, const P2 &_p2)
Signal the event with two parameters.
- template<typename P1 , typename P2 , typename P3 >
void **operator**() (const P1 &_p1, const P2 &_p2, const P3 &_p3)
Signal the event with three parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 >
void **operator**() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4)
Signal the event with four parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 >
void **operator**() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5)
Signal the event with five parameters.

- `template<typename P1, typename P2, typename P3, typename P4, typename P5, typename P6 >`
`void operator() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6)`
Signal the event with six parameters.
- `template<typename P1, typename P2, typename P3, typename P4, typename P5, typename P6, typename P7 >`
`void operator() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7)`
Signal the event with seven parameters.
- `template<typename P1, typename P2, typename P3, typename P4, typename P5, typename P6, typename P7, typename P8 >`
`void operator() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8)`
Signal the event with eight parameters.
- `template<typename P1, typename P2, typename P3, typename P4, typename P5, typename P6, typename P7, typename P8, typename P9 >`
`void operator() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9)`
Signal the event with nine parameters.
- `template<typename P1, typename P2, typename P3, typename P4, typename P5, typename P6, typename P7, typename P8, typename P9, typename P10 >`
`void operator() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9, const P10 &_p10)`
Signal the event with ten parameters.
- `void Signal ()`
Signal the event for all subscribers.
- `template<typename P >`
`void Signal (const P &_p)`
Signal the event with one parameter.
- `template<typename P1, typename P2 >`
`void Signal (const P1 &_p1, const P2 &_p2)`
Signal the event with two parameter.
- `template<typename P1, typename P2, typename P3 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3)`
Signal the event with three parameter.
- `template<typename P1, typename P2, typename P3, typename P4 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4)`
Signal the event with four parameter.
- `template<typename P1, typename P2, typename P3, typename P4, typename P5 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5)`
Signal the event with five parameter.
- `template<typename P1, typename P2, typename P3, typename P4, typename P5, typename P6 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6)`
Signal the event with six parameter.
- `template<typename P1, typename P2, typename P3, typename P4, typename P5, typename P6, typename P7 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7)`
Signal the event with seven parameter.
- `template<typename P1, typename P2, typename P3, typename P4, typename P5, typename P6, typename P7, typename P8 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8)`
Signal the event with eight parameter.

- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9)`
Signal the event with nine parameter.
- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9, const P10 &_p10)`
Signal the event with ten parameter.

Additional Inherited Members

10.77.1 Detailed Description

`template<typename T> class gazebo::event::EventT< T >`

A class for event processing.

10.77.2 Member Function Documentation

10.77.2.1 `template<typename T> void gazebo::event::EventT< T >::operator()() [inline]`

Access the signal.

10.77.2.2 `template<typename T> template<typename P > void gazebo::event::EventT< T >::operator()(const P & _p) [inline]`

Signal the event with one parameter.

Parameters

in	_p	the parameter.
----	----	----------------

10.77.2.3 `template<typename T> template<typename P1 , typename P2 > void gazebo::event::EventT< T >::operator()(const P1 & _p1, const P2 & _p2) [inline]`

Signal the event with two parameters.

Parameters

in	_p1	the first parameter.
in	_p2	the second parameter.

10.77.2.4 `template<typename T> template<typename P1 , typename P2 , typename P3 > void gazebo::event::EventT< T >::operator()(const P1 & _p1, const P2 & _p2, const P3 & _p3) [inline]`

Signal the event with three parameters.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.

10.77.2.5 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4) [inline]`

Signal the event with four parameters.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.

10.77.2.6 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5) [inline]`

Signal the event with five parameters.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.
in	<code>_p5</code>	the fift parameter.

10.77.2.7 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6) [inline]`

Signal the event with six parameters.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.
in	<code>_p5</code>	the fift parameter.
in	<code>_p6</code>	the sixt parameter.

10.77.2.8 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7) [inline]`

Signal the event with seven parameters.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.
in	<code>_p5</code>	the fifth parameter.
in	<code>_p6</code>	the sixth parameter.
in	<code>_p7</code>	the seventh parameter.

10.77.2.9 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8) [inline]`

Signal the event with eight parameters.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.
in	<code>_p5</code>	the fifth parameter.
in	<code>_p6</code>	the sixth parameter.
in	<code>_p7</code>	the seventh parameter.
in	<code>_p8</code>	the eighth parameter.

10.77.2.10 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9) [inline]`

Signal the event with nine parameters.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.
in	<code>_p5</code>	the fifth parameter.
in	<code>_p6</code>	the sixth parameter.
in	<code>_p7</code>	the seventh parameter.
in	<code>_p8</code>	the eighth parameter.
in	<code>_p9</code>	the ninth parameter.

10.77.2.11 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9, const P10 & _p10) [inline]`

Signal the event with ten parameters.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.
in	<code>_p5</code>	the fifth parameter.
in	<code>_p6</code>	the sixth parameter.
in	<code>_p7</code>	the seventh parameter.
in	<code>_p8</code>	the eighth parameter.
in	<code>_p9</code>	the ninth parameter.
in	<code>_p10</code>	the tenth parameter.

10.77.2.12 `template<typename T> void gazebo::event::EventT< T >::Signal () [inline]`

Signal the event for all subscribers.

10.77.2.13 `template<typename T> template<typename P > void gazebo::event::EventT< T >::Signal (const P & _p) [inline]`

Signal the event with one parameter.

Parameters

in	<code>_p</code>	parameter.
----	-----------------	------------

10.77.2.14 `template<typename T> template<typename P1 , typename P2 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2) [inline]`

Signal the event with two parameter.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.

10.77.2.15 `template<typename T> template<typename P1 , typename P2 , typename P3 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2, const P3 & _p3) [inline]`

Signal the event with three parameter.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.

10.77.2.16 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 > void gazebo::event::EventT< T >::Signal (const P1 & .p1, const P2 & .p2, const P3 & .p3, const P4 & .p4) [inline]`

Signal the event with four parameter.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.

10.77.2.17 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 > void gazebo::event::EventT< T >::Signal (const P1 & .p1, const P2 & .p2, const P3 & .p3, const P4 & .p4, const P5 & .p5) [inline]`

Signal the event with five parameter.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.
in	<code>_p5</code>	the fifth parameter.

10.77.2.18 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 > void gazebo::event::EventT< T >::Signal (const P1 & .p1, const P2 & .p2, const P3 & .p3, const P4 & .p4, const P5 & .p5, const P6 & .p6) [inline]`

Signal the event with six parameter.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.
in	<code>_p5</code>	the fifth parameter.
in	<code>_p6</code>	the sixth parameter.

10.77.2.19 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7) [inline]`

Signal the event with seven parameter.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.
in	<code>_p5</code>	the fifth parameter.
in	<code>_p6</code>	the sixth parameter.
in	<code>_p7</code>	the seventh parameter.

10.77.2.20 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8) [inline]`

Signal the event with eight parameter.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.
in	<code>_p5</code>	the fifth parameter.
in	<code>_p6</code>	the sixth parameter.
in	<code>_p7</code>	the seventh parameter.
in	<code>_p8</code>	the eighth parameter.

10.77.2.21 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9) [inline]`

Signal the event with nine parameter.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.
in	<code>_p5</code>	the fifth parameter.
in	<code>_p6</code>	the sixth parameter.
in	<code>_p7</code>	the seventh parameter.
in	<code>_p8</code>	the eighth parameter.
in	<code>_p9</code>	the ninth parameter.

10.77.2.22 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9, const P10 & _p10) [inline]`

Signal the event with ten parameter.

Parameters

in	<code>_p1</code>	the first parameter.
in	<code>_p2</code>	the second parameter.
in	<code>_p3</code>	the second parameter.
in	<code>_p4</code>	the first parameter.
in	<code>_p5</code>	the fifth parameter.
in	<code>_p6</code>	the sixth parameter.
in	<code>_p7</code>	the seventh parameter.
in	<code>_p8</code>	the eighth parameter.
in	<code>_p9</code>	the ninth parameter.
in	<code>_p10</code>	the tenth parameter.

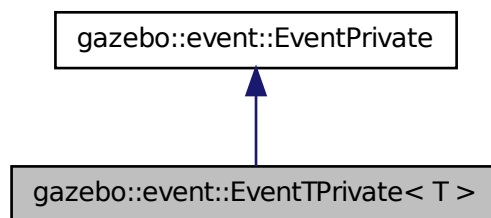
The documentation for this class was generated from the following file:

- **Event.hh**

10.78 gazebo::event::EventTPriate< T > Class Template Reference

```
#include <Event.hh>
```

Inheritance diagram for gazebo::event::EventTPriate< T >:



Public Attributes

- EvtConnectionMap **connections**
Array of connection callbacks.
- boost::mutex **connectionsEraseMutex**
A thread lock.

- `std::vector< int > connectionsToErase`

Set of connections to erased.

Additional Inherited Members

10.78.1 Member Data Documentation

10.78.1.1 `template<typename T> EvtConnectionMap gazebo::event::EventTPriate< T >::connections`

Array of connection callbacks.

10.78.1.2 `template<typename T> boost::mutex gazebo::event::EventTPriate< T >::connectionsEraseMutex`

A thread lock.

10.78.1.3 `template<typename T> std::vector<int> gazebo::event::EventTPriate< T >::connectionsToErase`

Set of connections to erased.

The documentation for this class was generated from the following file:

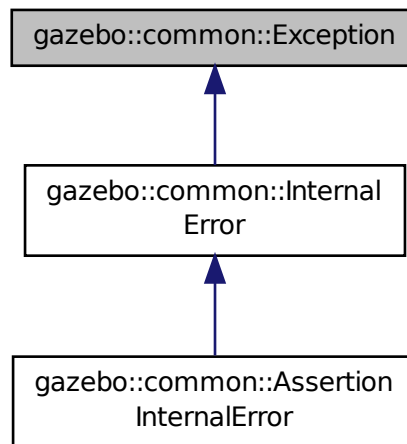
- **Event.hh**

10.79 gazebo::common::Exception Class Reference

Class for generating exceptions.

```
#include <common/common.hh>
```


Inheritance diagram for gazebo::common::Exception:



Public Member Functions

- **Exception** ()
Constructor.
- **Exception** (const char *_file, int _line, std::string _msg)
Default constructor.
- virtual **~Exception** ()
Destructor.
- std::string **GetErrorFile** () const
Return the error function.
- std::string **GetErrorStr** () const
Return the error string.
- void **Print** () const
Print the exception to std out.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::common::Exception** &_err)
stream insertion operator for Gazebo Error

10.79.1 Detailed Description

Class for generating exceptions.

10.79.2 Constructor & Destructor Documentation

10.79.2.1 gazebo::common::Exception::Exception ()

Constructor.

10.79.2.2 gazebo::common::Exception::Exception (const char * *_file*, int *_line*, std::string *_msg*)

Default constructor.

Parameters

in	<i>_file</i>	File name
in	<i>_line</i>	Line number where the error occurred
in	<i>_msg</i>	Error message

10.79.2.3 virtual gazebo::common::Exception::~~Exception () [virtual]

Destructor.

10.79.3 Member Function Documentation

10.79.3.1 std::string gazebo::common::Exception::GetErrorFile () const

Return the error function.

Returns

The error function name

10.79.3.2 std::string gazebo::common::Exception::GetErrorStr () const

Return the error string.

Returns

The error string

10.79.3.3 void gazebo::common::Exception::Print () const

Print the exception to std out.

10.79.4 Friends And Related Function Documentation

10.79.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::common::Exception & *_err*) [friend]

stream insertion operator for Gazebo Error

Parameters

in	<code>_out</code>	the output stream
in	<code>_err</code>	the exception

The documentation for this class was generated from the following file:

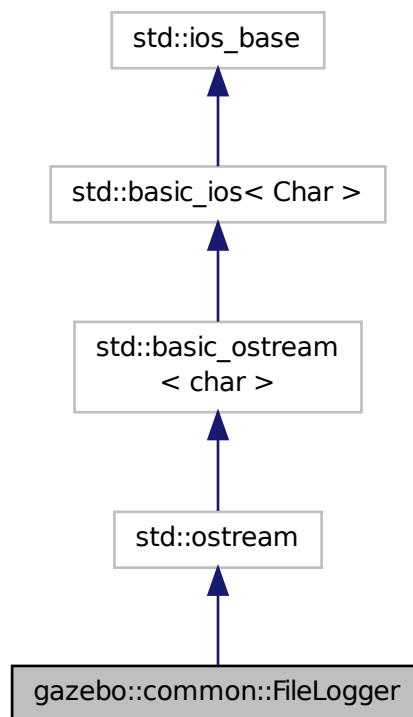
- **Exception.hh**

10.80 gazebo::common::FileLogger Class Reference

A logger that outputs messages to a file.

```
#include <Console.hh>
```

Inheritance diagram for gazebo::common::FileLogger:



Classes

- class **Buffer**

String buffer for the file logger.

Public Member Functions

- **FileLogger** (const std::string &_filename="")
Constructor.
- virtual **~FileLogger** ()
Destructor.
- void **Init** (const std::string &_filename)
Initialize the file logger.
- virtual **FileLogger & operator()** ()
Output a filename and line number, then return a reference to the logger.
- virtual **FileLogger & operator()** (const std::string &_file, int _line)
Output a filename and line number, then return a reference to the logger.

10.80.1 Detailed Description

A logger that outputs messages to a file.

10.80.2 Constructor & Destructor Documentation

10.80.2.1 gazebo::common::FileLogger::FileLogger (const std::string & _filename = " ")

Constructor.

Parameters

in	_filename	Filename to write into. If empty, FileLogger::Init (p. 448) must be called separately.
----	-----------	---

10.80.2.2 virtual gazebo::common::FileLogger::~FileLogger () [virtual]

Destructor.

10.80.3 Member Function Documentation

10.80.3.1 void gazebo::common::FileLogger::Init (const std::string & _filename)

Initialize the file logger.

Parameters

in	_filename	Name and path of the log file to write output into.
----	-----------	---

10.80.3.2 virtual FileLogger& gazebo::common::FileLogger::operator()() [virtual]

Output a filename and line number, then return a reference to the logger.

Returns

Reference to this logger.

10.80.3.3 virtual FileLogger& gazebo::common::FileLogger::operator() (const std::string & *_file*, int *_line*) [virtual]

Output a filename and line number, then return a reference to the logger.

Parameters

in	<i>_file</i>	Filename to output.
in	<i>_line</i>	Line number in the <i>_file</i> .

Returns

Reference to this logger.

The documentation for this class was generated from the following file:

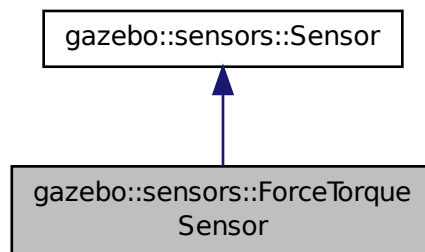
- **Console.hh**

10.81 gazebo::sensors::ForceTorqueSensor Class Reference

Sensor (p. 907) for measure force and torque on a joint.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::ForceTorqueSensor:

**Public Member Functions**

- **ForceTorqueSensor** ()
Constructor.
- virtual ~**ForceTorqueSensor** ()
Destructor.

- `template<typename T >`
event::ConnectionPtr ConnectUpdate (T _subscriber)
Connect a to the update signal.
- `void` **DisconnectUpdate** (**event::ConnectionPtr** &_conn)
Disconnect from the update signal.
- **math::Vector3 GetForce** () const
Get the current joint force.
- **physics::JointPtr GetJoint** () const
Get Parent Joint.
- `virtual std::string` **GetTopic** () const
Returns the topic name as set in SDF.
- **math::Vector3 GetTorque** () const
Get the current joint torque.
- `virtual void` **Init** ()
Initialize the sensor.
- `virtual bool` **IsActive** ()
Returns true if sensor generation is active.
- `virtual void` **Load** (const std::string &_worldName)
Load the sensor with default parameters.

Protected Member Functions

- `virtual void` **Fini** ()
Finalize the sensor.
- `virtual bool` **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Protected Attributes

- **event::EventT**< void(msgs::WrenchStamped)> **update**
Update event.

10.81.1 Detailed Description

Sensor (p. 907) for measure force and torque on a joint.

10.81.2 Constructor & Destructor Documentation

10.81.2.1 gazebo::sensors::ForceTorqueSensor::ForceTorqueSensor ()

Constructor.

10.81.2.2 `virtual gazebo::sensors::ForceTorqueSensor::~ForceTorqueSensor ()` [virtual]

Destructor.

10.81.3 Member Function Documentation

10.81.3.1 `template<typename T > event::ConnectionPtr gazebo::sensors::ForceTorqueSensor::ConnectUpdate (T _subscriber) [inline]`

Connect a to the update signal.

Parameters

in	<i>_subscriber</i>	Callback function.
----	--------------------	--------------------

Returns

The connection, which must be kept in scope.

10.81.3.2 `void gazebo::sensors::ForceTorqueSensor::DisconnectUpdate (event::ConnectionPtr & _conn) [inline]`

Disconnect from the update signal.

Parameters

in	<i>_conn</i>	Connection to remove.
----	--------------	-----------------------

10.81.3.3 `virtual void gazebo::sensors::ForceTorqueSensor::Fini () [protected],[virtual]`

Finalize the sensor.

Reimplemented from `gazebo::sensors::Sensor` (p. 912).

10.81.3.4 `math::Vector3 gazebo::sensors::ForceTorqueSensor::GetForce () const`

Get the current joint force.

Returns

The latested measured force.

10.81.3.5 `physics::JointPtr gazebo::sensors::ForceTorqueSensor::GetJoint () const`

Get Parent Joint.

Returns

Pointer to the joint containing this sensor

10.81.3.6 `virtual std::string gazebo::sensors::ForceTorqueSensor::GetTopic () const [virtual]`

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 914).

10.81.3.7 math::Vector3 gazebo::sensors::ForceTorqueSensor::GetTorque () const

Get the current joint torque.

Returns

The latest measured torque.

10.81.3.8 virtual void gazebo::sensors::ForceTorqueSensor::Init () [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.81.3.9 virtual bool gazebo::sensors::ForceTorqueSensor::IsActive () [virtual]

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.81.3.10 virtual void gazebo::sensors::ForceTorqueSensor::Load (const std::string & _worldName) [virtual]

Load the sensor with default parameters.

Parameters

in	<code>_worldName</code>	Name of world to load from.
----	-------------------------	-----------------------------

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.81.3.11 virtual bool gazebo::sensors::ForceTorqueSensor::UpdateImpl (bool) [protected],[virtual]

This gets overwritten by derived sensor types.

This function is called during `Sensor::Update`.
And in turn, `Sensor::Update` is called by
`SensorManager::Update`

Parameters

in	<code>_force</code>	True if update is forced, false if not
----	---------------------	--

Returns

True if the sensor was updated.

Reimplemented from `gazebo::sensors::Sensor` (p. 917).

10.81.4 Member Data Documentation

10.81.4.1 `event::EventT<void(msgs::WrenchStamped)> gazebo::sensors::ForceTorqueSensor::update` [protected]

Update event.

The documentation for this class was generated from the following file:

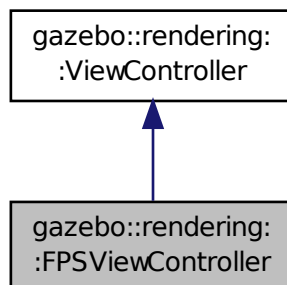
- `ForceTorqueSensor.hh`

10.82 gazebo::rendering::FPSViewController Class Reference

First Person Shooter style view controller.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for `gazebo::rendering::FPSViewController`:

**Public Member Functions**

- **FPSViewController** (`UserCameraPtr _camera`)
Constructor.
- virtual **~FPSViewController** ()
Destructor.
- void **HandleKeyPressEvent** (`const std::string &_key`)
Handle a key press event.
- void **HandleKeyReleaseEvent** (`const std::string &_key`)
Handle a key release event.

- virtual void **HandleMouseEvent** (const **common::MouseEvent** &_event)
Handle a mouse event.
- virtual void **Init** ()
Initialize the controller.
- virtual void **Update** ()
Update the camera position.

Static Public Member Functions

- static std::string **GetTypeString** ()
Get the type name of this view controller.

Additional Inherited Members

10.82.1 Detailed Description

First Person Shooter style view controller.

10.82.2 Constructor & Destructor Documentation

10.82.2.1 gazebo::rendering::FPSViewController::FPSViewController (**UserCameraPtr** _camera)

Constructor.

Parameters

in	Camera (p. 197)	to control
----	------------------------	------------

10.82.2.2 virtual gazebo::rendering::FPSViewController::~~FPSViewController () [virtual]

Destructor.

10.82.3 Member Function Documentation

10.82.3.1 static std::string gazebo::rendering::FPSViewController::GetTypeString () [static]

Get the type name of this view controller.

Returns

The name of the controller type: "fps"

10.82.3.2 void gazebo::rendering::FPSViewController::HandleKeyPressEvent (const std::string &_key) [virtual]

Handle a key press event.

Parameters

in	<code>_key</code>	The key that was pressed.
----	-------------------	---------------------------

Implements **gazebo::rendering::ViewController** (p. 1194).

10.82.3.3 `void gazebo::rendering::FPSViewController::HandleKeyReleaseEvent (const std::string & _key) [virtual]`

Handle a key release event.

Parameters

in	<code>_key</code>	The key that was released.
----	-------------------	----------------------------

Implements **gazebo::rendering::ViewController** (p. 1194).

10.82.3.4 `virtual void gazebo::rendering::FPSViewController::HandleMouseEvent (const common::MouseEvent & _event) [virtual]`

Handle a mouse event.

Parameters

in	<code>_event</code>	The mouse position.
----	---------------------	---------------------

Implements **gazebo::rendering::ViewController** (p. 1195).

10.82.3.5 `virtual void gazebo::rendering::FPSViewController::Init () [virtual]`

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 1195).

10.82.3.6 `virtual void gazebo::rendering::FPSViewController::Update () [virtual]`

Update the camera position.

Implements **gazebo::rendering::ViewController** (p. 1195).

The documentation for this class was generated from the following file:

- **FPSViewController.hh**

10.83 gazebo::physics::FrictionPyramid Class Reference

Parameters used for friction pyramid model.

```
#include <physics/physics.hh>
```

Public Member Functions

- **FrictionPyramid ()**

Constructor.

- virtual `~FrictionPyramid ()`

Destructor.

- double `GetMuPrimary ()`

Get the friction coefficient in the primary direction.

- double `GetMuSecondary ()`

Get the friction coefficient in the secondary direction.

- void `SetMuPrimary (double _mu)`

Set the friction coefficient in the primary direction.

- void `SetMuSecondary (double _mu)`

Set the friction coefficient in the secondary direction.

Public Attributes

- `math::Vector3 direction1`

Vector for specifying the primary friction direction, relative to the parent collision frame.

10.83.1 Detailed Description

Parameters used for friction pyramid model.

10.83.2 Constructor & Destructor Documentation

10.83.2.1 gazebo::physics::FrictionPyramid::FrictionPyramid ()

Constructor.

10.83.2.2 virtual gazebo::physics::FrictionPyramid::~FrictionPyramid () [virtual]

Destructor.

10.83.3 Member Function Documentation

10.83.3.1 double gazebo::physics::FrictionPyramid::GetMuPrimary ()

Get the friction coefficient in the primary direction.

Returns

Friction coefficient in primary direction.

10.83.3.2 double gazebo::physics::FrictionPyramid::GetMuSecondary ()

Get the friction coefficient in the secondary direction.

Returns

Friction coefficient in secondary direction.

10.83.3.3 void gazebo::physics::FrictionPyramid::SetMuPrimary (double *_mu*)

Set the friction coefficient in the primary direction.

Parameters

in	<i>_mu</i>	Friction coefficient.
----	------------	-----------------------

10.83.3.4 void gazebo::physics::FrictionPyramid::SetMuSecondary (double *_mu*)

Set the friction coefficient in the secondary direction.

Parameters

in	<i>_mu</i>	Friction coefficient.
----	------------	-----------------------

10.83.4 Member Data Documentation

10.83.4.1 math::Vector3 gazebo::physics::FrictionPyramid::direction1

Vector for specifying the primary friction direction, relative to the parent collision frame.

The component of this vector that is orthogonal to the surface normal will be set as the primary friction direction. If undefined, a vector constrained to be perpendicular to the contact normal in the global y-z plane is used.

See Also

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

The documentation for this class was generated from the following file:

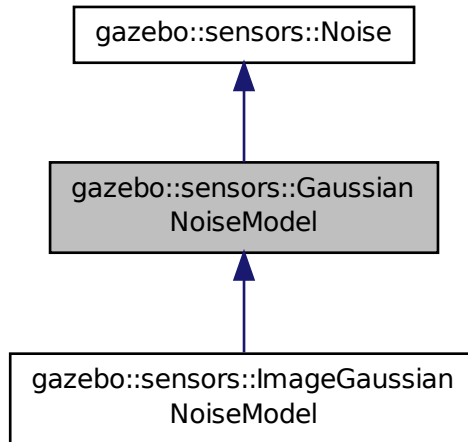
- **SurfaceParams.hh**

10.84 gazebo::sensors::GaussianNoiseModel Class Reference

Gaussian noise class.

```
#include <GaussianNoiseModel.hh>
```

Inheritance diagram for gazebo::sensors::GaussianNoiseModel:



Public Member Functions

- **GaussianNoiseModel** ()
Constructor.
- virtual **~GaussianNoiseModel** ()
Destructor.
- double **ApplyImpl** (double _in)
Apply noise to input data value.
- virtual void **Fini** ()
Finalize the noise model.
- double **GetBias** () const
Accessor for bias.
- double **GetMean** () const
Accessor for mean.
- double **GetStdDev** () const
Accessor for stddev.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load noise parameters from sdf.

Protected Attributes

- double **bias**
If type starts with GAUSSIAN, the bias we'll add.
- double **mean**
If type starts with GAUSSIAN, the mean of the distribution from which we sample when adding noise.

- double **precision**
If `type==GAUSSIAN_QUANTIZED`, the precision to which the output signal is rounded.
- bool **quantized**
True if the type is `GAUSSIAN_QUANTIZED`.
- double **stdDev**
If type starts with `GAUSSIAN`, the standard deviation of the distribution from which we sample when adding noise.

Additional Inherited Members

10.84.1 Detailed Description

Gaussian noise class.

Gaussian noise class for image sensors.

10.84.2 Constructor & Destructor Documentation

10.84.2.1 gazebo::sensors::GaussianNoiseModel::GaussianNoiseModel ()

Constructor.

10.84.2.2 virtual gazebo::sensors::GaussianNoiseModel::~~GaussianNoiseModel () [virtual]

Destructor.

10.84.3 Member Function Documentation

10.84.3.1 double gazebo::sensors::GaussianNoiseModel::ApplyImpl (double *_in*) [virtual]

Apply noise to input data value.

This gets overridden by derived classes, and called by Apply.

Parameters

<code>in</code>	<code>_in</code>	Input data value.
-----------------	------------------	-------------------

Returns

Data with noise applied.

Reimplemented from [gazebo::sensors::Noise](#) (p. 750).

10.84.3.2 virtual void gazebo::sensors::GaussianNoiseModel::Fini () [virtual]

Finalize the noise model.

Reimplemented from [gazebo::sensors::Noise](#) (p. 750).

Reimplemented in [gazebo::sensors::ImageGaussianNoiseModel](#) (p. 522).

10.84.3.3 `double gazebo::sensors::GaussianNoiseModel::GetBias () const`

Accessor for bias.

Returns

Bias on output.

10.84.3.4 `double gazebo::sensors::GaussianNoiseModel::GetMean () const`

Accessor for mean.

Returns

Mean of Gaussian noise.

10.84.3.5 `double gazebo::sensors::GaussianNoiseModel::GetStdDev () const`

Accessor for stddev.

Returns

Standard deviation of Gaussian noise.

10.84.3.6 `virtual void gazebo::sensors::GaussianNoiseModel::Load (sdf::ElementPtr _sdf) [virtual]`

Load noise parameters from sdf.

Parameters

<code>in</code>	<code>_sdf</code>	SDF parameters.
<code>in</code>	<code>_sensor</code>	Type of sensor.

Reimplemented from `gazebo::sensors::Noise` (p. 750).

Reimplemented in `gazebo::sensors::ImageGaussianNoiseModel` (p. 522).

10.84.4 Member Data Documentation

10.84.4.1 `double gazebo::sensors::GaussianNoiseModel::bias [protected]`

If type starts with GAUSSIAN, the bias we'll add.

10.84.4.2 `double gazebo::sensors::GaussianNoiseModel::mean [protected]`

If type starts with GAUSSIAN, the mean of the distribution from which we sample when adding noise.

10.84.4.3 double gazebo::sensors::GaussianNoiseModel::precision [protected]

If type==GAUSSIAN_QUANTIZED, the precision to which the output signal is rounded.

10.84.4.4 bool gazebo::sensors::GaussianNoiseModel::quantized [protected]

True if the type is GAUSSIAN_QUANTIZED.

10.84.4.5 double gazebo::sensors::GaussianNoiseModel::stdDev [protected]

If type starts with GAUSSIAN, the standard deviation of the distribution from which we sample when adding noise.

The documentation for this class was generated from the following file:

- **GaussianNoiseModel.hh**

10.85 google::protobuf::compiler::cpp::GazeboGenerator Class Reference

Google protobuf message generator for **gazebo::msgs** (p. 113).

```
#include <GazeboGenerator.hh>
```

Public Member Functions

- **GazeboGenerator** (const std::string &_name)
- virtual ~**GazeboGenerator** ()
- virtual bool **Generate** (const FileDescriptor *file, const string ¶meter, OutputDirectory *directory, string *error) const

10.85.1 Detailed Description

Google protobuf message generator for **gazebo::msgs** (p. 113).

10.85.2 Constructor & Destructor Documentation

10.85.2.1 google::protobuf::compiler::cpp::GazeboGenerator::GazeboGenerator (const std::string &_name)

10.85.2.2 virtual google::protobuf::compiler::cpp::GazeboGenerator::~~GazeboGenerator () [virtual]

10.85.3 Member Function Documentation

10.85.3.1 virtual bool google::protobuf::compiler::cpp::GazeboGenerator::Generate (const FileDescriptor * file, const string & parameter, OutputDirectory * directory, string * error) const [virtual]

The documentation for this class was generated from the following file:

- **GazeboGenerator.hh**

10.86 gazebo::physics::GearboxJoint< T > Class Template Reference

A double axis gearbox joint.

```
#include <physics/physics.hh>
```

Public Member Functions

- **GearboxJoint** (**BasePtr** _parent)
Constructor.
- virtual **~GearboxJoint** ()
Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual double **GetGearboxRatio** () const
Get gearbox joint gear ratio.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load joint.
- virtual void **SetGearboxRatio** (double _gearRatio)=0
Set gearbox joint gear ratio.

Protected Member Functions

- virtual void **Init** ()
Initialize joint.

Protected Attributes

- double **gearRatio**
Gearbox gearRatio.
- std::string **referenceBody**
reference link/body for computing joint angles

10.86.1 Detailed Description

```
template<class T>class gazebo::physics::GearboxJoint< T >
```

A double axis gearbox joint.

10.86.2 Constructor & Destructor Documentation

10.86.2.1 `template<class T > gazebo::physics::GearboxJoint< T >::GearboxJoint (BasePtr _parent) [inline]`

Constructor.

Parameters

in	<code>_parent</code>	Parent link
----	----------------------	-------------

References gazebo::physics::Base::GEARBOX_JOINT.

10.86.2.2 `template<class T> virtual gazebo::physics::GearboxJoint< T >::~~GearboxJoint () [inline], [virtual]`

Destructor.

10.86.3 Member Function Documentation

10.86.3.1 `template<class T> virtual unsigned int gazebo::physics::GearboxJoint< T >::GetAngleCount () const [inline], [virtual]`

10.86.3.2 `template<class T> virtual double gazebo::physics::GearboxJoint< T >::GetGearboxRatio () const [inline], [virtual]`

Get gearbox joint gear ratio.

Returns

Gear ratio value.

10.86.3.3 `template<class T> virtual void gazebo::physics::GearboxJoint< T >::Init () [inline], [protected], [virtual]`

Initialize joint.

References gazebo::msgs::Init().

10.86.3.4 `template<class T> virtual void gazebo::physics::GearboxJoint< T >::Load (sdf::ElementPtr _sdf) [inline], [virtual]`

Load joint.

Parameters

<code>in</code>	<code>_sdf</code>	Pointer to SDF element
-----------------	-------------------	------------------------

References gzerr.

10.86.3.5 `template<class T> virtual void gazebo::physics::GearboxJoint< T >::SetGearboxRatio (double _gearRatio) [pure virtual]`

Set gearbox joint gear ratio.

This must be implemented in a child class

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_gearRatio</code>	Gear ratio value.

10.86.4 Member Data Documentation

10.86.4.1 `template<class T> double gazebo::physics::GearboxJoint< T >::gearRatio` [protected]

Gearbox gearRatio.

10.86.4.2 `template<class T> std::string gazebo::physics::GearboxJoint< T >::referenceBody` [protected]

reference link/body for computing joint angles

The documentation for this class was generated from the following file:

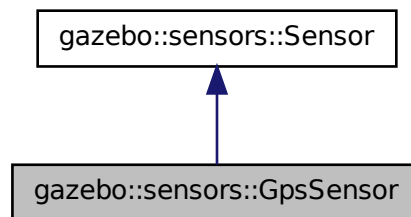
- **GearboxJoint.hh**

10.87 gazebo::sensors::GpsSensor Class Reference

GpsSensor (p. 464) to provide position measurement.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::GpsSensor:



Public Member Functions

- **GpsSensor** ()
Constructor.
- virtual **~GpsSensor** ()
Destructor.
- virtual void **Fini** ()
Finalize the sensor.
- double **GetAltitude** () const
Accessor for current altitude.
- **math::Angle GetLatitude** () const
Accessor for current latitude angle.
- **math::Angle GetLongitude** () const

Accessor for current longitude angle.

- virtual void **Init** ()
Initialize the sensor.
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)
Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.

Protected Member Functions

- virtual bool **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.87.1 Detailed Description

GpsSensor (p. 464) to provide position measurement.

10.87.2 Constructor & Destructor Documentation

10.87.2.1 gazebo::sensors::GpsSensor::GpsSensor ()

Constructor.

10.87.2.2 virtual gazebo::sensors::GpsSensor::~~GpsSensor () [virtual]

Destructor.

10.87.3 Member Function Documentation

10.87.3.1 virtual void gazebo::sensors::GpsSensor::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 912).

10.87.3.2 double gazebo::sensors::GpsSensor::GetAltitude () const

Accessor for current altitude.

Returns

Current altitude above sea level.

10.87.3.3 `math::Angle gazebo::sensors::GpsSensor::GetLatitude () const`

Accessor for current latitude angle.

Returns

Current latitude angle.

10.87.3.4 `math::Angle gazebo::sensors::GpsSensor::GetLongitude () const`

Accessor for current longitude angle.

Returns

Current longitude angle.

10.87.3.5 `virtual void gazebo::sensors::GpsSensor::Init () [virtual]`

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.87.3.6 `virtual void gazebo::sensors::GpsSensor::Load (const std::string & _worldName, sdf::ElementPtr _sdf) [virtual]`

Load the sensor with SDF parameters.

Parameters

in	<code>_sdf</code>	SDF Sensor (p. 907) parameters.
in	<code>_worldName</code>	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.87.3.7 `virtual void gazebo::sensors::GpsSensor::Load (const std::string & _worldName) [virtual]`

Load the sensor with default parameters.

Parameters

in	<code>_worldName</code>	Name of world to load from.
----	-------------------------	-----------------------------

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.87.3.8 `virtual bool gazebo::sensors::GpsSensor::UpdateImpl (bool) [protected],[virtual]`

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	<code>_force</code>	True if update is forced, false if not
----	---------------------	--

Returns

True if the sensor was updated.

Reimplemented from `gazebo::sensors::Sensor` (p. 917).

The documentation for this class was generated from the following file:

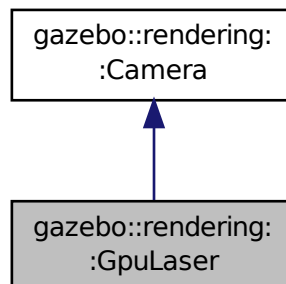
- `GpsSensor.hh`

10.88 gazebo::rendering::GpuLaser Class Reference

GPU based laser distance sensor.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::GpuLaser:



Public Member Functions

- **GpuLaser** (const std::string &_namePrefix, **ScenePtr** _scene, bool _autoRender=true)
Constructor.
- virtual **~GpuLaser** ()
Destructor.
- template<typename T >
event::ConnectionPtr ConnectNewLaserFrame (T _subscriber)
Connect to a laser frame signal.
- void **CreateLaserTexture** (const std::string &_textureName)
Create the texture which is used to render laser data.
- void **DisconnectNewLaserFrame** (**event::ConnectionPtr** &_c)

- Disconnect from a laser frame signal.*

 - virtual void **Fini** ()

Finalize the camera.
- double **GetCameraCount** () const
- Get the number of cameras required.*
- double **GetCosHorzFOV** () const
- Get Cos Horz field-of-view.*
- double **GetCosVertFOV** () const
- Get Cos Vert field-of-view.*
- double **GetFarClip** () const
- Get far clip.*
- double **GetHorzFOV** () const
- Get the horizontal field of view of the laser sensor.*
- double **GetHorzHalfAngle** () const
- Get (horizontal_max_angle + horizontal_min_angle) * 0.5.*
- const float * **GetLaserData** ()
- All things needed to get back z buffer for laser data.*
- double **GetNearClip** () const
- Get near clip.*
- double **GetRayCountRatio** () const
- Get the ray count ratio (equivalent to aspect ratio)*
- double **GetVertFOV** () const
- Get the vertical field-of-view.*
- double **GetVertHalfAngle** () const
- Get (vertical_max_angle + vertical_min_angle) * 0.5.*
- virtual void **Init** ()
- Initialize the camera.*
- bool **IsHorizontal** () const
- Gets if sensor is horizontal.*
- virtual void **Load** (sdf::ElementPtr _sdf)
- Load the camera with a set of parameters.*
- virtual void **Load** ()
- Load the camera with default parameters.*
- virtual void **notifyRenderSingleObject** (Ogre::Renderable *_rend, const Ogre::Pass *_p, const Ogre::AutoParamDataSource *_s, const Ogre::LightList *_ll, bool _supp)
- virtual void **PostRender** ()
- Post render.*
- void **SetCameraCount** (double _cameraCount)
- Set the number of cameras required.*
- void **SetCosHorzFOV** (double _chfov)
- Set the Cos Horz FOV.*
- void **SetCosVertFOV** (double _cvfov)
- Set the Cos Horz FOV.*
- void **SetFarClip** (double _far)
- Set the far clip distance.*
- void **SetHorzFOV** (double _hfov)
- Set the horizontal fov.*

- void **SetHorzHalfAngle** (double _angle)
Set the horizontal half angle.
- void **SetIsHorizontal** (bool _horizontal)
Set sensor horizontal or vertical.
- void **SetNearClip** (double _near)
Set the near clip distance.
- void **SetRangeCount** (unsigned int _w, unsigned int _h=1)
Set the number of laser samples in the width and height.
- void **SetRayCountRatio** (double _rayCountRatio)
Sets the ray count ratio (equivalen to aspect ratio)
- void **SetVertFOV** (double _vfov)
Set the vertical fov.
- void **SetVertHalfAngle** (double _angle)
Set the vertical half angle.

Protected Attributes

- unsigned int **cameraCount**
Number of cameras needed to generate the rays.
- double **chfov**
Cos horizontal field-of-view.
- double **cvfov**
Cos vertical field-of-view.
- double **far**
Far clip plane.
- double **hfov**
Horizontal field-of-view.
- double **horzHalfAngle**
Horizontal half angle.
- bool **isHorizontal**
True if the sensor is horizontal only.
- double **near**
Near clip plane.
- double **rayCountRatio**
Ray count ratio.
- double **vertHalfAngle**
Vertical half angle.
- double **vfov**
Vertical field-of-view.

Additional Inherited Members

10.88.1 Detailed Description

GPU based laser distance sensor.

10.88.2 Constructor & Destructor Documentation

10.88.2.1 `gazebo::rendering::GpuLaser::GpuLaser (const std::string & _namePrefix, ScenePtr _scene, bool _autoRender = true)`

Constructor.

Parameters

in	<i>_namePrefix</i>	Unique prefix name for the camera.
in	<i>_scene</i>	Scene (p. 879) that will contain the camera
in	<i>_autoRender</i>	Almost everyone should leave this as true.

10.88.2.2 `virtual gazebo::rendering::GpuLaser::~GpuLaser () [virtual]`

Destructor.

10.88.3 Member Function Documentation

10.88.3.1 `template<typename T > event::ConnectionPtr gazebo::rendering::GpuLaser::ConnectNewLaserFrame (T _subscriber) [inline]`

Connect to a laser frame signal.

Parameters

in	<i>_subscriber</i>	Callback that is called when a new image is generated
----	--------------------	---

Returns

A pointer to the connection. This must be kept in scope.

10.88.3.2 `void gazebo::rendering::GpuLaser::CreateLaserTexture (const std::string & _textureName)`

Create the texture which is used to render laser data.

Parameters

in	<i>_textureName</i>	Name of the new texture.
----	---------------------	--------------------------

10.88.3.3 `void gazebo::rendering::GpuLaser::DisconnectNewLaserFrame (event::ConnectionPtr & _c) [inline]`

Disconnect from a laser frame signal.

Parameters

in	<i>_c</i>	The connection to disconnect
----	-----------	------------------------------

10.88.3.4 virtual void gazebo::rendering::GpuLaser::Fini () [virtual]

Finalize the camera.

This function is called before the camera is destructed

Reimplemented from **gazebo::rendering::Camera** (p. 207).

10.88.3.5 double gazebo::rendering::GpuLaser::GetCameraCount () const

Get the number of cameras required.

Returns

Number of cameras needed to generate the rays

10.88.3.6 double gazebo::rendering::GpuLaser::GetCosHorzFOV () const

Get Cos Horz field-of-view.

Returns

$2 * \text{atan}(\tan(\text{this->hfov}/2) / \cos(\text{this->vfov}/2))$

10.88.3.7 double gazebo::rendering::GpuLaser::GetCosVertFOV () const

Get Cos Vert field-of-view.

Returns

$2 * \text{atan}(\tan(\text{this->vfov}/2) / \cos(\text{this->hfov}/2))$

10.88.3.8 double gazebo::rendering::GpuLaser::GetFarClip () const

Get far clip.

Returns

far clip distance

10.88.3.9 double gazebo::rendering::GpuLaser::GetHorzFOV () const

Get the horizontal field of view of the laser sensor.

Returns

The horizontal field of view of the laser sensor.

10.88.3.10 `double gazebo::rendering::GpuLaser::GetHorzHalfAngle () const`

Get $(\text{horizontal_max_angle} + \text{horizontal_min_angle}) * 0.5$.

Returns

$(\text{horizontal_max_angle} + \text{horizontal_min_angle}) * 0.5$

10.88.3.11 `const float* gazebo::rendering::GpuLaser::GetLaserData ()`

All things needed to get back z buffer for laser data.

Returns

Array of laser data.

10.88.3.12 `double gazebo::rendering::GpuLaser::GetNearClip () const`

Get near clip.

Returns

near clip distance

10.88.3.13 `double gazebo::rendering::GpuLaser::GetRayCountRatio () const`

Get the ray count ratio (equivalent to aspect ratio)

Returns

The ray count ratio (equivalent to aspect ratio)

10.88.3.14 `double gazebo::rendering::GpuLaser::GetVertFOV () const`

Get the vertical field-of-view.

Returns

The vertical field of view of the laser sensor.

10.88.3.15 `double gazebo::rendering::GpuLaser::GetVertHalfAngle () const`

Get $(\text{vertical_max_angle} + \text{vertical_min_angle}) * 0.5$.

Returns

$(\text{vertical_max_angle} + \text{vertical_min_angle}) * 0.5$

10.88.3.16 virtual void gazebo::rendering::GpuLaser::Init () [virtual]

Initialize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 214).

10.88.3.17 bool gazebo::rendering::GpuLaser::IsHorizontal () const

Gets if sensor is horizontal.

Returns

True if horizontal, false if not

10.88.3.18 virtual void gazebo::rendering::GpuLaser::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the camera with a set of parameters.

Parameters

in	<i>_sdf</i>	The SDF camera info
----	-------------	---------------------

Reimplemented from **gazebo::rendering::Camera** (p. 215).

10.88.3.19 virtual void gazebo::rendering::GpuLaser::Load () [virtual]

Load the camera with default parameters.

Reimplemented from **gazebo::rendering::Camera** (p. 215).

10.88.3.20 virtual void gazebo::rendering::GpuLaser::notifyRenderSingleObject (Ogre::Renderable * *_rend*, const Ogre::Pass * *_p*, const Ogre::AutoParamDataSource * *_s*, const Ogre::LightList * *_ll*, bool *_supp*) [virtual]

10.88.3.21 virtual void gazebo::rendering::GpuLaser::PostRender () [virtual]

Post render.

Called after the render signal.

Reimplemented from **gazebo::rendering::Camera** (p. 216).

10.88.3.22 void gazebo::rendering::GpuLaser::SetCameraCount (double *_cameraCount*)

Set the number of cameras required.

Parameters

in	<i>_cameraCount</i>	The number of cameras required to generate the rays
----	---------------------	---

10.88.3.23 void gazebo::rendering::GpuLaser::SetCosHorzFOV (double *_chfov*)

Set the Cos Horz FOV.

Parameters

in	<i>_chfov</i>	Cos Horz FOV
----	---------------	--------------

10.88.3.24 void gazebo::rendering::GpuLaser::SetCosVertFOV (double *_cvfov*)

Set the Cos Horz FOV.

Parameters

in	<i>_cvfov</i>	Cos Horz FOV
----	---------------	--------------

10.88.3.25 void gazebo::rendering::GpuLaser::SetFarClip (double *_far*)

Set the far clip distance.

Parameters

in	<i>_far</i>	far clip distance
----	-------------	-------------------

10.88.3.26 void gazebo::rendering::GpuLaser::SetHorzFOV (double *_hfov*)

Set the horizontal fov.

Parameters

in	<i>_hfov</i>	horizontal fov
----	--------------	----------------

10.88.3.27 void gazebo::rendering::GpuLaser::SetHorzHalfAngle (double *_angle*)

Set the horizontal half angle.

Parameters

in	<i>_angle</i>	horizontal half angle
----	---------------	-----------------------

10.88.3.28 void gazebo::rendering::GpuLaser::SetIsHorizontal (bool *_horizontal*)

Set sensor horizontal or vertical.

Parameters

in	<i>_horizontal</i>	True if horizontal, false if not
----	--------------------	----------------------------------

10.88.3.29 void gazebo::rendering::GpuLaser::SetNearClip (double *_near*)

Set the near clip distance.

Parameters

in	<i>_near</i>	near clip distance
----	--------------	--------------------

10.88.3.30 void gazebo::rendering::GpuLaser::SetRangeCount (unsigned int *_w*, unsigned int *_h* = 1)

Set the number of laser samples in the width and height.

Parameters

in	<i>_w</i>	Number of samples in the horizontal sweep
in	<i>_h</i>	Number of samples in the vertical sweep

10.88.3.31 void gazebo::rendering::GpuLaser::SetRayCountRatio (double *_rayCountRatio*)

Sets the ray count ratio (equivalent to aspect ratio)

Parameters

in	<i>_rayCountRatio</i>	ray count ratio (equivalent to aspect ratio)
----	-----------------------	--

10.88.3.32 void gazebo::rendering::GpuLaser::SetVertFOV (double *_vfov*)

Set the vertical fov.

Parameters

in	<i>_vfov</i>	vertical fov
----	--------------	--------------

10.88.3.33 void gazebo::rendering::GpuLaser::SetVertHalfAngle (double *_angle*)

Set the vertical half angle.

Parameters

in	<i>_angle</i>	vertical half angle
----	---------------	---------------------

10.88.4 Member Data Documentation

10.88.4.1 unsigned int gazebo::rendering::GpuLaser::cameraCount [protected]

Number of cameras needed to generate the rays.

10.88.4.2 `double gazebo::rendering::GpuLaser::chfov` [protected]

Cos horizontal field-of-view.

10.88.4.3 `double gazebo::rendering::GpuLaser::cvfov` [protected]

Cos vertical field-of-view.

10.88.4.4 `double gazebo::rendering::GpuLaser::far` [protected]

Far clip plane.

10.88.4.5 `double gazebo::rendering::GpuLaser::hfov` [protected]

Horizontal field-of-view.

10.88.4.6 `double gazebo::rendering::GpuLaser::horzHalfAngle` [protected]

Horizontal half angle.

10.88.4.7 `bool gazebo::rendering::GpuLaser::isHorizontal` [protected]

True if the sensor is horizontal only.

10.88.4.8 `double gazebo::rendering::GpuLaser::near` [protected]

Near clip plane.

10.88.4.9 `double gazebo::rendering::GpuLaser::rayCountRatio` [protected]

Ray count ratio.

10.88.4.10 `double gazebo::rendering::GpuLaser::vertHalfAngle` [protected]

Vertical half angle.

10.88.4.11 `double gazebo::rendering::GpuLaser::vfov` [protected]

Vertical field-of-view.

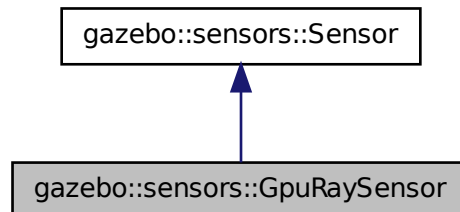
The documentation for this class was generated from the following file:

- **GpuLaser.hh**

10.89 gazebo::sensors::GpuRaySensor Class Reference

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::GpuRaySensor:



Public Member Functions

- **GpuRaySensor** ()
Constructor.
- virtual **~GpuRaySensor** ()
Destructor.
- **event::ConnectionPtr ConnectNewLaserFrame** (boost::function< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)> _subscriber)
Connect to the new laser frame event.
- void **DisconnectNewLaserFrame** (event::ConnectionPtr &_conn)
Disconnect Laser Frame.
- **math::Angle GetAngleMax** () const
Get the maximum angle.
- **math::Angle GetAngleMin** () const
Get the minimum angle.
- double **GetAngleResolution** () const
Get radians between each range.
- unsigned int **GetCameraCount** () const
Gets the camera count.
- double **GetCosHorzFOV** () const
Get Cos Horz field-of-view.
- double **GetCosVertFOV** () const
Get Cos Vert field-of-view.
- int **GetFiducial** (int _index) const
Get detected fiducial value for a ray.
- double **GetHorzFOV** () const
Get the horizontal field of view of the laser sensor.
- double **GetHorzHalfAngle** () const

*Get $(horizontal_max_angle + horizontal_min_angle) * 0.5$.*

- **rendering::GpuLaserPtr GetLaserCamera () const**
*Returns a pointer to the internally kept **rendering::GpuLaser** (p. 467).*
- double **GetRange** (int _index)
Get detected range for a ray.
- int **GetRangeCount** () const
Get the range count.
- double **GetRangeCountRatio** () const
Return the ratio of horizontal range count to vertical range count.
- double **GetRangeMax** () const
Get the maximum range.
- double **GetRangeMin** () const
Get the minimum range.
- double **GetRangeResolution** () const
Get the range resolution If RangeResolution is 1, the number of simulated rays is equal to the number of returned range readings.
- void **GetRanges** (std::vector< double > &_ranges)
Get all the ranges.
- int **GetRayCount** () const
Get the ray count.
- double **GetRayCountRatio** () const
Return the ratio of horizontal ray count to vertical ray count.
- double **GetRetro** (int _index) const
Get detected retro (intensity) value for a ray.
- virtual std::string **GetTopic** () const
Returns the topic name as set in SDF.
- double **GetVertFOV** () const
Get the vertical field-of-view.
- double **GetVertHalfAngle** () const
*Get $(vertical_max_angle + vertical_min_angle) * 0.5$.*
- **math::Angle GetVerticalAngleMax** () const
Get the vertical scan line top angle.
- **math::Angle GetVerticalAngleMin** () const
Get the vertical scan bottom angle.
- double **GetVerticalAngleResolution** () const
Get the vertical angle in radians between each range.
- int **GetVerticalRangeCount** () const
Get the vertical scan line count.
- int **GetVerticalRayCount** () const
Get the vertical scan line count.
- virtual void **Init** ()
Initialize the ray.
- virtual bool **IsActive** ()
Returns true if sensor generation is active.
- bool **IsHorizontal** () const
Gets if sensor is horizontal.
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)

- Load the sensor with SDF parameters.*

 - virtual void **Load** (const std::string &_worldName)

Load the sensor with default parameters.
- void **SetAngleMax** (double _angle)

Set the scan maximum angle.
- void **SetAngleMin** (double _angle)

Set the scan minimum angle.
- void **SetVerticalAngleMax** (double _angle)

Set the vertical scan line top angle.
- void **SetVerticalAngleMin** (double _angle)

Set the vertical scan bottom angle.

Protected Member Functions

- virtual void **Fini** ()

Finalize the ray.
- virtual bool **UpdateImpl** (bool _force)

This gets overwritten by derived sensor types.

Protected Attributes

- sdf::ElementPtr **cameraElem**
- Camera SDF element.*
- sdf::ElementPtr **horzElem**
- Horizontal SDF element.*
- unsigned int **horzRangeCount**
- Horizontal range count.*
- unsigned int **horzRayCount**
- Horizontal ray count.*
- double **rangeCountRatio**
- Range count ratio.*
- sdf::ElementPtr **rangeElem**
- Range SDF element.*
- sdf::ElementPtr **scanElem**
- Scan SDF element.*
- sdf::ElementPtr **vertElem**
- Vertical SDF element.*
- unsigned int **vertRangeCount**
- Vertical range count.*
- unsigned int **vertRayCount**
- Vertical ray count.*

10.89.1 Constructor & Destructor Documentation

10.89.1.1 gazebo::sensors::GpuRaySensor::GpuRaySensor ()

Constructor.

10.89.1.2 `virtual gazebo::sensors::GpuRaySensor::~~GpuRaySensor () [virtual]`

Destructor.

10.89.2 Member Function Documentation

10.89.2.1 `event::ConnectionPtr gazebo::sensors::GpuRaySensor::ConnectNewLaserFrame (boost::function< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)> _subscriber)`

Connect to the new laser frame event.

Parameters

<code>in</code>	<code>_subscriber</code>	Event callback.
-----------------	--------------------------	-----------------

10.89.2.2 `void gazebo::sensors::GpuRaySensor::DisconnectNewLaserFrame (event::ConnectionPtr & _conn)`

Disconnect Laser Frame.

Parameters

<code>in, out</code>	<code>_conn</code>	Connection pointer to disconnect.
----------------------	--------------------	-----------------------------------

10.89.2.3 `virtual void gazebo::sensors::GpuRaySensor::Fini () [protected],[virtual]`

Finalize the ray.

Reimplemented from `gazebo::sensors::Sensor` (p. 912).

10.89.2.4 `math::Angle gazebo::sensors::GpuRaySensor::GetAngleMax () const`

Get the maximum angle.

Returns

the maximum angle

10.89.2.5 `math::Angle gazebo::sensors::GpuRaySensor::GetAngleMin () const`

Get the minimum angle.

Returns

The minimum angle

10.89.2.6 `double gazebo::sensors::GpuRaySensor::GetAngleResolution () const`

Get radians between each range.

10.89.2.7 unsigned int gazebo::sensors::GpuRaySensor::GetCameraCount () const

Gets the camera count.

Returns

Number of cameras

10.89.2.8 double gazebo::sensors::GpuRaySensor::GetCosHorzFOV () const

Get Cos Horz field-of-view.

Returns

$2 * \text{atan}(\tan(\text{this->hfov}/2) / \cos(\text{this->vfov}/2))$

10.89.2.9 double gazebo::sensors::GpuRaySensor::GetCosVertFOV () const

Get Cos Vert field-of-view.

Returns

$2 * \text{atan}(\tan(\text{this->vfov}/2) / \cos(\text{this->hfov}/2))$

10.89.2.10 int gazebo::sensors::GpuRaySensor::GetFiducial (int *_index*) const

Get detected fiducial value for a ray.

Warning: If you are accessing all the ray data in a loop it's possible that the Ray will update in the middle of your access loop. This means some data will come from one scan, and some from another scan. You can solve this problem by using SetActive(false) <your accessor loop> SetActive(true).

Parameters

in	<i>_index</i>	Index of specific ray
----	---------------	-----------------------

Returns

Fiducial value of ray

10.89.2.11 double gazebo::sensors::GpuRaySensor::GetHorzFOV () const

Get the horizontal field of view of the laser sensor.

Returns

The horizontal field of view of the laser sensor.

10.89.2.12 `double gazebo::sensors::GpuRaySensor::GetHorzHalfAngle () const`

Get $(\text{horizontal_max_angle} + \text{horizontal_min_angle}) * 0.5$.

Returns

$(\text{horizontal_max_angle} + \text{horizontal_min_angle}) * 0.5$

10.89.2.13 `rendering::GpuLaserPtr gazebo::sensors::GpuRaySensor::GetLaserCamera () const` `[inline]`

Returns a pointer to the internally kept `rendering::GpuLaser` (p. 467).

Returns

Pointer to `GpuLaser`

10.89.2.14 `double gazebo::sensors::GpuRaySensor::GetRange (int _index)`

Get detected range for a ray.

Warning: If you are accessing all the ray data in a loop it's possible that the Ray will update in the middle of your access loop. This means some data will come from one scan, and some from another scan. You can solve this problem by using `SetActive(false)` <your accessor loop> `SetActive(true)`.

Parameters

<code>in</code>	<code>_index</code>	Index of specific ray
-----------------	---------------------	-----------------------

Returns

Returns `DBL_MAX` for no detection.

10.89.2.15 `int gazebo::sensors::GpuRaySensor::GetRangeCount () const`

Get the range count.

Returns

The number of ranges

10.89.2.16 `double gazebo::sensors::GpuRaySensor::GetRangeCountRatio () const`

Return the ratio of horizontal range count to vertical range count.

A ray count is the number of simulated rays. Whereas a range count is the total number of data points returned. When range count != ray count, then values are interpolated between rays.

10.89.2.17 `double gazebo::sensors::GpuRaySensor::GetRangeMax () const`

Get the maximum range.

Returns

The maximum range

10.89.2.18 `double gazebo::sensors::GpuRaySensor::GetRangeMin () const`

Get the minimum range.

Returns

The minimum range

10.89.2.19 `double gazebo::sensors::GpuRaySensor::GetRangeResolution () const`

Get the range resolution If RangeResolution is 1, the number of simulated rays is equal to the number of returned range readings.

If it's less than 1, fewer simulated rays than actual returned range readings are used, the results are interpolated from two nearest neighbors, and vice versa.

Returns

The Range Resolution

10.89.2.20 `void gazebo::sensors::GpuRaySensor::GetRanges (std::vector< double > & _ranges)`

Get all the ranges.

Parameters

out	_range	A vector that will contain all the range data
-----	--------	---

10.89.2.21 `int gazebo::sensors::GpuRaySensor::GetRayCount () const`

Get the ray count.

Returns

The number of rays

10.89.2.22 `double gazebo::sensors::GpuRaySensor::GetRayCountRatio () const`

Return the ratio of horizontal ray count to vertical ray count.

A ray count is the number of simulated rays. Whereas a range count is the total number of data points returned. When range count != ray count, then values are interpolated between rays.

10.89.2.23 `double gazebo::sensors::GpuRaySensor::GetRetro (int _index) const`

Get detected retro (intensity) value for a ray.

Warning: If you are accessing all the ray data in a loop it's possible that the Ray will update in the middle of your access loop. This means some data will come from one scan, and some from another scan. You can solve this problem by using `SetActive(false)` <your accessor loop> `SetActive(true)`.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of specific ray
-----------------	----------------------------	-----------------------

Returns

Intensity value of ray

10.89.2.24 `virtual std::string gazebo::sensors::GpuRaySensor::GetTopic () const` [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented from `gazebo::sensors::Sensor` (p.914).

10.89.2.25 `double gazebo::sensors::GpuRaySensor::GetVertFOV () const`

Get the vertical field-of-view.

10.89.2.26 `double gazebo::sensors::GpuRaySensor::GetVertHalfAngle () const`

Get $(\text{vertical_max_angle} + \text{vertical_min_angle}) * 0.5$.

Returns

$(\text{vertical_max_angle} + \text{vertical_min_angle}) * 0.5$

10.89.2.27 `math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMax () const`

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.89.2.28 `math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMin () const`

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.89.2.29 `double gazebo::sensors::GpuRaySensor::GetVerticalAngleResolution () const`

Get the vertical angle in radians between each range.

Returns

Resolution of the angle

10.89.2.30 `int gazebo::sensors::GpuRaySensor::GetVerticalRangeCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.89.2.31 `int gazebo::sensors::GpuRaySensor::GetVerticalRayCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.89.2.32 `virtual void gazebo::sensors::GpuRaySensor::Init () [virtual]`

Initialize the ray.

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.89.2.33 `virtual bool gazebo::sensors::GpuRaySensor::IsActive () [virtual]`

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.89.2.34 `bool gazebo::sensors::GpuRaySensor::IsHorizontal () const`

Gets if sensor is horizontal.

Returns

True if horizontal, false if not

10.89.2.35 `virtual void gazebo::sensors::GpuRaySensor::Load (const std::string & _worldName, sdf::ElementPtr _sdf) [virtual]`

Load the sensor with SDF parameters.

Parameters

in	<code>_sdf</code>	SDF Sensor (p. 907) parameters
in	<code>_worldName</code>	Name of world to load from

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.89.2.36 `virtual void gazebo::sensors::GpuRaySensor::Load (const std::string & _worldName) [virtual]`

Load the sensor with default parameters.

Parameters

in	<code>_worldName</code>	Name of world to load from
----	-------------------------	----------------------------

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.89.2.37 `void gazebo::sensors::GpuRaySensor::SetAngleMax (double _angle)`

Set the scan maximum angle.

Parameters

in	<code>_angle</code>	The maximum angle
----	---------------------	-------------------

10.89.2.38 `void gazebo::sensors::GpuRaySensor::SetAngleMin (double _angle)`

Set the scan minimum angle.

Parameters

in	<code>_angle</code>	The minimum angle
----	---------------------	-------------------

10.89.2.39 `void gazebo::sensors::GpuRaySensor::SetVerticalAngleMax (double _angle)`

Set the vertical scan line top angle.

Parameters

in	<i>_angle</i>	The Maximum angle of the scan block
----	---------------	-------------------------------------

10.89.2.40 void gazebo::sensors::GpuRaySensor::SetVerticalAngleMin (double *_angle*)

Set the vertical scan bottom angle.

Parameters

in	<i>_angle</i>	The minimum angle of the scan block
----	---------------	-------------------------------------

10.89.2.41 virtual bool gazebo::sensors::GpuRaySensor::UpdateImpl (bool) [protected],[virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	<i>_force</i>	True if update is forced, false if not
----	---------------	--

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 917).

10.89.3 Member Data Documentation

10.89.3.1 sdf::ElementPtr gazebo::sensors::GpuRaySensor::cameraElem [protected]

Camera SDF element.

10.89.3.2 sdf::ElementPtr gazebo::sensors::GpuRaySensor::horzElem [protected]

Horizontal SDF element.

10.89.3.3 unsigned int gazebo::sensors::GpuRaySensor::horzRangeCount [protected]

Horizontal range count.

10.89.3.4 unsigned int gazebo::sensors::GpuRaySensor::horzRayCount [protected]

Horizontal ray count.

10.89.3.5 double gazebo::sensors::GpuRaySensor::rangeCountRatio [protected]

Range count ratio.

10.89.3.6 sdf::ElementPtr gazebo::sensors::GpuRaySensor::rangeElem [protected]

Range SDF element.

10.89.3.7 sdf::ElementPtr gazebo::sensors::GpuRaySensor::scanElem [protected]

Scan SDF element.

10.89.3.8 sdf::ElementPtr gazebo::sensors::GpuRaySensor::vertElem [protected]

Vertical SDF element.

10.89.3.9 unsigned int gazebo::sensors::GpuRaySensor::vertRangeCount [protected]

Vertical range count.

10.89.3.10 unsigned int gazebo::sensors::GpuRaySensor::vertRayCount [protected]

Vertical ray count.

The documentation for this class was generated from the following file:

- **GpuRaySensor.hh**

10.90 gazebo::rendering::Grid Class Reference

Displays a grid of cells, drawn with lines.

```
#include <rendering/rendering.hh>
```

Public Member Functions

- **Grid** (**Scene** *_scene, uint32_t _cellCount, float _cellLength, float _lineWidth, const **common::Color** &_color)
Constructor.
- **~Grid** ()
Destructor.
- void **Enable** (bool _enable)
Enable or disable the grid.
- uint32_t **GetCellCount** () const
Get the number of cells.
- float **GetCellLength** () const
Get the cell length.
- **common::Color** **GetColor** () const

Return the grid color.

- uint32_t **GetHeight** () const
Get the height of the grid.
- float **GetLineWidth** () const
Get the width of the grid line.
- Ogre::SceneNode * **GetSceneNode** ()
*Get the **Ogre** (p. 137) scene node associated with this grid.*
- void **Init** ()
Initialize the grid.
- void **SetCellCount** (uint32_t _count)
Set the number of cells.
- void **SetCellLength** (float _len)
Set the cell length.
- void **SetColor** (const common::Color &_color)
Sets the color of the grid.
- void **SetHeight** (uint32_t _count)
Set the height of the grid.
- void **SetLineWidth** (float _width)
Set the line width.
- void **SetUserData** (const Ogre::Any &_data)
Sets user data on all ogre objects we own.

10.90.1 Detailed Description

Displays a grid of cells, drawn with lines.

Displays a grid of cells, drawn with lines. A grid with an identity orientation is drawn along the XY plane.

10.90.2 Constructor & Destructor Documentation

10.90.2.1 gazebo::rendering::Grid::Grid (Scene * _scene, uint32_t _cellCount, float _cellLength, float _lineWidth, const common::Color & _color)

Constructor.

Parameters

in	<code>_scene</code>	The scene this object is part of
in	<code>_cellCount</code>	The number of cells to draw
in	<code>_cellLength</code>	The size of each cell
in	<code>_lineWidth</code>	The width of the lines to use
in	<code>_color</code>	The color of the grid

10.90.2.2 gazebo::rendering::Grid::~~Grid ()

Destructor.

10.90.3 Member Function Documentation

10.90.3.1 void gazebo::rendering::Grid::Enable (bool *_enable*)

Enable or disable the grid.

Parameters

<code>in</code>	<code>_enable</code>	Set to true to view the grid, false to make invisible.
-----------------	----------------------	--

10.90.3.2 uint32_t gazebo::rendering::Grid::GetCellCount () const [inline]

Get the number of cells.

10.90.3.3 float gazebo::rendering::Grid::GetCellLength () const [inline]

Get the cell length.

Returns

The cell length

10.90.3.4 common::Color gazebo::rendering::Grid::GetColor () const [inline]

Return the grid color.

Returns

The grid color

10.90.3.5 uint32_t gazebo::rendering::Grid::GetHeight () const [inline]

Get the height of the grid.

Returns

The height

10.90.3.6 float gazebo::rendering::Grid::GetLineWidth () const [inline]

Get the width of the grid line.

Returns

The line width

10.90.3.7 `Ogre::SceneNode*` gazebo::rendering::Grid::GetSceneNode () [inline]

Get the **Ogre** (p. 137) scene node associated with this grid.

Returns

The **Ogre** (p. 137) scene node associated with this grid

10.90.3.8 `void` gazebo::rendering::Grid::Init ()

Initialize the grid.

10.90.3.9 `void` gazebo::rendering::Grid::SetCellCount (`uint32_t` *_count*)

Set the number of cells.

Parameters

<code>in</code>	<i>_count</i>	The number of cells
-----------------	---------------	---------------------

10.90.3.10 `void` gazebo::rendering::Grid::SetCellLength (`float` *_len*)

Set the cell length.

Parameters

<code>in</code>	<i>_len</i>	The cell length
-----------------	-------------	-----------------

10.90.3.11 `void` gazebo::rendering::Grid::SetColor (`const common::Color &` *_color*)

Sets the color of the grid.

Parameters

<code>in</code>	<i>_color</i>	The grid color
-----------------	---------------	----------------

10.90.3.12 `void` gazebo::rendering::Grid::SetHeight (`uint32_t` *_count*)

Set the height of the grid.

Parameters

<code>in</code>	<i>_count</i>	Grid (p. 488) height
-----------------	---------------	-----------------------------

10.90.3.13 `void` gazebo::rendering::Grid::SetLineWidth (`float` *_width*)

Set the line width.

Parameters

in	<code>_width</code>	The width of the grid
----	---------------------	-----------------------

10.90.3.14 void gazebo::rendering::Grid::SetUserData (const Ogre::Any & *_data*)

Sets user data on all ogre objects we own.

Parameters

in	<code>_data</code>	The user data
----	--------------------	---------------

The documentation for this class was generated from the following file:

- **Grid.hh**

10.91 gazebo::physics::Gripper Class Reference

A gripper abstraction.

```
#include <physics/physics.hh>
```

Public Member Functions

- **Gripper (ModelPtr _model)**
Constructor.
- virtual **~Gripper ()**
Destructor.
- std::string **GetName ()** const
Return the name of the gripper.
- virtual void **Init ()**
Initialize.
- bool **IsAttached ()** const
True if the gripper is attached to another model.
- virtual void **Load (sdf::ElementPtr _sdf)**
Load the gripper.

Protected Attributes

- **transport::NodePtr node**
Node for communication.

10.91.1 Detailed Description

A gripper abstraction.

A gripper is a collection of links that act as a gripper. This class will intelligently generate fixed joints between the gripper and an object within the gripper. This allows the object to be manipulated without falling or behaving poorly.

10.91.2 Constructor & Destructor Documentation

10.91.2.1 gazebo::physics::Gripper::Gripper (ModelPtr *_model*) [explicit]

Constructor.

Parameters

<i>in</i>	<i>_model</i>	The model which contains the Gripper (p. 492).
-----------	---------------	---

10.91.2.2 virtual gazebo::physics::Gripper::~~Gripper () [virtual]

Destructor.

10.91.3 Member Function Documentation

10.91.3.1 std::string gazebo::physics::Gripper::GetName () const

Return the name of the gripper.

10.91.3.2 virtual void gazebo::physics::Gripper::Init () [virtual]

Initialize.

10.91.3.3 bool gazebo::physics::Gripper::IsAttached () const

True if the gripper is attached to another model.

Returns

True if the gripper is active and a joint has been created between the gripper and another model.

10.91.3.4 virtual void gazebo::physics::Gripper::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the gripper.

Parameters

<i>in</i>	<i>_sdf</i>	Shared point to an sdf element that contains the list of links in the gripper.
-----------	-------------	--

10.91.4 Member Data Documentation

10.91.4.1 transport::NodePtr gazebo::physics::Gripper::node [protected]

Node for communication.

The documentation for this class was generated from the following file:

- **Gripper.hh**

10.92 gazebo::rendering::GUIOverlay Class Reference

A class that creates a CEGUI overlay on a render window.

```
#include <rendering/rendering.hh>
```

Public Member Functions

- **GUIOverlay** ()
Constructor.
- virtual **~GUIOverlay** ()
Destructor.
- bool **AttachCameraToImage** (**CameraPtr** &_camera, const std::string &_windowName)
*Use this function to draw the output from a **rendering::Camera** (p. 197) to and overlay window.*
- bool **AttachCameraToImage** (**DepthCameraPtr** &_camera, const std::string &_windowName)
*Use this function to draw the output from a **rendering::DepthCamera** (p. 383) to and overlay window.*
- template<typename T >
void **ButtonCallback** (const std::string &_buttonName, void(T::*_fp)(), T *_obj)
Register a CEGUI button callback.
- void **CreateWindow** (const std::string &_type, const std::string &_name, const std::string &_parent, const **math::Vector2d** &_position, const **math::Vector2d** &_size, const std::string &_text)
Create a new window on the overlay.
- bool **HandleKeyPressEvent** (const std::string &_key)
Handle a key press event.
- bool **HandleKeyReleaseEvent** (const std::string &_key)
Handle a key release event.
- bool **HandleMouseEvent** (const **common::MouseEvent** &_evt)
Handle a mouse event.
- void **Hide** ()
Make the overlay invisible.
- void **Init** (Ogre::RenderTarget *_renderTarget)
Initialize the overlay.
- bool **IsInitialized** ()
Return true if the overlay has been initialized.
- void **LoadLayout** (const std::string &_filename)
Load a CEGUI layout file.
- void **Resize** (unsigned int _width, unsigned int _height)
Resize the window.
- void **Show** ()
Make the overlay visible.
- void **Update** ()
Update the overlay's objects.

Public Attributes

- std::map< std::string,
boost::function< void()> > **callbacks**
Map of callback functions to names.

10.92.1 Detailed Description

A class that creates a CEGUI overlay on a render window.

10.92.2 Constructor & Destructor Documentation

10.92.2.1 gazebo::rendering::GUIOverlay::GUIOverlay ()

Constructor.

10.92.2.2 virtual gazebo::rendering::GUIOverlay::~GUIOverlay () [virtual]

Destructor.

10.92.3 Member Function Documentation

10.92.3.1 bool gazebo::rendering::GUIOverlay::AttachCameraToImage (CameraPtr & _camera, const std::string & _windowName)

Use this function to draw the output from a **rendering::Camera** (p. 197) to and overlay window.

Parameters

in	<i>_camera</i>	Pointer to the camera.
in	<i>_windowName</i>	Name of the window to receive the camera image

Returns

True if successful

10.92.3.2 bool gazebo::rendering::GUIOverlay::AttachCameraToImage (DepthCameraPtr & _camera, const std::string & _windowName)

Use this function to draw the output from a **rendering::DepthCamera** (p. 383) to and overlay window.

Parameters

in	<i>_camera</i>	Pointer to the camera.
in	<i>_windowName</i>	Name of the window to receive the camera image

Returns

True if successful

10.92.3.3 template<typename T > void gazebo::rendering::GUIOverlay::ButtonCallback (const std::string & _buttonName, void(T::*)() _fp, T * _obj) [inline]

Register a CEGUI button callback.

Assign a callback to a name button.

Parameters

in	<code>_buttonName</code>	Name of the button.
in	<code>_fp</code>	Function pointer to the callback.
in	<code>_obj</code>	Class pointer that contains <code>_fp</code> .

References callbacks, and gzerr.

10.92.3.4 `void gazebo::rendering::GUIOverlay::CreateWindow (const std::string & _type, const std::string & _name, const std::string & _parent, const math::Vector2d & _position, const math::Vector2d & _size, const std::string & _text)`

Create a new window on the overlay.

Parameters

in	<code>_type</code>	The window type. This should match a CEGUI window type. See <code>CEGUI::WindowManager::getSingleton().createWindow()</code> .
in	<code>_name</code>	Unique name for the window.
in	<code>_parent</code>	Name of the parent window.
in	<code>_position</code>	Position of the window within the parent.
in	<code>_size</code>	Size of the window.
in	<code>_text</code>	Display title of the window.

10.92.3.5 `bool gazebo::rendering::GUIOverlay::HandleKeyPressEvent (const std::string & _key)`

Handle a key press event.

Parameters

in	<code>_key</code>	The key pressed.
----	-------------------	------------------

Returns

True if the key press event was handled.

10.92.3.6 `bool gazebo::rendering::GUIOverlay::HandleKeyReleaseEvent (const std::string & _key)`

Handle a key release event.

Parameters

in	<code>_key</code>	The key released.
----	-------------------	-------------------

Returns

True if the key release event was handled.

10.92.3.7 `bool gazebo::rendering::GUIOverlay::HandleMouseEvent (const common::MouseEvent & _evt)`

Handle a mouse event.

Parameters

<i>in</i>	<i>_evt</i>	The mouse event.
-----------	-------------	------------------

Returns

True if the mouse event was handled.

10.92.3.8 void gazebo::rendering::GUIOverlay::Hide ()

Make the overlay invisible.

10.92.3.9 void gazebo::rendering::GUIOverlay::Init (Ogre::RenderTarget * *_renderTarget*)

Initialize the overlay.

Parameters

<i>in</i>	<i>_renderTarget</i>	The render target which will have the overlay.
-----------	----------------------	--

10.92.3.10 bool gazebo::rendering::GUIOverlay::IsInitialized ()

Return true if the overlay has been initialized.

Returns

True if initialized

10.92.3.11 void gazebo::rendering::GUIOverlay::LoadLayout (const std::string & *_filename*)

Load a CEGUI layout file.

Parameters

<i>in</i>	<i>_filename</i>	Name of the layout file.
-----------	------------------	--------------------------

10.92.3.12 void gazebo::rendering::GUIOverlay::Resize (unsigned int *_width*, unsigned int *_height*)

Resize the window.

10.92.3.13 void gazebo::rendering::GUIOverlay::Show ()

Make the overlay visible.

10.92.3.14 void gazebo::rendering::GUIOverlay::Update ()

Update the overlay's objects.

10.92.4 Member Data Documentation

10.92.4.1 `std::map<std::string, boost::function<void()> > gazebo::rendering::GUIOverlay::callbacks`

Map of callback functions to names.

Referenced by `ButtonCallback()`.

The documentation for this class was generated from the following file:

- `GUIOverlay.hh`

10.93 gazebo::rendering::GUIOverlayPrivate Class Reference

Private data for the `GUIOverlay` (p. 494) class.

```
#include <GUIOverlayPrivate.hh>
```

Public Attributes

- `std::vector< event::ConnectionPtr > connections`
All the connections.
- `bool initialized`
True if initialized.
- `std::string layoutFilename`
The layout file used to create gui elements.
- `unsigned int rttImageSetCount`
Used in the `AttachCameraToImage` function to create unique names.

10.93.1 Detailed Description

Private data for the `GUIOverlay` (p. 494) class.

10.93.2 Member Data Documentation

10.93.2.1 `std::vector<event::ConnectionPtr> gazebo::rendering::GUIOverlayPrivate::connections`

All the connections.

10.93.2.2 `bool gazebo::rendering::GUIOverlayPrivate::initialized`

True if initialized.

10.93.2.3 `std::string gazebo::rendering::GUIOverlayPrivate::layoutFilename`

The layout file used to create gui elements.

10.93.2.4 unsigned int gazebo::rendering::GUIOverlayPrivate::rttlImageSetCount

Used in the AttachCameraToImage function to create unique names.

The documentation for this class was generated from the following file:

- **GUIOverlayPrivate.hh**

10.94 gazebo::rendering::GzTerrainMatGen Class Reference

```
#include <Heightmap.hh>
```

Classes

- class **SM2Profile**
Shader model 2 profile target.

Public Member Functions

- **GzTerrainMatGen** ()
Constructor.
- virtual **~GzTerrainMatGen** ()
Destructor.

10.94.1 Constructor & Destructor Documentation

10.94.1.1 gazebo::rendering::GzTerrainMatGen::GzTerrainMatGen ()

Constructor.

10.94.1.2 virtual gazebo::rendering::GzTerrainMatGen::~GzTerrainMatGen () [virtual]

Destructor.

The documentation for this class was generated from the following file:

- **Heightmap.hh**

10.95 gazebo::rendering::Heightmap Class Reference

Rendering a terrain using heightmap information.

```
#include <rendering/rendering.hh>
```

Public Member Functions

- **Heightmap** (**ScenePtr** _scene)
Constructor.
- virtual **~Heightmap** ()
Destructor.
- bool **Flatten** (**CameraPtr** _camera, **math::Vector2i** _mousePos, double _outsideRadius, double _insideRadius, double _weight=0.1)
Flatten the terrain based on a mouse press.
- double **GetAvgHeight** (**Ogre::Vector3** _pos, double _brushSize)
Get the average height around a point.
- double **GetHeight** (double _x, double _y, double _z=1000)
Get the height at a location.
- **common::Image GetImage** () const
Get the heightmap as an image.
- **Ogre::TerrainGroup::RayResult GetMouseHit** (**CameraPtr** _camera, **math::Vector2i** _mousePos)
Calculate a mouse ray hit on the terrain.
- **Ogre::TerrainGroup * GetOgreTerrain** () const
Get a pointer to the OGRE terrain group object.
- unsigned int **GetTerrainSubdivisionCount** () const
Get the number of subdivision the terrain will be split into.
- void **Load** ()
Load the heightmap.
- void **LoadFromMsg** (**ConstVisualPtr** &_msg)
Load the heightmap from a visual message.
- bool **Lower** (**CameraPtr** _camera, **math::Vector2i** _mousePos, double _outsideRadius, double _insideRadius, double _weight=0.1)
Lower the terrain based on a mouse press.
- bool **Raise** (**CameraPtr** _camera, **math::Vector2i** _mousePos, double _outsideRadius, double _insideRadius, double _weight=0.1)
Raise the terrain based on a mouse press.
- void **SetWireframe** (bool _show)
Set the heightmap to render in wireframe mode.
- bool **Smooth** (**CameraPtr** _camera, **math::Vector2i** _mousePos, double _outsideRadius, double _insideRadius, double _weight=0.1)
Smooth the terrain based on a mouse press.
- void **SplitHeights** (const std::vector< float > &_heightmap, int _n, std::vector< std::vector< float > > &_v)
Split a terrain into subterrains.

10.95.1 Detailed Description

Rendering a terrain using heightmap information.

10.95.2 Constructor & Destructor Documentation

10.95.2.1 gazebo::rendering::Heightmap::Heightmap (ScenePtr *_scene*)

Constructor.

Parameters

in	<i>_scene</i>	Pointer to the scene that will contain the heightmap
----	---------------	--

10.95.2.2 virtual gazebo::rendering::Heightmap::~Heightmap () [virtual]

Destructor.

10.95.3 Member Function Documentation

10.95.3.1 bool gazebo::rendering::Heightmap::Flatten (CameraPtr *_camera*, math::Vector2i *_mousePos*, double *_outsideRadius*, double *_insideRadius*, double *_weight* = 0.1)

Flatten the terrain based on a mouse press.

Parameters

in	<i>_camera</i>	Camera (p. 197) associated with the mouse press.
in	<i>_mousePos</i>	Position of the mouse in viewport coordinates.
in	<i>_outsideRadius</i>	Controls the radius of effect.
in	<i>_insideRadius</i>	Controls the size of the radius with the maximum effect (value between 0 and 1).
in	<i>_weight</i>	Controls modification magnitude.

Returns

True if the terrain was modified

10.95.3.2 double gazebo::rendering::Heightmap::GetAvgHeight (Ogre::Vector3 *_pos*, double *_brushSize*)

Get the average height around a point.

Parameters

in	<i>_pos</i>	Position in world coordinates.
in	<i>_brushSize</i>	Controls the radius of effect.

10.95.3.3 double gazebo::rendering::Heightmap::GetHeight (double *_x*, double *_y*, double *_z* = 1000)

Get the height at a location.

Parameters

in	<i>_x</i>	X location
in	<i>_y</i>	Y location
in	<i>_z</i>	Z location

Returns

The height at the specified location

10.95.3.4 `common::Image gazebo::rendering::Heightmap::GetImage () const`

Get the heightmap as an image.

Returns

An image that contains the terrain data.

10.95.3.5 `Ogre::TerrainGroup::RayResult gazebo::rendering::Heightmap::GetMouseHit (CameraPtr _camera, math::Vector2i _mousePos)`

Calculate a mouse ray hit on the terrain.

Parameters

<code>in</code>	<code>_camera</code>	Camera (p. 197) associated with the mouse press.
<code>in</code>	<code>_mousePos</code>	Position of the mouse in viewport coordinates.

Returns

The result of the mouse ray hit.

10.95.3.6 `Ogre::TerrainGroup* gazebo::rendering::Heightmap::GetOgreTerrain () const`

Get a pointer to the OGRE terrain group object.

Returns

Pointer to the OGRE terrain.

10.95.3.7 `unsigned int gazebo::rendering::Heightmap::GetTerrainSubdivisionCount () const`

Get the number of subdivision the terrain will be split into.

Returns

Number of terrain subdivisions

10.95.3.8 `void gazebo::rendering::Heightmap::Load ()`

Load the heightmap.

10.95.3.9 void gazebo::rendering::Heightmap::LoadFromMsg (ConstVisualPtr & *_msg*)

Load the heightmap from a visual message.

Parameters

in	<i>_msg</i>	The visual message containing heightmap info
----	-------------	--

10.95.3.10 bool gazebo::rendering::Heightmap::Lower (CameraPtr *_camera*, math::Vector2i *_mousePos*, double *_outsideRadius*, double *_insideRadius*, double *_weight* = 0.1)

Lower the terrain based on a mouse press.

Parameters

in	<i>_camera</i>	Camera (p. 197) associated with the mouse press.
in	<i>_mousePos</i>	Position of the mouse in viewport coordinates.
in	<i>_outsideRadius</i>	Controls the radius of effect.
in	<i>_insideRadius</i>	Controls the size of the radius with the maximum effect (value between 0 and 1).
in	<i>_weight</i>	Controls modification magnitude.

Returns

True if the terrain was modified

10.95.3.11 bool gazebo::rendering::Heightmap::Raise (CameraPtr *_camera*, math::Vector2i *_mousePos*, double *_outsideRadius*, double *_insideRadius*, double *_weight* = 0.1)

Raise the terrain based on a mouse press.

Parameters

in	<i>_camera</i>	Camera (p. 197) associated with the mouse press.
in	<i>_mousePos</i>	Position of the mouse in viewport coordinates.
in	<i>_outsideRadius</i>	Controls the radius of effect.
in	<i>_insideRadius</i>	Controls the size of the radius with the maximum effect (value between 0 and 1).
in	<i>_weight</i>	Controls modification magnitude.

Returns

True if the terrain was modified

10.95.3.12 void gazebo::rendering::Heightmap::SetWireframe (bool *_show*)

Set the heightmap to render in wireframe mode.

Parameters

in	<i>_show</i>	True to render wireframe, false to render solid.
----	--------------	--

10.95.3.13 `bool gazebo::rendering::Heightmap::Smooth (CameraPtr _camera, math::Vector2i _mousePos, double _outsideRadius, double _insideRadius, double _weight = 0.1)`

Smooth the terrain based on a mouse press.

Parameters

in	<code>_camera</code>	Camera (p. 197) associated with the mouse press.
in	<code>_mousePos</code>	Position of the mouse in viewport coordinates.
in	<code>_outsideRadius</code>	Controls the radius of effect.
in	<code>_insideRadius</code>	Controls the size of the radius with the maximum effect (value between 0 and 1).
in	<code>_weight</code>	Controls modification magnitude.

Returns

True if the terrain was modified

10.95.3.14 `void gazebo::rendering::Heightmap::SplitHeights (const std::vector< float > & _heightmap, int _n, std::vector< std::vector< float > > & _v)`

Split a terrain into subterrains.

Parameters

in	<code>_heightmap</code>	Source vector of floats with the heights.
in	<code>_n</code>	Number of subterrains.
out	<code>_v</code>	Destination vector with the subterrains.

The documentation for this class was generated from the following file:

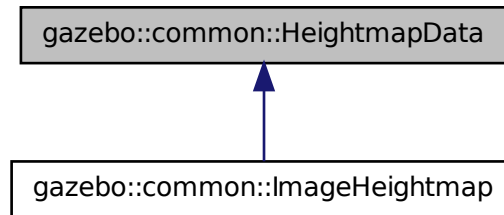
- **Heightmap.hh**

10.96 gazebo::common::HeightmapData Class Reference

Encapsulates a generic heightmap data file.

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::common::HeightmapData:



Public Member Functions

- virtual `~HeightmapData ()`
Destructor.
- virtual void **FillHeightMap** (int `_subSampling`, unsigned int `_vertSize`, const `math::Vector3` & `_size`, const `math::Vector3` & `_scale`, bool `_flipY`, std::vector< float > & `_heights`)=0
Create a lookup table of the terrain's height.
- virtual unsigned int **GetHeight** () const =0
Get the terrain's height.
- virtual float **GetMaxElevation** () const =0
Get the maximum terrain's elevation.
- virtual unsigned int **GetWidth** () const =0
Get the terrain's width.

10.96.1 Detailed Description

Encapsulates a generic heightmap data file.

10.96.2 Constructor & Destructor Documentation

10.96.2.1 virtual gazebo::common::HeightmapData::~HeightmapData () [inline],[virtual]

Destructor.

10.96.3 Member Function Documentation

10.96.3.1 virtual void gazebo::common::HeightmapData::FillHeightMap (int `_subSampling`, unsigned int `_vertSize`, const `math::Vector3` & `_size`, const `math::Vector3` & `_scale`, bool `_flipY`, std::vector< float > & `_heights`) [pure virtual]

Create a lookup table of the terrain's height.

Parameters

in	<code>_subsampling</code>	Multiplier used to increase the resolution. Ex: A subsampling of 2 in a terrain of 129x129 means that the height vector will be 257 * 257.
in	<code>_vertSize</code>	Number of points per row.
in	<code>_size</code>	Real dimensions of the terrain.
in	<code>_scale</code>	Vector3 used to scale the height.
in	<code>_flipY</code>	If true, it inverts the order in which the vector is filled.
out	<code>_heights</code>	Vector containing the terrain heights.

Implemented in **gazebo::common::ImageHeightmap** (p. 524).

10.96.3.2 `virtual unsigned int gazebo::common::HeightmapData::GetHeight () const [pure virtual]`

Get the terrain's height.

Returns

The terrain's height.

Implemented in **gazebo::common::ImageHeightmap** (p. 524).

10.96.3.3 `virtual float gazebo::common::HeightmapData::GetMaxElevation () const [pure virtual]`

Get the maximum terrain's elevation.

Returns

The maximum terrain's elevation.

Implemented in **gazebo::common::ImageHeightmap** (p. 525).

10.96.3.4 `virtual unsigned int gazebo::common::HeightmapData::GetWidth () const [pure virtual]`

Get the terrain's width.

Returns

The terrain's width.

Implemented in **gazebo::common::ImageHeightmap** (p. 525).

The documentation for this class was generated from the following file:

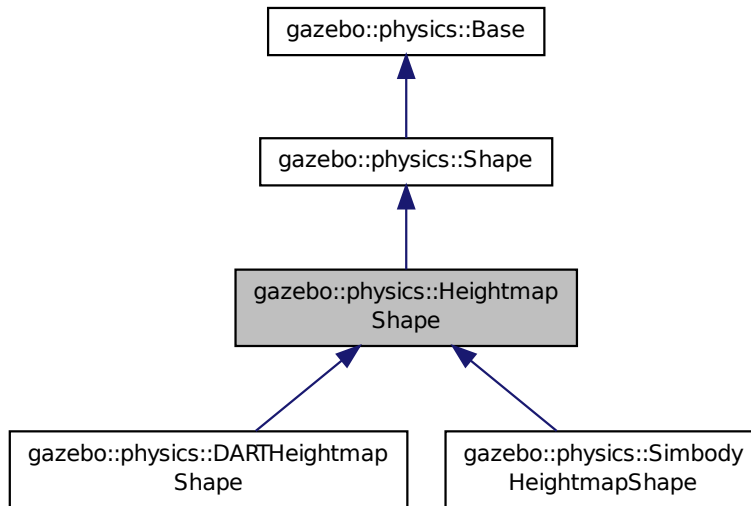
- **HeightmapData.hh**

10.97 gazebo::physics::HeightmapShape Class Reference

HeightmapShape (p. 506) collision shape builds a heightmap from an image.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::HeightmapShape:



Public Member Functions

- **HeightmapShape** (*CollisionPtr* _parent)
Constructor.
- virtual **~HeightmapShape** ()
Destructor.
- void **FillMsg** (*msgs::Geometry* &_msg)
Fill a geometry message with this shape's data.
- float **GetHeight** (int _x, int _y) const
Get a height at a position.
- **common::Image** **GetImage** () const
Return an image representation of the heightmap.
- float **GetMaxHeight** () const
Get the maximum height.
- float **GetMinHeight** () const
Get the minimum height.
- **math::Vector3** **GetPos** () const
Get the origin in world coordinate frame.
- **math::Vector3** **GetSize** () const
Get the size in meters.
- int **GetSubSampling** () const
Get the amount of subsampling.
- std::string **GetURI** () const
Get the URI of the heightmap image.

- **math::Vector2i GetVertexCount** () const
Return the number of vertices, which equals the size of the image used to load the heightmap.
- virtual void **Init** ()
Initialize the heightmap.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the heightmap.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
Update the heightmap from a message.
- virtual void **SetScale** (const **math::Vector3** &_scale)
Set the scale of the heightmap shape.

Protected Attributes

- bool **flipY**
True to flip the heights along the y direction.
- **common::HeightmapData * heightmapData**
HeightmapData used to generate the heights.
- std::vector< float > **heights**
Lookup table of heights.
- **common::ImageHeightmap img**
Image used to generate the heights.
- int **subSampling**
The amount of subsampling. Default is 2.
- unsigned int **vertSize**
Size of the height lookup table.

Additional Inherited Members

10.97.1 Detailed Description

HeightmapShape (p. 506) collision shape builds a heightmap from an image.

The supplied image must be square with $N*N+1$ pixels per side, where N is an integer.

10.97.2 Constructor & Destructor Documentation

10.97.2.1 gazebo::physics::HeightmapShape::HeightmapShape (CollisionPtr _parent) [explicit]

Constructor.

Parameters

in	<code>_parent</code>	Parent Collision (p. 235) object.
----	----------------------	--

10.97.2.2 virtual gazebo::physics::HeightmapShape::~~HeightmapShape () [virtual]

Destructor.

10.97.3 Member Function Documentation

10.97.3.1 void gazebo::physics::HeightmapShape::FillMsg (msgs::Geometry & _msg) [virtual]

Fill a geometry message with this shape's data.

Parameters

in	_msg	Message to fill.
----	------	------------------

Implements **gazebo::physics::Shape** (p. 934).

10.97.3.2 float gazebo::physics::HeightmapShape::GetHeight (int _x, int _y) const

Get a height at a position.

Parameters

in	_x	X position.
in	_y	Y position.

Returns

The height at a the specified location.

10.97.3.3 common::Image gazebo::physics::HeightmapShape::GetImage () const

Return an image representation of the heightmap.

Returns

Image where white pixels represents the highest locations, and black pixels the lowest.

10.97.3.4 float gazebo::physics::HeightmapShape::GetMaxHeight () const

Get the maximum height.

Returns

The maximum height.

10.97.3.5 float gazebo::physics::HeightmapShape::GetMinHeight () const

Get the minimum height.

Returns

The minimum height.

10.97.3.6 `math::Vector3 gazebo::physics::HeightmapShape::GetPos () const`

Get the origin in world coordinate frame.

Returns

The origin in world coordinate frame.

10.97.3.7 `math::Vector3 gazebo::physics::HeightmapShape::GetSize () const`

Get the size in meters.

Returns

The size in meters.

10.97.3.8 `int gazebo::physics::HeightmapShape::GetSubSampling () const`

Get the amount of subsampling.

Returns

Amount of subsampling.

10.97.3.9 `std::string gazebo::physics::HeightmapShape::GetURI () const`

Get the URI of the heightmap image.

Returns

The heightmap image URI.

10.97.3.10 `math::Vector2i gazebo::physics::HeightmapShape::GetVertexCount () const`

Return the number of vertices, which equals the size of the image used to load the heightmap.

Returns

`math::Vector2i` (p. 1156), result.x = width, result.y = length/height.

10.97.3.11 `virtual void gazebo::physics::HeightmapShape::Init () [virtual]`

Initialize the heightmap.

Implements `gazebo::physics::Shape` (p. 935).

Reimplemented in `gazebo::physics::SimbodyHeightmapShape` (p. 950), and `gazebo::physics::DARTHeightmapShape` (p. 317).

10.97.3.12 virtual void gazebo::physics::HeightmapShape::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the heightmap.

Parameters

in	<i>_sdf</i>	SDF value to load from.
----	-------------	-------------------------

Reimplemented from **gazebo::physics::Base** (p. 177).

10.97.3.13 virtual void gazebo::physics::HeightmapShape::ProcessMsg (const msgs::Geometry & *_msg*) [virtual]

Update the heightmap from a message.

Parameters

in	<i>_msg</i>	Message to update from.
----	-------------	-------------------------

Implements **gazebo::physics::Shape** (p. 935).

10.97.3.14 virtual void gazebo::physics::HeightmapShape::SetScale (const math::Vector3 & *_scale*) [virtual]

Set the scale of the heightmap shape.

Parameters

in	<i>_scale</i>	Scale to set the heightmap shape to.
----	---------------	--------------------------------------

Implements **gazebo::physics::Shape** (p. 935).

10.97.4 Member Data Documentation

10.97.4.1 bool gazebo::physics::HeightmapShape::flipY [protected]

True to flip the heights along the y direction.

10.97.4.2 common::HeightmapData* gazebo::physics::HeightmapShape::heightmapData [protected]

HeightmapData used to generate the heights.

10.97.4.3 std::vector<float> gazebo::physics::HeightmapShape::heights [protected]

Lookup table of heights.

10.97.4.4 common::ImageHeightmap gazebo::physics::HeightmapShape::img [protected]

Image used to generate the heights.

10.97.4.5 `int gazebo::physics::HeightmapShape::subSampling` [protected]

The amount of subsampling. Default is 2.

10.97.4.6 `unsigned int gazebo::physics::HeightmapShape::vertSize` [protected]

Size of the height lookup table.

The documentation for this class was generated from the following file:

- **HeightmapShape.hh**

10.98 gazebo::physics::Hinge2Joint< T > Class Template Reference

A two axis hinge joint.

```
#include <physics/physics.hh>
```

Public Member Functions

- **Hinge2Joint** (**BasePtr** _parent)
Constructor.
- virtual **~Hinge2Joint** ()
Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the joint.

10.98.1 Detailed Description

```
template<class T>class gazebo::physics::Hinge2Joint< T >
```

A two axis hinge joint.

10.98.2 Constructor & Destructor Documentation

10.98.2.1 `template<class T> gazebo::physics::Hinge2Joint< T >::Hinge2Joint (BasePtr _parent)` [inline], [explicit]

Constructor.

Parameters

<code>in</code>	<code>_parent</code>	Parent link.
-----------------	----------------------	--------------

10.98.2.2 `template<class T> virtual gazebo::physics::Hinge2Joint< T >::~~Hinge2Joint () [inline], [virtual]`

Destructor.

10.98.3 Member Function Documentation

10.98.3.1 `template<class T> virtual unsigned int gazebo::physics::Hinge2Joint< T >::GetAngleCount () const [inline], [virtual]`

10.98.3.2 `template<class T> virtual void gazebo::physics::Hinge2Joint< T >::Load (sdf::ElementPtr _sdf) [inline], [virtual]`

Load the joint.

Parameters

in	_sdf	SDF values to load from.
----	------	--------------------------

Reimplemented in `gazebo::physics::SimbodyHinge2Joint` (p. 954), and `gazebo::physics::DARTHinge2Joint` (p. 321).

The documentation for this class was generated from the following file:

- `Hinge2Joint.hh`

10.99 gazebo::physics::HingeJoint< T > Class Template Reference

A single axis hinge joint.

```
#include <physics/physics.hh>
```

Public Member Functions

- **HingeJoint** (**BasePtr** _parent)
Constructor.
- virtual **~HingeJoint** ()
Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (sdf::ElementPtr _sdf)
Load joint.

Protected Member Functions

- virtual void **Init** ()
Initialize joint.

10.99.1 Detailed Description

```
template<class T>class gazebo::physics::HingeJoint< T >
```

A single axis hinge joint.

10.99.2 Constructor & Destructor Documentation

10.99.2.1 `template<class T> gazebo::physics::HingeJoint< T >::HingeJoint (BasePtr _parent) [inline]`

Constructor.

Parameters

<code>in</code>	<code><i>_parent</i></code>	Parent link
-----------------	-----------------------------	-------------

10.99.2.2 `template<class T> virtual gazebo::physics::HingeJoint< T >::~~HingeJoint () [inline],[virtual]`

Destructor.

10.99.3 Member Function Documentation

10.99.3.1 `template<class T> virtual unsigned int gazebo::physics::HingeJoint< T >::GetAngleCount () const [inline],[virtual]`

10.99.3.2 `template<class T> virtual void gazebo::physics::HingeJoint< T >::Init () [inline],[protected],[virtual]`

Initialize joint.

Reimplemented in `gazebo::physics::DARTHingeJoint` (p. 326).

10.99.3.3 `template<class T> virtual void gazebo::physics::HingeJoint< T >::Load (sdf::ElementPtr _sdf) [inline],[virtual]`

Load joint.

Parameters

<code>in</code>	<code><i>_sdf</i></code>	Pointer to SDF element
-----------------	--------------------------	------------------------

Reimplemented in `gazebo::physics::SimbodyHingeJoint` (p. 958), and `gazebo::physics::DARTHingeJoint` (p. 326).

The documentation for this class was generated from the following file:

- `HingeJoint.hh`

10.100 gazebo::common::Image Class Reference

Encapsulates an image.

```
#include <common/common.hh>
```

Public Types

- enum **PixelFormat** {
UNKNOWN_PIXEL_FORMAT = 0, **L_INT8**, **L_INT16**, **RGB_INT8**,
RGBA_INT8, **BGRA_INT8**, **RGB_INT16**, **RGB_INT32**,
BGR_INT8, **BGR_INT16**, **BGR_INT32**, **R_FLOAT16**,
RGB_FLOAT16, **R_FLOAT32**, **RGB_FLOAT32**, **BAYER_RGGB8**,
BAYER_RGGR8, **BAYER_GBRG8**, **BAYER_GRBG8**, **PIXEL_FORMAT_COUNT** }

Pixel formats enumeration.

Public Member Functions

- **Image** (const std::string &_filename="")
Constructor.
- virtual ~**Image** ()
Destructor.
- **Color GetAvgColor** ()
Get the average color.
- unsigned int **GetBPP** () const
Get the size of one pixel in bits.
- void **GetData** (unsigned char **_data, unsigned int &_count) const
Get the image as a data array.
- std::string **GetFilename** () const
Get the full filename of the image.
- unsigned int **GetHeight** () const
Get the height.
- **Color GetMaxColor** () const
Get the max color.
- int **GetPitch** () const
- **Color GetPixel** (unsigned int _x, unsigned int _y) const
Get a pixel color value.
- **PixelFormat GetPixelFormat** () const
Get the pixel format.
- void **GetRGBData** (unsigned char **_data, unsigned int &_count) const
Get only the RGB data from the image.
- unsigned int **GetWidth** () const
Get the width.
- int **Load** (const std::string &_filename)
Load an image.
- void **Rescale** (int _width, int _height)
Rescale the image.
- void **SavePNG** (const std::string &_filename)

Save the image in PNG format.

- void **SetFromData** (const unsigned char *_data, unsigned int _width, unsigned int _height, **Image::PixelFormat** _format)

Set the image from raw data.

- bool **Valid** () const

Returns whether this is a valid image.

Static Public Member Functions

- static **Image::PixelFormat ConvertPixelFormat** (const std::string &_format)

*Convert a string to a **Image::PixelFormat** (p. 516).*

10.100.1 Detailed Description

Encapsulates an image.

10.100.2 Member Enumeration Documentation

10.100.2.1 enum gazebo::common::Image::PixelFormat

Pixel formats enumeration.

Enumerator:

UNKNOWN_PIXEL_FORMAT

L_INT8

L_INT16

RGB_INT8

RGBA_INT8

BGRA_INT8

RGB_INT16

RGB_INT32

BGR_INT8

BGR_INT16

BGR_INT32

R_FLOAT16

RGB_FLOAT16

R_FLOAT32

RGB_FLOAT32

BAYER_RGGB8

BAYER_RGGR8

BAYER_GBRG8

BAYER_GRBG8

PIXEL_FORMAT_COUNT

10.100.3 Constructor & Destructor Documentation

10.100.3.1 `gazebo::common::Image::Image (const std::string & _filename = " ") [explicit]`

Constructor.

Parameters

<i>in</i>	<i>_filename</i>	the path to the image
-----------	------------------	-----------------------

10.100.3.2 `virtual gazebo::common::Image::~~Image () [virtual]`

Destructor.

10.100.4 Member Function Documentation

10.100.4.1 `static Image::PixelFormat gazebo::common::Image::ConvertPixelFormat (const std::string & _format) [static]`

Convert a string to a **Image::PixelFormat** (p. 516).

Parameters

<i>in</i>	<i>_format</i>	Pixel format string.
-----------	----------------	----------------------

See Also

`Image::PixelFormatNames`

Returns

Image::PixelFormat (p. 516)

10.100.4.2 `Color gazebo::common::Image::GetAvgColor ()`

Get the average color.

Returns

The average color

10.100.4.3 `unsigned int gazebo::common::Image::GetBPP () const`

Get the size of one pixel in bits.

Returns

The BPP of the image

10.100.4.4 `void gazebo::common::Image::GetData (unsigned char ** _data, unsigned int & _count) const`

Get the image as a data array.

Parameters

out	<code><i>_data</i></code>	Pointer to a NULL array of char.
out	<code><i>_count</i></code>	The resulting data array size

10.100.4.5 `std::string gazebo::common::Image::GetFilename () const`

Get the full filename of the image.

Returns

The filename used to load the image

10.100.4.6 `unsigned int gazebo::common::Image::GetHeight () const`

Get the height.

Returns

The image height

10.100.4.7 `Color gazebo::common::Image::GetMaxColor () const`

Get the max color.

Returns

The max color

10.100.4.8 `int gazebo::common::Image::GetPitch () const`

Returns

The pitch of the image

10.100.4.9 `Color gazebo::common::Image::GetPixel (unsigned int _x, unsigned int _y) const`

Get a pixel color value.

Parameters

in	<code><i>_x</i></code>	Column location in the image
in	<code><i>_y</i></code>	Row location in the image

Returns

The color of the given pixel

10.100.4.10 PixelFormat gazebo::common::Image::GetPixelFormat () const

Get the pixel format.

Returns

PixelFormat

10.100.4.11 void gazebo::common::Image::GetRGBData (unsigned char ** _data, unsigned int & _count) const

Get only the RGB data from the image.

This will drop the alpha channel if one is present.

Parameters

out	<i>_data</i>	Pointer to a NULL array of char.
out	<i>_count</i>	The resulting data array size

10.100.4.12 unsigned int gazebo::common::Image::GetWidth () const

Get the width.

Returns

The image width

10.100.4.13 int gazebo::common::Image::Load (const std::string & _filename)

Load an image.

Return 0 on success

Parameters

in	<i>_filename</i>	the path to the image file
----	------------------	----------------------------

Returns

0 when the operation succeeds to open a file or -1 when fails.

10.100.4.14 void gazebo::common::Image::Rescale (int _width, int _height)

Rescale the image.

Parameters

in	<i>_width</i>	New image width
in	<i>_height</i>	New image height

10.100.4.15 `void gazebo::common::Image::SavePNG (const std::string & _filename)`

Save the image in PNG format.

Parameters

in	<i>_filename</i>	The name of the saved image
----	------------------	-----------------------------

10.100.4.16 `void gazebo::common::Image::SetFromData (const unsigned char * _data, unsigned int _width, unsigned int _height, Image::PixelFormat _format)`

Set the image from raw data.

Parameters

in	<i>_data</i>	Pointer to the raw image data
in	<i>_width</i>	Width in pixels
in	<i>_height</i>	Height in pixels
in	<i>_format</i>	Pixel format of the provided data

10.100.4.17 `bool gazebo::common::Image::Valid () const`

Returns whether this is a valid image.

Returns

true if image has a bitmap

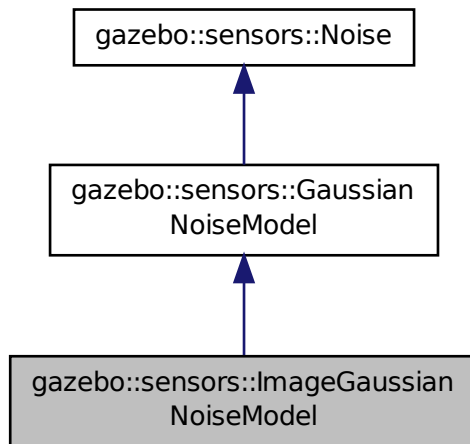
The documentation for this class was generated from the following file:

- **Image.hh**

10.101 gazebo::sensors::ImageGaussianNoiseModel Class Reference

```
#include <GaussianNoiseModel.hh>
```

Inheritance diagram for gazebo::sensors::ImageGaussianNoiseModel:



Public Member Functions

- **ImageGaussianNoiseModel** ()
Constructor.
- virtual \sim **ImageGaussianNoiseModel** ()
Destructor.
- virtual void **Fini** ()
Finalize the noise model.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load noise parameters from sdf.
- virtual void **SetCamera** (rendering::CameraPtr _camera)
Set camera needed to create image noise.

Public Attributes

- boost::shared_ptr
< GaussianNoiseCompositorListener > **gaussianNoiseCompositorListener**
Gaussian noise compositor listener.
- Ogre::CompositorInstance * **gaussianNoiseInstance**
Gaussian noise compositor.

Additional Inherited Members

10.101.1 Constructor & Destructor Documentation

10.101.1.1 `gazebo::sensors::ImageGaussianNoiseModel::ImageGaussianNoiseModel ()`

Constructor.

10.101.1.2 `virtual gazebo::sensors::ImageGaussianNoiseModel::~~ImageGaussianNoiseModel () [virtual]`

Destructor.

10.101.2 Member Function Documentation

10.101.2.1 `virtual void gazebo::sensors::ImageGaussianNoiseModel::Fini () [virtual]`

Finalize the noise model.

Reimplemented from `gazebo::sensors::GaussianNoiseModel` (p. 459).

10.101.2.2 `virtual void gazebo::sensors::ImageGaussianNoiseModel::Load (sdf::ElementPtr _sdf) [virtual]`

Load noise parameters from sdf.

Parameters

in	<code>_sdf</code>	SDF parameters.
in	<code>_sensor</code>	Type of sensor.

Reimplemented from `gazebo::sensors::GaussianNoiseModel` (p. 460).

10.101.2.3 `virtual void gazebo::sensors::ImageGaussianNoiseModel::SetCamera (rendering::CameraPtr _camera) [virtual]`

Set camera needed to create image noise.

This is only needed for image sensors, i.e. camera/multicamera/depth sensors, which use shaders for more efficient noise generation.

Parameters

in	<code>_camera</code>	Camera associated to an image sensor
----	----------------------	--------------------------------------

Reimplemented from `gazebo::sensors::Noise` (p. 751).

10.101.3 Member Data Documentation

10.101.3.1 `boost::shared_ptr<GaussianNoiseCompositorListener> gazebo::sensors::ImageGaussianNoiseModel::gaussianNoiseCompositorListener`

Gaussian noise compositor listener.

10.101.3.2 Ogre::CompositorInstance* gazebo::sensors::ImageGaussianNoiseModel::gaussianNoiseInstance

Gaussian noise compositor.

The documentation for this class was generated from the following file:

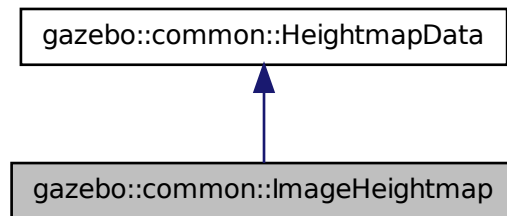
- **GaussianNoiseModel.hh**

10.102 gazebo::common::ImageHeightmap Class Reference

Encapsulates an image that will be interpreted as a heightmap.

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::common::ImageHeightmap:



Public Member Functions

- **ImageHeightmap** ()
Constructor.
- void **FillHeightMap** (int _subSampling, unsigned int _vertSize, const **math::Vector3** &_size, const **math::Vector3** &_scale, bool _flipY, std::vector< float > &_heights)
Create a lookup table of the terrain's height.
- std::string **GetFilename** () const
Get the full filename of the image.
- unsigned int **GetHeight** () const
Get the terrain's height.
- float **GetMaxElevation** () const
Get the maximum terrain's elevation.
- unsigned int **GetWidth** () const
Get the terrain's width.
- int **Load** (const std::string &_filename="")
Load an image file as a heightmap.

10.102.1 Detailed Description

Encapsulates an image that will be interpreted as a heightmap.

10.102.2 Constructor & Destructor Documentation

10.102.2.1 gazebo::common::ImageHeightmap::ImageHeightmap ()

Constructor.

Parameters

in	<code>_filename</code>	the path to the image
----	------------------------	-----------------------

10.102.3 Member Function Documentation

10.102.3.1 void gazebo::common::ImageHeightmap::FillHeightMap (int *_subSampling*, unsigned int *_vertSize*, const math::Vector3 & *_size*, const math::Vector3 & *_scale*, bool *_flipY*, std::vector< float > & *_heights*) [virtual]

Create a lookup table of the terrain's height.

Parameters

in	<code>_subsampling</code>	Multiplier used to increase the resolution. Ex: A subsampling of 2 in a terrain of 129x129 means that the height vector will be 257 * 257.
in	<code>_vertSize</code>	Number of points per row.
in	<code>_size</code>	Real dimensions of the terrain.
in	<code>_scale</code>	Vector3 used to scale the height.
in	<code>_flipY</code>	If true, it inverts the order in which the vector is filled.
out	<code>_heights</code>	Vector containing the terrain heights.

Implements **gazebo::common::HeightmapData** (p. 505).

10.102.3.2 std::string gazebo::common::ImageHeightmap::GetFilename () const

Get the full filename of the image.

Returns

The filename used to load the image

10.102.3.3 unsigned int gazebo::common::ImageHeightmap::GetHeight () const [virtual]

Get the terrain's height.

Returns

The terrain's height.

Implements **gazebo::common::HeightmapData** (p. 506).

10.102.3.4 `float gazebo::common::ImageHeightmap::GetMaxElevation () const [virtual]`

Get the maximum terrain's elevation.

Returns

The maximum terrain's elevation.

Implements **gazebo::common::HeightmapData** (p. 506).

10.102.3.5 `unsigned int gazebo::common::ImageHeightmap::GetWidth () const [virtual]`

Get the terrain's width.

Returns

The terrain's width.

Implements **gazebo::common::HeightmapData** (p. 506).

10.102.3.6 `int gazebo::common::ImageHeightmap::Load (const std::string & _filename = " ")`

Load an image file as a heightmap.

Parameters

<code>in</code>	<code>_filename</code>	the path to the image file.
-----------------	------------------------	-----------------------------

Returns

True when the operation succeeds to open a file.

The documentation for this class was generated from the following file:

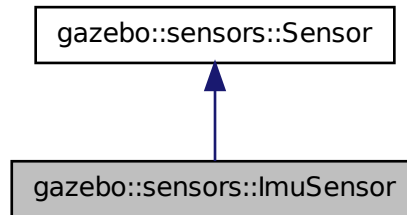
- **ImageHeightmap.hh**

10.103 gazebo::sensors::ImuSensor Class Reference

An IMU sensor.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::ImuSensor:



Public Member Functions

- **ImuSensor** ()
Constructor.
- virtual \sim **ImuSensor** ()
Destructor.
- **math::Vector3 GetAngularVelocity** () const
Returns the angular velocity.
- **msgs::IMU GetImuMessage** () const
Returns the imu message.
- **math::Vector3 GetLinearAcceleration** () const
Returns the imu linear acceleration.
- **math::Quaternion GetOrientation** () const
get orientation of the IMU relative to the reference pose
- virtual void **Init** ()
Initialize the IMU.
- virtual bool **IsActive** ()
Returns true if sensor generation is active.
- void **SetReferencePose** ()
Sets the current pose as the IMU reference pose.

Protected Member Functions

- virtual void **Fini** ()
Finalize the sensor.
- void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)
Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.
- virtual bool **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.103.1 Detailed Description

An IMU sensor.

10.103.2 Constructor & Destructor Documentation

10.103.2.1 gazebo::sensors::ImuSensor::ImuSensor ()

Constructor.

10.103.2.2 virtual gazebo::sensors::ImuSensor::~~ImuSensor () [virtual]

Destructor.

10.103.3 Member Function Documentation

10.103.3.1 virtual void gazebo::sensors::ImuSensor::Fini () [protected],[virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 912).

10.103.3.2 math::Vector3 gazebo::sensors::ImuSensor::GetAngularVelocity () const

Returns the angular velocity.

Returns

Angular velocity.

10.103.3.3 msgs::IMU gazebo::sensors::ImuSensor::GetImuMessage () const

Returns the imu message.

Returns

Imu message.

10.103.3.4 math::Vector3 gazebo::sensors::ImuSensor::GetLinearAcceleration () const

Returns the imu linear acceleration.

Returns

Linear acceleration.

10.103.3.5 `math::Quaternion gazebo::sensors::ImuSensor::GetOrientation () const`

get orientation of the IMU relative to the reference pose

Returns

returns the orientation quaternion of the IMU relative to the imu reference pose.

10.103.3.6 `virtual void gazebo::sensors::ImuSensor::Init () [virtual]`

Initialize the IMU.

Reimplemented from `gazebo::sensors::Sensor` (p.915).

10.103.3.7 `virtual bool gazebo::sensors::ImuSensor::IsActive () [virtual]`

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from `gazebo::sensors::Sensor` (p.915).

10.103.3.8 `void gazebo::sensors::ImuSensor::Load (const std::string & _worldName, sdf::ElementPtr _sdf) [protected], [virtual]`

Load the sensor with SDF parameters.

Parameters

<code>in</code>	<code>_sdf</code>	SDF <code>Sensor</code> (p.907) parameters.
<code>in</code>	<code>_worldName</code>	Name of world to load from.

Reimplemented from `gazebo::sensors::Sensor` (p.915).

10.103.3.9 `virtual void gazebo::sensors::ImuSensor::Load (const std::string & _worldName) [protected], [virtual]`

Load the sensor with default parameters.

Parameters

<code>in</code>	<code>_worldName</code>	Name of world to load from.
-----------------	-------------------------	-----------------------------

Reimplemented from `gazebo::sensors::Sensor` (p.915).

10.103.3.10 `void gazebo::sensors::ImuSensor::SetReferencePose ()`

Sets the current pose as the IMU reference pose.

10.103.3.11 `virtual bool gazebo::sensors::ImuSensor::UpdateImpl (bool)` [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

<code>in</code>	<code>_force</code>	True if update is forced, false if not
-----------------	---------------------	--

Returns

True if the sensor was updated.

Reimplemented from `gazebo::sensors::Sensor` (p. 917).

The documentation for this class was generated from the following file:

- `ImuSensor.hh`

10.104 gazebo::physics::Inertial Class Reference

A class for inertial information about a link.

```
#include <physics/physics.hh>
```

Public Member Functions

- **Inertial** ()
Default Constructor.
- **Inertial** (double `_mass`)
Constructor.
- **Inertial** (const **Inertial** &`_inertial`)
Copy constructor.
- virtual `~Inertial` ()
Destructor.
- const `math::Vector3` & **GetCoG** () const
Get the center of gravity.
- **Inertial GetInertial** (const `math::Pose` &`_frameOffset`) const
*Get equivalent Inertia values with the **Link** (p. 595) frame offset, while holding the Pose of CoG constant in the world frame.*
- double **GetIXX** () const
Get IXX.
- double **GetIXY** () const
Get IXY.
- double **GetIXZ** () const
Get IXZ.

- double **GetIYY** () const
Get IYY.
- double **GetIYZ** () const
Get IYZ.
- double **GetIZZ** () const
Get IZZ.
- double **GetMass** () const
Get the mass.
- **math::Matrix3 GetMOI** (const **math::Pose** &_pose) const
*Get the equivalent inertia from a point in local **Link** (p. 595) frame. If you specify **GetMOI(this->GetPose())** (p. 534), you should get back the Moment of Inertia (MOI) exactly as specified in the SDF.*
- **math::Matrix3 GetMOI** () const
returns Moments of Inertia as a Matrix3
- const **math::Pose GetPose** () const
*Get the pose about which the mass and inertia matrix is specified in the **Link** (p. 595) frame.*
- **math::Vector3 GetPrincipalMoments** () const
Get the principal moments of inertia (Ixx, Iyy, Izz).
- **math::Vector3 GetProductsofInertia** () const
Get the products of inertia (Ixy, Ixz, Iyz).
- void **Load** (sdf::ElementPtr _sdf)
Load from SDF values.
- **Inertial operator+** (const **Inertial** &_inertial) const
Addition operator.
- const **Inertial** & **operator+=** (const **Inertial** &_inertial)
Addition equal operator.
- **Inertial** & **operator=** (const **Inertial** &_inertial)
Equal operator.
- void **ProcessMsg** (const msgs::Inertial &_msg)
Update parameters from a message.
- void **Reset** ()
Reset all the mass properties.
- void **Rotate** (const **math::Quaternion** &_rot)
Rotate this mass.
- void **SetCoG** (double _cx, double _cy, double _cz)
Set the center of gravity.
- void **SetCoG** (const **math::Vector3** &_center)
Set the center of gravity.
- void **SetCoG** (double _cx, double _cy, double _cz, double _rx, double _ry, double _rz)
*Set the center of gravity and rotation offset of inertial coordinate frame relative to **Link** (p. 595) frame.*
- void **SetCoG** (const **math::Pose** &_c)
Set the center of gravity.
- void **SetInertiaMatrix** (double _ixx, double _iyy, double _izz, double _ixy, double _ixz, double iyz)
Set the mass matrix.
- void **SetIXX** (double _v)
Set IXX.
- void **SetIXY** (double _v)
Set IXY.

- void **SetIXZ** (double _v)
Set IXZ.
- void **SetIYY** (double _v)
Set IYY.
- void **SetIYZ** (double _v)
Set IYZ.
- void **SetIZZ** (double _v)
Set IZZ.
- void **SetMass** (double m)
Set the mass.
- void **SetMOI** (const **math::Matrix3** &_moi)
Sets Moments of Inertia (MOI) from a Matrix3.
- void **UpdateParameters** (sdf::ElementPtr _sdf)
update the parameters using new sdf values.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::Inertial** &_inertial)
Output operator.

10.104.1 Detailed Description

A class for inertial information about a link.

10.104.2 Constructor & Destructor Documentation

10.104.2.1 gazebo::physics::Inertial::Inertial ()

Default Constructor.

10.104.2.2 gazebo::physics::Inertial::Inertial (double _mass) [explicit]

Constructor.

Parameters

<code>in</code>	<code>_mass</code>	Mass value in kg if using metric.
-----------------	--------------------	-----------------------------------

10.104.2.3 gazebo::physics::Inertial::Inertial (const Inertial & _inertial)

Copy constructor.

Parameters

<code>in</code>	<code>_inertial</code>	Inertial (p. 529) element to copy
-----------------	------------------------	--

10.104.2.4 `virtual gazebo::physics::Inertial::~~Inertial () [virtual]`

Destructor.

10.104.3 Member Function Documentation

10.104.3.1 `const math::Vector3& gazebo::physics::Inertial::GetCoG () const [inline]`

Get the center of gravity.

Returns

The center of gravity.

10.104.3.2 `Inertial gazebo::physics::Inertial::GetInertial (const math::Pose & _frameOffset) const`

Get equivalent Inertia values with the **Link** (p. 595) frame offset, while holding the Pose of CoG constant in the world frame.

Parameters

<code>in</code>	<code>_frameOffset</code>	amount to offset the Link (p. 595) frame by, this is a transform defined in the Link (p. 595) frame.
-----------------	---------------------------	--

Returns

Inertial (p. 529) parameters with the shifted frame.

10.104.3.3 `double gazebo::physics::Inertial::GetIXX () const`

Get IXX.

Returns

IXX value

10.104.3.4 `double gazebo::physics::Inertial::GetIXY () const`

Get IXY.

Returns

IXY value

10.104.3.5 `double gazebo::physics::Inertial::GetIXZ () const`

Get IXZ.

Returns

IXZ value

10.104.3.6 `double gazebo::physics::Inertial::GetYI () const`

Get IYY.

Returns

IYY value

10.104.3.7 `double gazebo::physics::Inertial::GetIZ () const`

Get IXZ.

Returns

IYZ value

10.104.3.8 `double gazebo::physics::Inertial::GetIZZ () const`

Get IZZ.

Returns

IZZ value

10.104.3.9 `double gazebo::physics::Inertial::GetMass () const`

Get the mass.

10.104.3.10 `math::Matrix3 gazebo::physics::Inertial::GetMOI (const math::Pose & _pose) const`

Get the equivalent inertia from a point in local **Link** (p. 595) frame. If you specify `GetMOI(this->GetPose())` (p. 534), you should get back the Moment of Inertia (MOI) exactly as specified in the SDF.

If `_pose` is different from pose of the **Inertial** (p. 529) block, then the MOI is rotated accordingly, and contributions from changes in MOI location due to point mass is added to the final MOI.

Parameters

<code>in</code>	<code>_pose</code>	location in Link (p. 595) local frame
-----------------	--------------------	--

Returns

equivalent inertia at `_pose`

10.104.3.11 `math::Matrix3 gazebo::physics::Inertial::GetMOI () const`

returns Moments of Inertia as a Matrix3

Returns

Moments of Inertia as a Matrix3

10.104.3.12 `const math::Pose gazebo::physics::Inertial::GetPose () const [inline]`

Get the pose about which the mass and inertia matrix is specified in the **Link** (p. 595) frame.

Returns

The inertial pose.

10.104.3.13 `math::Vector3 gazebo::physics::Inertial::GetPrincipalMoments () const`

Get the principal moments of inertia (Ixx, Iyy, Izz).

Returns

The principal moments.

10.104.3.14 `math::Vector3 gazebo::physics::Inertial::GetProductsofInertia () const`

Get the products of inertia (Ixy, Ixz, Iyz).

Returns

The products of inertia.

10.104.3.15 `void gazebo::physics::Inertial::Load (sdf::ElementPtr _sdf)`

Load from SDF values.

Parameters

<code>in</code>	<code>_sdf</code>	SDF value to load from.
-----------------	-------------------	-------------------------

10.104.3.16 `Inertial gazebo::physics::Inertial::operator+ (const Inertial & _inertial) const`

Addition operator.

Assuming both CG and Moment of Inertia (MOI) are defined in the same reference **Link** (p. 595) frame. New CG is computed from masses and perspective offsets, and both MOI contributions relocated to the new cog.

Parameters

<code>in</code>	<code>_inertial</code>	Inertial (p. 529) to add.
-----------------	------------------------	----------------------------------

Returns

The result of the addition.

10.104.3.17 `const Inertial& gazebo::physics::Inertial::operator+=(const Inertial & _inertial)`

Addition equal operator.

Parameters

<code>in</code>	<code>_inertial</code>	Inertial (p. 529) to add.
-----------------	------------------------	----------------------------------

Returns

Reference to this object.

10.104.3.18 `Inertial& gazebo::physics::Inertial::operator=(const Inertial & _inertial)`

Equal operator.

Parameters

<code>in</code>	<code>_inertial</code>	Inertial (p. 529) to copy.
-----------------	------------------------	-----------------------------------

Returns

Reference to this object.

10.104.3.19 `void gazebo::physics::Inertial::ProcessMsg (const msgs::Inertial & _msg)`

Update parameters from a message.

Parameters

<code>in</code>	<code>_msg</code>	Message to read
-----------------	-------------------	-----------------

10.104.3.20 `void gazebo::physics::Inertial::Reset ()`

Reset all the mass properties.

10.104.3.21 `void gazebo::physics::Inertial::Rotate (const math::Quaternion & _rot)`

Rotate this mass.

Parameters

<code>in</code>	<code>_rot</code>	Rotation amount.
-----------------	-------------------	------------------

10.104.3.22 `void gazebo::physics::Inertial::SetCoG (double _cx, double _cy, double _cz)`

Set the center of gravity.

Parameters

in	<code>_cx</code>	X position.
in	<code>_cy</code>	Y position.
in	<code>_cz</code>	Z position.

10.104.3.23 `void gazebo::physics::Inertial::SetCoG (const math::Vector3 & _center)`

Set the center of gravity.

Parameters

in	<code>_center</code>	Center of the gravity.
----	----------------------	------------------------

10.104.3.24 `void gazebo::physics::Inertial::SetCoG (double _cx, double _cy, double _cz, double _rx, double _ry, double _rz)`

Set the center of gravity and rotation offset of inertial coordinate frame relative to **Link** (p. 595) frame.

Parameters

in	<code>_cx</code>	Center offset in x-direction in Link (p. 595) frame
in	<code>_cy</code>	Center offset in y-direction in Link (p. 595) frame
in	<code>_cz</code>	Center offset in z-direction in Link (p. 595) frame
in	<code>_rx</code>	Roll angle offset of inertial coordinate frame.
in	<code>_ry</code>	Pitch angle offset of inertial coordinate frame.
in	<code>_rz</code>	Yaw angle offset of inertial coordinate frame.

10.104.3.25 `void gazebo::physics::Inertial::SetCoG (const math::Pose & _c)`

Set the center of gravity.

Parameters

in	<code>_c</code>	Transform to center of gravity.
----	-----------------	---------------------------------

10.104.3.26 `void gazebo::physics::Inertial::SetInertiaMatrix (double _ixx, double _iyy, double _izz, double _ixy, double _ixz, double _iyz)`

Set the mass matrix.

Parameters

in	<code>_ixx</code>	X second moment of inertia (MOI) about x axis.
in	<code>_iyy</code>	Y second moment of inertia about y axis.
in	<code>_izz</code>	Z second moment of inertia about z axis.
in	<code>_ixy</code>	XY inertia.

<code>in</code>	<code>_ixz</code>	XZ inertia.
<code>in</code>	<code>_jyz</code>	YZ inertia.

10.104.3.27 `void gazebo::physics::Inertial::SetIXX (double _v)`

Set IXX.

Parameters

<code>in</code>	<code>_v</code>	IXX value
-----------------	-----------------	-----------

10.104.3.28 `void gazebo::physics::Inertial::SetIXY (double _v)`

Set IXY.

Parameters

<code>in</code>	<code>_v</code>	IXY value
-----------------	-----------------	-----------

10.104.3.29 `void gazebo::physics::Inertial::SetIXZ (double _v)`

Set IXZ.

Parameters

<code>in</code>	<code>_v</code>	IXZ value
-----------------	-----------------	-----------

10.104.3.30 `void gazebo::physics::Inertial::SetIYY (double _v)`

Set IYY.

Parameters

<code>in</code>	<code>_v</code>	IYY value
-----------------	-----------------	-----------

10.104.3.31 `void gazebo::physics::Inertial::SetIYZ (double _v)`

Set IYZ.

Parameters

<code>in</code>	<code>_v</code>	IXX value
-----------------	-----------------	-----------

10.104.3.32 `void gazebo::physics::Inertial::SetIZZ (double _v)`

Set IZZ.

Parameters

in	<i>_v</i>	IZZ value
----	-----------	-----------

10.104.3.33 void gazebo::physics::Inertial::SetMass (double *m*)

Set the mass.

10.104.3.34 void gazebo::physics::Inertial::SetMOI (const math::Matrix3 & *moi*)

Sets Moments of Inertia (MOI) from a Matrix3.

Parameters

in	<i>Moments</i>	of Inertia as a Matrix3
----	----------------	-------------------------

10.104.3.35 void gazebo::physics::Inertial::UpdateParameters (sdf::ElementPtr *sdf*)

update the parameters using new sdf values.

Parameters

in	<i>_sdf</i>	Update values from.
----	-------------	---------------------

10.104.4 Friends And Related Function Documentation

10.104.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::physics::Inertial & *_inertial*) [friend]

Output operator.

Parameters

in	<i>_out</i>	Output stream.
in	<i>_inertial</i>	Inertial (p. 529) object to output.

The documentation for this class was generated from the following file:

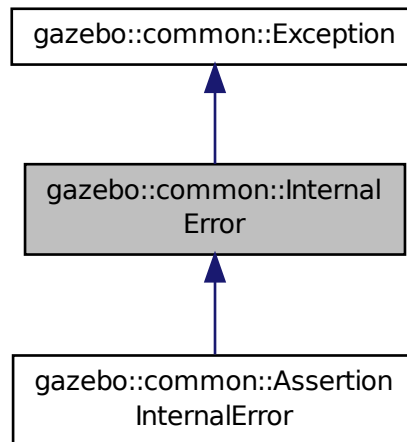
- **Inertial.hh**

10.105 gazebo::common::InternalError Class Reference

Class for generating Internal Gazebo Errors: those errors which should never happen and represent programming bugs.

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::common::InternalError:



Public Member Functions

- **InternalError** ()
Constructor.
- **InternalError** (const char *_file, int _line, const std::string &_msg)
Default constructor.
- virtual \sim **InternalError** ()
Destructor.

10.105.1 Detailed Description

Class for generating Internal Gazebo Errors: those errors which should never happen and represent programming bugs.

10.105.2 Constructor & Destructor Documentation

10.105.2.1 gazebo::common::InternalError::InternalError ()

Constructor.

10.105.2.2 gazebo::common::InternalError::InternalError (const char * _file, int _line, const std::string & _msg)

Default constructor.

Parameters

in	<code>_file</code>	File name
in	<code>_line</code>	Line number where the error occurred
in	<code>_msg</code>	Error message

10.105.2.3 virtual gazebo::common::InternalError::~InternalError () [virtual]

Destructor.

The documentation for this class was generated from the following file:

- **Exception.hh**

10.106 gazebo::transport::IOManager Class Reference

Manages boost::asio IO.

```
#include <transport/transport.hh>
```

Public Member Functions

- **IOManager** ()
Constructor.
- **~IOManager** ()
Destructor.
- void **DecCount** ()
Decrement the event count by 1.
- unsigned int **GetCount** () const
Get the event count.
- boost::asio::io_service & **GetIO** ()
Get handle to boost::asio IO service.
- void **IncCount** ()
Increment the event count by 1.
- void **Stop** ()
Stop the IO service.

10.106.1 Detailed Description

Manages boost::asio IO.

10.106.2 Constructor & Destructor Documentation

10.106.2.1 gazebo::transport::IOManager::IOManager ()

Constructor.

10.106.2.2 gazebo::transport::IOManager::~~IOManager ()

Destructor.

10.106.3 Member Function Documentation

10.106.3.1 void gazebo::transport::IOManager::DecCount ()

Decrement the event count by 1.

10.106.3.2 unsigned int gazebo::transport::IOManager::GetCount () const

Get the event count.

Returns

The event count

10.106.3.3 boost::asio::io_service& gazebo::transport::IOManager::GetIO ()

Get handle to boost::asio IO service.

Returns

Handle to boost::asio IO service

10.106.3.4 void gazebo::transport::IOManager::IncCount ()

Increment the event count by 1.

10.106.3.5 void gazebo::transport::IOManager::Stop ()

Stop the IO service.

The documentation for this class was generated from the following file:

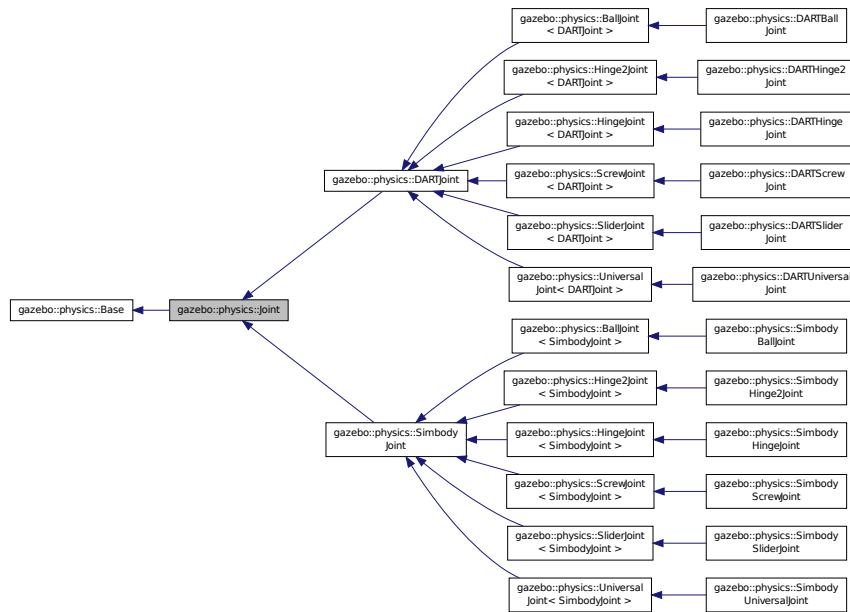
- **IOManager.hh**

10.107 gazebo::physics::Joint Class Reference

Base (p. 168) class for all joints.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::Joint:



Public Types

- enum **Attribute** {
FUDGE_FACTOR, **SUSPENSION_ERP**, **SUSPENSION_CFM**, **STOP_ERP**,
STOP_CFM, **ERP**, **CFM**, **FMAX**,
VEL, **HI_STOP**, **LO_STOP** }

Joint (p. 541) attribute types.

Public Member Functions

- **Joint** (**BasePtr** _parent)
Constructor.
- virtual **~Joint** ()
Destructor.
- virtual void **ApplyDamping** () **GAZEBO_DEPRECATED(3.0)**
Callback to apply damping force to joint.
- virtual void **ApplyStiffnessDamping** ()
Callback to apply spring stiffness and viscous damping effects to joint.
- virtual bool **AreConnected** (**LinkPtr** _one, **LinkPtr** _two) const =0
Determines if the two bodies are connected by a joint.
- virtual void **Attach** (**LinkPtr** _parent, **LinkPtr** _child)
Attach the two bodies with this joint.
- virtual void **CacheForceTorque** ()
*Cache **Joint** (p. 541) Force Torque Values if necessary for physics engine.*
- double **CheckAndTruncateForce** (unsigned int _index, double _effort)

check if the force against velocityLimit and effortLimit, truncate if necessary.

- template<typename T >
event::ConnectionPtr ConnectJointUpdate (T _subscriber)
Connect a boost::slot the the joint update signal.
- virtual void **Detach** ()
Detach this joint from all links.
- void **DisconnectJointUpdate** (event::ConnectionPtr &_conn)
Disconnect a boost::slot the the joint update signal.
- void **FillMsg** (msgs::Joint &_msg)
Fill a joint message.
- virtual void **Fini** ()
Finalize the object.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const =0
Get the anchor point.
- **math::Pose GetAnchorErrorPose** () const
Get pose offset between anchor pose on child and parent, expressed in the parent link frame.
- **math::Angle GetAngle** (unsigned int _index) const
Get the angle of rotation of an axis(index)
- virtual unsigned int **GetAngleCount** () const =0
Get the angle count.
- virtual double **GetAttribute** (const std::string &_key, unsigned int _index) **GAZEBO_DEPRECATED(3.0)=0**
Get a non-generic parameter for the joint.
- **math::Quaternion GetAxisFrame** (unsigned int _index) const
Get orientation of reference frame for specified axis, relative to world frame.
- **LinkPtr GetChild** () const
Get the child link.
- double **GetDamping** (unsigned int _index)
Returns the current joint damping coefficient.
- double **GetDampingCoefficient** () const **GAZEBO_DEPRECATED(3.0)**
Get damping coefficient of this joint Depreated, use GetDamping(_index) instead.
- virtual double **GetEffortLimit** (unsigned int _index)
Get the effort limit on axis(index).
- virtual double **GetForce** (unsigned int _index)
- virtual **JointWrench GetForceTorque** (unsigned int _index)=0
get internal force and torque values at a joint.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const =0
Get the axis of rotation in global coordinate frame.
- virtual **math::Angle GetHighStop** (unsigned int _index)=0
Get the high stop of an axis(index).
- double **GetInertiaRatio** (const unsigned int _index) const
Computes moment of inertia (MOI) across a specified joint axis.
- double **GetInertiaRatio** (const **math::Vector3** &_axis) const
Computes moment of inertia (MOI) across an arbitrary axis specified in the world frame.
- **math::Pose GetInitialAnchorPose** () const
Get initial Anchor Pose specified by model <joint><pose>...</pose></joint>
- virtual **LinkPtr GetJointLink** (unsigned int _index) const =0
Get the link to which the joint is attached according the _index.

- virtual **math::Vector3 GetLinkForce** (unsigned int _index) const =0
*Get the forces applied to the center of mass of a **physics::Link** (p. 595) due to the existence of this **Joint** (p. 541).*
- virtual **math::Vector3 GetLinkTorque** (unsigned int _index) const =0
*Get the torque applied to the center of mass of a **physics::Link** (p. 595) due to the existence of this **Joint** (p. 541).*
- **math::Vector3 GetLocalAxis** (unsigned int _index) const
Get the axis of rotation.
- **math::Angle GetLowerLimit** (unsigned int _index) const
*: get the joint upper limit (replaces *GetLowStop* and *GetHighStop*)*
- virtual **math::Angle GetLowStop** (unsigned int _index)=0
Get the low stop of an axis(index).
- virtual double **GetMaxForce** (unsigned int _index)=0
Get the max allowed force of an axis(index).
- virtual double **GetParam** (const std::string &_key, unsigned int _index)=0
Get a non-generic parameter for the joint.
- **LinkPtr GetParent** () const
Get the parent link.
- **math::Pose GetParentWorldPose** () const
Get anchor pose on parent link relative to world frame.
- double **GetSpringReferencePosition** (unsigned int _index) const
Get joint spring reference position.
- double **GetStiffness** (unsigned int _index)
Returns the current joint spring stiffness coefficient.
- double **GetStopDissipation** (unsigned int _index) const
Get joint stop dissipation.
- double **GetStopStiffness** (unsigned int _index) const
Get joint stop stiffness.
- **math::Angle GetUpperLimit** (unsigned int _index) const
*: get the joint lower limit (replacee *GetLowStop* and *GetHighStop*)*
- virtual double **GetVelocity** (unsigned int _index) const =0
Get the rotation rate of an axis(index)
- virtual double **GetVelocityLimit** (unsigned int _index)
Get the velocity limit on axis(index).
- double **GetWorldEnergyPotentialSpring** (unsigned int _index) const
Returns this joint's spring potential energy, based on the reference position of the spring.
- **math::Pose GetWorldPose** () const
Get pose of joint frame relative to world frame.
- virtual void **Init** ()
Initialize a joint.
- void **Load** (**LinkPtr** _parent, **LinkPtr** _child, const **math::Pose** &_pose)
*Set pose, parent and child links of a **physics::Joint** (p. 541).*
- virtual void **Load** (sdf::ElementPtr _sdf)
*Load **physics::Joint** (p. 541) from a SDF sdf::Element.*
- virtual void **Reset** ()
Reset the joint.
- virtual void **SetAnchor** (unsigned int _index, const **math::Vector3** &_anchor)=0
Set the anchor point.
- void **SetAngle** (unsigned int _index, **math::Angle** _angle)

If the **Joint** (p. 541) is static, Gazebo stores the state of this **Joint** (p. 541) as a scalar inside the **Joint** (p. 541) class, so this call will NOT move the joint dynamically for a static **Model** (p. 678).

- virtual void **SetAttribute** (const std::string &_key, unsigned int _index, const boost::any &_value) **GAZEBO_DEPRECATED(3.0)=0**
Set a non-generic parameter for the joint.
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)=0
Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetDamping** (unsigned int _index, double _damping)=0
Set the joint damping.
- virtual void **SetEffortLimit** (unsigned int _index, double _effort)
Set the effort limit on a joint axis.
- virtual void **SetForce** (unsigned int _index, double _effort)=0
*Set the force applied to this **physics::Joint** (p. 541).*
- virtual bool **SetHighStop** (unsigned int _index, const **math::Angle** &_angle)
Set the high stop of an axis(index).
- void **SetLowerLimit** (unsigned int _index, **math::Angle** _limit)
: set the joint upper limit (replaces SetLowStop and SetHighStop)
- virtual bool **SetLowStop** (unsigned int _index, const **math::Angle** &_angle)
Set the low stop of an axis(index).
- virtual void **SetMaxForce** (unsigned int _index, double _force)=0
Set the max allowed force of an axis(index).
- void **SetModel** (**ModelPtr** _model)
Set the model this joint belongs too.
- virtual bool **SetParam** (const std::string &_key, unsigned int _index, const boost::any &_value)=0
Set a non-generic parameter for the joint.
- virtual void **SetProvideFeedback** (bool _enable)
Set whether the joint should generate feedback.
- void **SetState** (const **JointState** &_state)
Set the joint state.
- virtual void **SetStiffness** (unsigned int _index, double _stiffness)=0
Set the joint spring stiffness.
- virtual void **SetStiffnessDamping** (unsigned int _index, double _stiffness, double _damping, double _reference=0)=0
Set the joint spring stiffness.
- void **SetStopDissipation** (unsigned int _index, double _dissipation)
Set joint stop dissipation.
- void **SetStopStiffness** (unsigned int _index, double _stiffness)
Set joint stop stiffness.
- void **SetUpperLimit** (unsigned int _index, **math::Angle** _limit)
: set the joint lower limit (replaces GetLowStop and GetHighStop)
- virtual void **SetVelocity** (unsigned int _index, double _vel)=0
Set the velocity of an axis(index).
- void **Update** ()
Update the joint.
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)
Update the parameters using new sdf values.

Protected Member Functions

- virtual **math::Angle GetAngleImpl** (unsigned int `_index`) const =0
Get the angle of an axis helper function.

Protected Attributes

- **LinkPtr anchorLink**
Anchor link.
- **math::Vector3 anchorPos**
Anchor pose.
- **math::Pose anchorPose**
Anchor pose specified in SDF `<joint><pose>` tag.
- **gazebo::event::ConnectionPtr applyDamping**
apply damping for adding viscous damping forces on updates
- bool **axisParentModelFrame** [2]
Flags that are set to true if an axis value is expressed in the parent model frame.
- **LinkPtr childLink**
The first link this joint connects to.
- double **dampingCoefficient**
*joint dissipationCoefficient *Deprecated*: not used, replaced by dissipationCoefficient array*
- double **dissipationCoefficient** [2]
*joint viscous damping coefficient *Replaces dampingCoefficient**
- double **effortLimit** [2]
*Store *Joint* (p. 541) effort limit as specified in SDF.*
- **math::Angle lowerLimit** [2]
*Store *Joint* (p. 541) position lower limit as specified in SDF.*
- **ModelPtr model**
Pointer to the parent model.
- **math::Pose parentAnchorPose**
Anchor pose relative to parent link frame.
- **LinkPtr parentLink**
The second link this joint connects to.
- bool **provideFeedback**
Provide Feedback data for contact forces.
- double **springReferencePosition** [2]
joint spring reference (zero load) position
- double **stiffnessCoefficient** [2]
joint stiffnessCoefficient
- **math::Angle upperLimit** [2]
*Store *Joint* (p. 541) position upper limit as specified in SDF.*
- bool **useCFMDamping**
*option to use implicit damping *Deprecated*, pushing this flag into individual physics engine, for example: *ODEJoint::useImplicitSpringDamper*.*
- double **velocityLimit** [2]
*Store *Joint* (p. 541) velocity limit as specified in SDF.*
- **JointWrench wrench**
*Cache *Joint* (p. 541) force torque values in case physics engine clears them at the end of update step.*

10.107.1 Detailed Description

Base (p. 168) class for all joints.

10.107.2 Member Enumeration Documentation

10.107.2.1 enum gazebo::physics::Joint::Attribute

Joint (p. 541) attribute types.

Enumerator:

- FUDGE_FACTOR** Fudge factor.
- SUSPENSION_ERP** Suspension error reduction parameter.
- SUSPENSION_CFM** Suspension constraint force mixing.
- STOP_ERP** Stop limit error reduction parameter.
- STOP_CFM** Stop limit constraint force mixing.
- ERP** Error reduction parameter.
- CFM** Constraint force mixing.
- FMAX** Maximum force.
- VEL** Velocity.
- HI_STOP** High stop angle.
- LO_STOP** Low stop angle.

10.107.3 Constructor & Destructor Documentation

10.107.3.1 gazebo::physics::Joint::Joint (BasePtr _parent) [explicit]

Constructor.

Parameters

in	Joint (p. 541)	parent
----	-----------------------	--------

10.107.3.2 virtual gazebo::physics::Joint::~~Joint () [virtual]

Destructor.

10.107.4 Member Function Documentation

10.107.4.1 virtual void gazebo::physics::Joint::ApplyDamping () [virtual]

Callback to apply damping force to joint.

Deprecated by ApplySpringStiffnessDamping.

Reimplemented in **gazebo::physics::DARTJoint** (p. 330).

10.107.4.2 `virtual void gazebo::physics::Joint::ApplyStiffnessDamping () [virtual]`

Callback to apply spring stiffness and viscous damping effects to joint.

: rename to `ApplySpringStiffnessDamping()`

10.107.4.3 `virtual bool gazebo::physics::Joint::AreConnected (LinkPtr _one, LinkPtr _two) const [pure virtual]`

Determines if the two bodies are connected by a joint.

Parameters

<code>in</code>	<code>_one</code>	First link.
<code>in</code>	<code>_two</code>	Second link.

Returns

True if the two links are connected by a joint.

Implemented in `gazebo::physics::DARTJoint` (p. 330), and `gazebo::physics::SimbodyJoint` (p. 963).

10.107.4.4 `virtual void gazebo::physics::Joint::Attach (LinkPtr _parent, LinkPtr _child) [virtual]`

Attach the two bodies with this joint.

Parameters

<code>in</code>	<code>_parent</code>	Parent link.
<code>in</code>	<code>_child</code>	Child link.

Reimplemented in `gazebo::physics::DARTJoint` (p. 330).

10.107.4.5 `virtual void gazebo::physics::Joint::CacheForceTorque () [inline],[virtual]`

Cache **Joint** (p. 541) Force Torque Values if necessary for physics engine.

Reimplemented in `gazebo::physics::SimbodyJoint` (p. 963).

10.107.4.6 `double gazebo::physics::Joint::CheckAndTruncateForce (unsigned int _index, double _effort)`

check if the force against velocityLimit and effortLimit, truncate if necessary.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_effort</code>	Force value.

Returns

truncated effort

10.107.4.7 `template<typename T > event::ConnectionPtr gazebo::physics::Joint::ConnectJointUpdate (T _subscriber)`
`[inline]`

Connect a boost::slot the the joint update signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	Callback for the connection.
-----------------	---------------------------------	------------------------------

Returns

Connection pointer, which must be kept in scope.

10.107.4.8 `virtual void gazebo::physics::Joint::Detach ()` `[virtual]`

Detach this joint from all links.

Reimplemented in `gazebo::physics::DARTJoint` (p. 331), and `gazebo::physics::SimbodyJoint` (p. 963).

10.107.4.9 `void gazebo::physics::Joint::DisconnectJointUpdate (event::ConnectionPtr & _conn)` `[inline]`

Disconnect a boost::slot the the joint update signal.

Parameters

<code>in</code>	<code><i>_conn</i></code>	Connection to disconnect.
-----------------	---------------------------	---------------------------

10.107.4.10 `void gazebo::physics::Joint::FillMsg (msgs::Joint & _msg)`

Fill a joint message.

Parameters

<code>out</code>	<code><i>_msg</i></code>	Message to fill with this joint's properties.
------------------	--------------------------	---

10.107.4.11 `virtual void gazebo::physics::Joint::Fini ()` `[virtual]`

Finalize the object.

Reimplemented from `gazebo::physics::Base` (p. 174).

10.107.4.12 `virtual math::Vector3 gazebo::physics::Joint::GetAnchor (unsigned int _index) const` `[pure virtual]`

Get the anchor point.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Anchor value for the axis.

Implemented in `gazebo::physics::SimbodyJoint` (p. 963), `gazebo::physics::SimbodyUniversalJoint` (p. 1018), `gazebo::physics::SimbodyHinge2Joint` (p. 952), `gazebo::physics::DARTHinge2Joint` (p. 319), `gazebo::physics::DARTHingeJoint` (p. 324), `gazebo::physics::SimbodyBallJoint` (p. 938), `gazebo::physics::DARTBallJoint` (p. 304), `gazebo::physics::DARTSliderJoint` (p. 373), `gazebo::physics::DARTUniversalJoint` (p. 380), and `gazebo::physics::DARTScrewJoint` (p. 366).

10.107.4.13 `math::Pose gazebo::physics::Joint::GetAnchorErrorPose () const`

Get pose offset between anchor pose on child and parent, expressed in the parent link frame.

This can be used to compute the bilateral constraint error.

Returns

Pose offset between anchor pose on child and parent, in parent link frame.

10.107.4.14 `math::Angle gazebo::physics::Joint::GetAngle (unsigned int _index) const`

Get the angle of rotation of an axis(index)

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Angle of the axis.

10.107.4.15 `virtual unsigned int gazebo::physics::Joint::GetAngleCount () const` `[pure virtual]`

Get the angle count.

Returns

The number of DOF for the joint.

Implemented in `gazebo::physics::DARTJoint` (p. 331), `gazebo::physics::UniversalJoint< DARTJoint >` (p. 1136), `gazebo::physics::UniversalJoint< SimbodyJoint >` (p. 1136), `gazebo::physics::BallJoint< DARTJoint >` (p. 168), `gazebo::physics::BallJoint< SimbodyJoint >` (p. 168), `gazebo::physics::SliderJoint< DARTJoint >` (p. 1045), `gazebo::physics::SliderJoint< SimbodyJoint >` (p. 1045), `gazebo::physics::Hinge2Joint< DARTJoint >` (p. 513), `gazebo::physics::Hinge2Joint< SimbodyJoint >` (p. 513), `gazebo::physics::ScrewJoint< DARTJoint >` (p. 898), `gazebo::physics::ScrewJoint< SimbodyJoint >` (p. 898), `gazebo::physics::HingeJoint< DARTJoint >` (p. 514), and `gazebo::physics::HingeJoint< SimbodyJoint >` (p. 514).

10.107.4.16 `virtual math::Angle gazebo::physics::Joint::GetAngleImpl (unsigned int _index) const` `[protected]`, `[pure virtual]`

Get the angle of an axis helper function.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Angle of the axis.

Implemented in `gazebo::physics::SimbodyScrewJoint` (p.1004), `gazebo::physics::SimbodyUniversalJoint` (p.1018), `gazebo::physics::SimbodyHinge2Joint` (p.953), `gazebo::physics::SimbodyHingeJoint` (p.957), `gazebo::physics::DARTScrewJoint` (p.366), `gazebo::physics::SimbodySliderJoint` (p.1012), `gazebo::physics::SimbodyBallJoint` (p.938), `gazebo::physics::DARTBallJoint` (p.304), `gazebo::physics::DARTHinge2Joint` (p.320), `gazebo::physics::DARTHingeJoint` (p.325), `gazebo::physics::DARTSliderJoint` (p.373), and `gazebo::physics::DARTUniversalJoint` (p.380).

10.107.4.17 `virtual double gazebo::physics::Joint::GetAttribute (const std::string & _key, unsigned int _index) [pure virtual]`

Get a non-generic parameter for the joint.

Deprecated by GetParam

Parameters

<code>in</code>	<code>_key</code>	String key.
<code>in</code>	<code>_index</code>	Index of the axis.

Implemented in `gazebo::physics::DARTJoint` (p.331), `gazebo::physics::SimbodyScrewJoint` (p.1004), and `gazebo::physics::SimbodyJoint` (p.964).

10.107.4.18 `math::Quaternion gazebo::physics::Joint::GetAxisFrame (unsigned int _index) const`

Get orientation of reference frame for specified axis, relative to world frame.

The value of axisParentModelFrame is used to determine the appropriate frame.

Parameters

<code>in</code>	<code>_index</code>	joint axis index.
-----------------	---------------------	-------------------

Returns

Orientation of axis frame relative to world frame.

10.107.4.19 `LinkPtr gazebo::physics::Joint::GetChild () const`

Get the child link.

Returns

Pointer to the child link.

10.107.4.20 `double gazebo::physics::Joint::GetDamping (unsigned int _index)`

Returns the current joint damping coefficient.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis to get, currently ignored, to be implemented.
-----------------	----------------------------	---

Returns

Joint (p. 541) viscous damping coefficient for this joint.

10.107.4.21 `double gazebo::physics::Joint::GetDampingCoefficient () const`

Get damping coefficient of this joint Depreated, use `GetDamping(_index)` instead.

Returns

viscous joint damping coefficient

10.107.4.22 `virtual double gazebo::physics::Joint::GetEffortLimit (unsigned int _index)` `[virtual]`

Get the effort limit on axis(*index*).

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of axis, where 0=first axis and 1=second axis
-----------------	----------------------------	---

Returns

Effort limit specified in SDF

10.107.4.23 `virtual double gazebo::physics::Joint::GetForce (unsigned int _index)` `[virtual]`

Todo : not yet implemented. Get external forces applied at this **Joint** (p. 541). Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

The force applied to an axis.

Reimplemented in `gazebo::physics::DARTJoint` (p. 332), and `gazebo::physics::SimbodyJoint` (p. 964).

10.107.4.24 `virtual JointWrench gazebo::physics::Joint::GetForceTorque (unsigned int _index) [pure virtual]`

get internal force and torque values at a joint.

The force and torque values are returned in a **JointWrench** (p. 581) data structure. Where **JointWrench.body1Force** (p. 582) contains the force applied by the parent **Link** (p. 595) on the **Joint** (p. 541) specified in the parent **Link** (p. 595) frame, and **JointWrench.body2Force** (p. 583) contains the force applied by the child **Link** (p. 595) on the **Joint** (p. 541) specified in the child **Link** (p. 595) frame. Note that this sign convention is opposite of the reaction forces of the **Joint** (p. 541) on the Links.

FIXME TODO: change name of this function to something like: GetNegatedForceTorqueInLinkFrame and make GetForceTorque call return non-negated reaction forces in perspective **Link** (p. 595) frames.

Note that for ODE you must set `<provide_feedback>true</provide_feedback>` in the joint sdf to use this.

Parameters

<code>in</code>	<code><i>_index</i></code>	Not used right now
-----------------	----------------------------	--------------------

Returns

The force and torque at the joint, see above for details on conventions.

Implemented in **gazebo::physics::SimbodyJoint** (p. 964), and **gazebo::physics::DARTJoint** (p. 332).

10.107.4.25 `virtual math::Vector3 gazebo::physics::Joint::GetGlobalAxis (unsigned int _index) const [pure virtual]`

Get the axis of rotation in global coordinate frame.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis to get.
-----------------	----------------------------	---------------------------

Returns

Axis value for the provided index.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 1005), **gazebo::physics::SimbodyUniversalJoint** (p. 1018), **gazebo::physics::SimbodyHinge2Joint** (p. 953), **gazebo::physics::SimbodyBallJoint** (p. 938), **gazebo::physics::SimbodyHingeJoint** (p. 958), **gazebo::physics::SimbodySliderJoint** (p. 1012), **gazebo::physics::DARTScrewJoint** (p. 367), **gazebo::physics::DARTHinge2Joint** (p. 320), **gazebo::physics::DARTHingeJoint** (p. 325), **gazebo::physics::DARTBallJoint** (p. 304), **gazebo::physics::DARTSliderJoint** (p. 373), and **gazebo::physics::DARTUniversalJoint** (p. 380).

10.107.4.26 `virtual math::Angle gazebo::physics::Joint::GetHighStop (unsigned int _index) [pure virtual]`

Get the high stop of an axis(index).

This function is replaced by `GetUpperLimit(unsigned int)`. If you are interested in getting the value of `dParamHiStop*`, use `GetAttribute(hi_stop, _index)`

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Angle of the high stop value.

Implemented in `gazebo::physics::SimbodyJoint` (p. 965), `gazebo::physics::DARTJoint` (p. 332), `gazebo::physics::DARTScrewJoint` (p. 367), `gazebo::physics::SimbodyBallJoint` (p. 938), `gazebo::physics::DARTBallJoint` (p. 304), and `gazebo::physics::SimbodyScrewJoint` (p. 1005).

10.107.4.27 `double gazebo::physics::Joint::GetInertiaRatio (const unsigned int _index) const`

Computes moment of inertia (MOI) across a specified joint axis.

The ratio is given in the form of `MOI_child / MOI_parent`. If `MOI_parent` is zero, this function will return 0. The inertia ratio for each joint axis indicates the sensitivity of the joint to actuation torques.

Parameters

<code>in</code>	<code><i>_index</i></code>	axis number about which MOI ratio is computed.
-----------------	----------------------------	--

Returns

ratio of child MOI to parent MOI.

10.107.4.28 `double gazebo::physics::Joint::GetInertiaRatio (const math::Vector3 & _axis) const`

Computes moment of inertia (MOI) across an arbitrary axis specified in the world frame.

The ratio is given in the form of `MOI_child / MOI_parent`. If `MOI_parent` is zero, this function will return 0. The moment of inertia ratio along constrained directions of a joint has an impact on the performance of Projected Gauss Seidel (PGS) iterative LCP methods.

Parameters

<code>in</code>	<code><i>_axis</i></code>	axis in world frame for which MOI ratio is computed.
-----------------	---------------------------	--

Returns

ratio of child MOI to parent MOI.

10.107.4.29 `math::Pose gazebo::physics::Joint::GetInitialAnchorPose () const`

Get initial Anchor Pose specified by model `<joint><pose>...</pose></joint>`

Returns

`Joint::anchorPose` (p. 566), initial joint anchor pose.

10.107.4.30 `virtual LinkPtr gazebo::physics::Joint::GetJointLink (unsigned int _index) const` [pure virtual]

Get the link to which the joint is attached according the `_index`.

Parameters

<code>in</code>	<code>_index</code>	Index of the link to retrieve.
-----------------	---------------------	--------------------------------

Returns

Pointer to the request link. NULL if the index was invalid.

Implemented in **gazebo::physics::DARTJoint** (p. 333), and **gazebo::physics::SimbodyJoint** (p. 965).

10.107.4.31 `virtual math::Vector3 gazebo::physics::Joint::GetLinkForce (unsigned int _index) const` [pure virtual]

Get the forces applied to the center of mass of a **physics::Link** (p. 595) due to the existence of this **Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code>index</code>	The index of the link(0 or 1).
-----------------	--------------------	--------------------------------

Returns

Force applied to the link.

Implemented in **gazebo::physics::DARTJoint** (p. 333), and **gazebo::physics::SimbodyJoint** (p. 965).

10.107.4.32 `virtual math::Vector3 gazebo::physics::Joint::GetLinkTorque (unsigned int _index) const` [pure virtual]

Get the torque applied to the center of mass of a **physics::Link** (p. 595) due to the existence of this **Joint** (p. 541).

Note that the unit of torque should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code>index</code>	The index of the link(0 or 1)
-----------------	--------------------	-------------------------------

Returns

Torque applied to the link.

Implemented in **gazebo::physics::DARTJoint** (p. 333), and **gazebo::physics::SimbodyJoint** (p. 965).

10.107.4.33 `math::Vector3 gazebo::physics::Joint::GetLocalAxis (unsigned int _index) const`

Get the axis of rotation.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to get.
-----------------	---------------------	---------------------------

Returns

Axis value for the provided index.

10.107.4.34 `math::Angle gazebo::physics::Joint::GetLowerLimit (unsigned int _index) const`

: get the joint upper limit (replaces GetLowStop and GetHighStop)

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Lower limit of the axis.

10.107.4.35 `virtual math::Angle gazebo::physics::Joint::GetLowStop (unsigned int _index) [pure virtual]`

Get the low stop of an axis(index).

This function is replaced by `GetLowerLimit(unsigned int)`. If you are interested in getting the value of `dParamHiStop*`, use `GetAttribute(hi_stop, _index)`

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Angle of the low stop value.

Implemented in `gazebo::physics::SimbodyJoint` (p. 966), `gazebo::physics::DARTJoint` (p. 334), `gazebo::physics::DARTScrewJoint` (p. 367), `gazebo::physics::SimbodyBallJoint` (p. 939), `gazebo::physics::DARTBallJoint` (p. 305), and `gazebo::physics::SimbodyScrewJoint` (p. 1005).

10.107.4.36 `virtual double gazebo::physics::Joint::GetMaxForce (unsigned int _index) [pure virtual]`

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

The maximum force.

Implemented in `gazebo::physics::SimbodyScrewJoint` (p. 1005), `gazebo::physics::DARTScrewJoint` (p. 367), `gazebo::physics::DARTHinge2Joint` (p. 320), `gazebo::physics::DARTHingeJoint` (p. 325), `gazebo::physics::SimbodyUniversalJoint` (p. 1018), `gazebo::physics::SimbodyHinge2Joint` (p. 953), `gazebo::physics::DARTSliderJoint` (p. 374), `gazebo::physics::DARTUniversalJoint` (p. 381), `gazebo::physics::SimbodyHingeJoint` (p. 958), `gazebo::physics::SimbodySliderJoint` (p. 1012), `gazebo::physics::SimbodyBallJoint` (p. 939), and `gazebo::physics::DARTBallJoint` (p. 305).

10.107.4.37 `virtual double gazebo::physics::Joint::GetParam (const std::string & _key, unsigned int _index) [pure virtual]`

Get a non-generic parameter for the joint.

Parameters

<code>in</code>	<code><i>_key</i></code>	String key.
<code>in</code>	<code><i>_index</i></code>	Index of the axis.

Implemented in **`gazebo::physics::DARTJoint`** (p. 334), **`gazebo::physics::SimbodyScrewJoint`** (p. 1006), and **`gazebo::physics::SimbodyJoint`** (p. 966).

10.107.4.38 `LinkPtr gazebo::physics::Joint::GetParent () const`

Get the parent link.

Returns

Pointer to the parent link.

Reimplemented from **`gazebo::physics::Base`** (p. 175).

10.107.4.39 `math::Pose gazebo::physics::Joint::GetParentWorldPose () const`

Get anchor pose on parent link relative to world frame.

When there is zero joint error, this should match the value returned by **`Joint::GetWorldPose()`** (p. 559) for the constrained degrees of freedom.

Returns

Anchor pose on parent link in world frame.

10.107.4.40 `double gazebo::physics::Joint::GetSpringReferencePosition (unsigned int _index) const`

Get joint spring reference position.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis to get.
-----------------	----------------------------	---------------------------

Returns

`Joint` (p. 541) spring reference position (in radians for angular joints).

10.107.4.41 `double gazebo::physics::Joint::GetStiffness (unsigned int _index)`

Returns the current joint spring stiffness coefficient.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to get, currently ignored, to be implemented.
-----------------	---------------------	---

Returns

Joint (p. 541) spring stiffness coefficient for this joint. : rename to GetSpringStiffness()

10.107.4.42 `double gazebo::physics::Joint::GetStopDissipation (unsigned int _index) const`

Get joint stop dissipation.

Parameters

<code>in</code>	<code>_index</code>	joint axis index.
-----------------	---------------------	-------------------

Returns

joint stop dissipation coefficient.

10.107.4.43 `double gazebo::physics::Joint::GetStopStiffness (unsigned int _index) const`

Get joint stop stiffness.

Parameters

<code>in</code>	<code>_index</code>	joint axis index.
-----------------	---------------------	-------------------

Returns

joint stop stiffness coefficient.

10.107.4.44 `math::Angle gazebo::physics::Joint::GetUpperLimit (unsigned int _index) const`

: get the joint lower limit (replacee GetLowStop and GetHighStop)

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Upper limit of the axis.

10.107.4.45 `virtual double gazebo::physics::Joint::GetVelocity (unsigned int _index) const` [pure virtual]

Get the rotation rate of an axis(index)

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

The rotaional velocity of the joint axis.

Implemented in **`gazebo::physics::SimbodyScrewJoint`** (p. 1007), **`gazebo::physics::DARTScrewJoint`** (p. 368), **`gazebo::physics::DARTHingeJoint`** (p. 326), **`gazebo::physics::SimbodyUniversalJoint`** (p. 1019), **`gazebo::physics::DARTHinge2Joint`** (p. 321), **`gazebo::physics::DARTSliderJoint`** (p. 374), **`gazebo::physics::SimbodyHinge2Joint`** (p. 953), **`gazebo::physics::DARTUniversalJoint`** (p. 381), **`gazebo::physics::SimbodyBallJoint`** (p. 939), **`gazebo::physics::SimbodyHingeJoint`** (p. 958), **`gazebo::physics::SimbodySliderJoint`** (p. 1012), and **`gazebo::physics::DARTBallJoint`** (p. 305).

10.107.4.46 `virtual double gazebo::physics::Joint::GetVelocityLimit (unsigned int _index) [virtual]`

Get the velocity limit on axis(index).

Parameters

<code>in</code>	<code>_index</code>	Index of axis, where 0=first axis and 1=second axis
-----------------	---------------------	---

Returns

Velocity limit specified in SDF

10.107.4.47 `double gazebo::physics::Joint::GetWorldEnergyPotentialSpring (unsigned int _index) const`

Returns this joint's spring potential energy, based on the reference position of the spring.

If using metric system, the unit of energy will be Joules.

Returns

this joint's spring potential energy,

10.107.4.48 `math::Pose gazebo::physics::Joint::GetWorldPose () const`

Get pose of joint frame relative to world frame.

Note that the joint frame is defined with a fixed offset from the child link frame.

Returns

Pose of joint frame relative to world frame.

10.107.4.49 `virtual void gazebo::physics::Joint::Init () [virtual]`

Initialize a joint.

Reimplemented from **`gazebo::physics::Base`** (p. 177).

Reimplemented in `gazebo::physics::ScrewJoint< DARTJoint >` (p. 898), `gazebo::physics::ScrewJoint< SimbodyJoint >` (p. 898), `gazebo::physics::UniversalJoint< DARTJoint >` (p. 1136), `gazebo::physics::UniversalJoint< SimbodyJoint >` (p. 1136), `gazebo::physics::BallJoint< DARTJoint >` (p. 168), `gazebo::physics::BallJoint< SimbodyJoint >` (p. 168), `gazebo::physics::HingeJoint< DARTJoint >` (p. 514), `gazebo::physics::HingeJoint< SimbodyJoint >` (p. 514), `gazebo::physics::DARTScrewJoint` (p. 369), `gazebo::physics::DARTJoint` (p. 334), `gazebo::physics::DARTHinge2Joint` (p. 321), `gazebo::physics::DARTHingeJoint` (p. 326), `gazebo::physics::DARTBallJoint` (p. 306), `gazebo::physics::DARTSliderJoint` (p. 374), and `gazebo::physics::DARTUniversalJoint` (p. 381).

10.107.4.50 `void gazebo::physics::Joint::Load (LinkPtr _parent, LinkPtr _child, const math::Pose & _pose)`

Set pose, parent and child links of a `physics::Joint` (p. 541).

Parameters

in	<code>_parent</code>	Parent link.
in	<code>_child</code>	Child link.
in	<code>_pose</code>	Pose containing <code>Joint</code> (p. 541) Anchor offset from child link.

10.107.4.51 `virtual void gazebo::physics::Joint::Load (sdf::ElementPtr _sdf) [virtual]`

Load `physics::Joint` (p. 541) from a SDF `sdf::Element`.

Parameters

in	<code>_sdf</code>	SDF values to load from.
----	-------------------	--------------------------

Reimplemented from `gazebo::physics::Base` (p. 177).

Reimplemented in `gazebo::physics::SimbodySliderJoint` (p. 1013), `gazebo::physics::BallJoint< DARTJoint >` (p. 168), `gazebo::physics::BallJoint< SimbodyJoint >` (p. 168), `gazebo::physics::UniversalJoint< DARTJoint >` (p. 1136), `gazebo::physics::UniversalJoint< SimbodyJoint >` (p. 1136), `gazebo::physics::Hinge2Joint< DARTJoint >` (p. 513), `gazebo::physics::Hinge2Joint< SimbodyJoint >` (p. 513), `gazebo::physics::ScrewJoint< DARTJoint >` (p. 898), `gazebo::physics::ScrewJoint< SimbodyJoint >` (p. 898), `gazebo::physics::HingeJoint< DARTJoint >` (p. 514), `gazebo::physics::HingeJoint< SimbodyJoint >` (p. 514), `gazebo::physics::SliderJoint< DARTJoint >` (p. 1045), `gazebo::physics::SliderJoint< SimbodyJoint >` (p. 1045), `gazebo::physics::SimbodyHingeJoint` (p. 958), `gazebo::physics::SimbodyUniversalJoint` (p. 1019), `gazebo::physics::SimbodyHinge2Joint` (p. 954), `gazebo::physics::SimbodyScrewJoint` (p. 1007), `gazebo::physics::SimbodyJoint` (p. 966), `gazebo::physics::DARTJoint` (p. 334), `gazebo::physics::SimbodyBallJoint` (p. 940), `gazebo::physics::DARTHinge2Joint` (p. 321), `gazebo::physics::DARTHingeJoint` (p. 326), `gazebo::physics::DARTBallJoint` (p. 306), `gazebo::physics::DARTScrewJoint` (p. 369), `gazebo::physics::DARTSliderJoint` (p. 374), and `gazebo::physics::DARTUniversalJoint` (p. 381).

10.107.4.52 `virtual void gazebo::physics::Joint::Reset () [virtual]`

Reset the joint.

Reimplemented from `gazebo::physics::Base` (p. 179).

Reimplemented in `gazebo::physics::DARTJoint` (p. 335), and `gazebo::physics::SimbodyJoint` (p. 967).

10.107.4.53 `virtual void gazebo::physics::Joint::SetAnchor (unsigned int _index, const math::Vector3 & _anchor)` [pure virtual]

Set the anchor point.

Parameters

in	<i>_index</i>	Indx of the axis.
in	<i>_anchor</i>	Anchor value.

Implemented in `gazebo::physics::DARTJoint` (p.335), `gazebo::physics::SimbodyJoint` (p.967), and `gazebo::physics::DARTScrewJoint` (p.369).

10.107.4.54 `void gazebo::physics::Joint::SetAngle (unsigned int _index, math::Angle _angle)`

If the **Joint** (p.541) is static, Gazebo stores the state of this **Joint** (p.541) as a scalar inside the **Joint** (p.541) class, so this call will NOT move the joint dynamically for a static **Model** (p.678).

But if this **Model** (p.678) is not static, then it is updated dynamically, all the conected children **Link** (p.595)'s are moved as a result of the **Joint** (p.541) angle setting. Dynamic **Joint** (p.541) angle update is accomplished by calling **JointController::SetJointPosition** (p.571).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_angle</i>	Angle to set the joint to.

10.107.4.55 `virtual void gazebo::physics::Joint::SetAttribute (const std::string & _key, unsigned int _index, const boost::any & _value)` [pure virtual]

Set a non-generic parameter for the joint.

replaces SetAttribute(Attribute, int, double) Deprecated by bool SetParam

Parameters

in	<i>_key</i>	String key.
in	<i>_index</i>	Index of the axis.
in	<i>_value</i>	Value of the attribute.

Implemented in `gazebo::physics::DARTJoint` (p.335), `gazebo::physics::SimbodyScrewJoint` (p.1007), and `gazebo::physics::SimbodyJoint` (p.967).

10.107.4.56 `virtual void gazebo::physics::Joint::SetAxis (unsigned int _index, const math::Vector3 & _axis)` [pure virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	<i>_index</i>	Index of the axis to set.
in	<i>_axis</i>	Vector in local joint frame of axis direction (must have length greater than zero).

Implemented in `gazebo::physics::SimbodyJoint` (p. 968), `gazebo::physics::SimbodyBallJoint` (p. 940), `gazebo::physics::DARTBallJoint` (p. 306), `gazebo::physics::DARTScrewJoint` (p. 369), `gazebo::physics::DARTHinge2Joint` (p. 321), `gazebo::physics::DARTHingeJoint` (p. 326), `gazebo::physics::SimbodyUniversalJoint` (p. 1019), `gazebo::physics::SimbodyHinge2Joint` (p. 954), `gazebo::physics::DARTSliderJoint` (p. 375), `gazebo::physics::DARTUniversalJoint` (p. 382), `gazebo::physics::SimbodyHingeJoint` (p. 959), `gazebo::physics::SimbodyScrewJoint` (p. 1007), and `gazebo::physics::SimbodySliderJoint` (p. 1013).

10.107.4.57 `virtual void gazebo::physics::Joint::SetDamping (unsigned int _index, double _damping) [pure virtual]`

Set the joint damping.

Parameters

in	<code>_index</code>	Index of the axis to set, currently ignored, to be implemented.
in	<code>_damping</code>	Damping value for the axis.

Implemented in `gazebo::physics::DARTJoint` (p. 335), and `gazebo::physics::SimbodyJoint` (p. 968).

10.107.4.58 `virtual void gazebo::physics::Joint::SetEffortLimit (unsigned int _index, double _effort) [virtual]`

Set the effort limit on a joint axis.

Parameters

in	<code>_index</code>	Index of the axis to set.
in	<code>_effort</code>	Effort limit for the axis.

10.107.4.59 `virtual void gazebo::physics::Joint::SetForce (unsigned int _index, double _effort) [pure virtual]`

Set the force applied to this `physics::Joint` (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to `SetForce` to the same joint in the same time step will accumulate forces on that `Joint` (p. 541)). Forces are truncated by `effortLimit` before applied.

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_effort</code>	Force value.

Implemented in `gazebo::physics::DARTJoint` (p. 335), and `gazebo::physics::SimbodyJoint` (p. 968).

10.107.4.60 `virtual bool gazebo::physics::Joint::SetHighStop (unsigned int _index, const math::Angle & _angle) [virtual]`

Set the high stop of an axis(index).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_angle</code>	High stop angle.

Reimplemented in **gazebo::physics::SimbodyJoint** (p.969), **gazebo::physics::SimbodyBallJoint** (p.940), **gazebo::physics::DARTJoint** (p.336), **gazebo::physics::DARTBallJoint** (p.307), and **gazebo::physics::SimbodyScrewJoint** (p.1008).

10.107.4.61 `void gazebo::physics::Joint::SetLowerLimit (unsigned int _index, math::Angle _limit)`

: set the joint upper limit (replaces SetLowStop and SetHighStop)

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_limit</code>	Lower limit of the axis.

10.107.4.62 `virtual bool gazebo::physics::Joint::SetLowStop (unsigned int _index, const math::Angle & _angle)`
`[virtual]`

Set the low stop of an axis(index).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_angle</code>	Low stop angle.

Reimplemented in **gazebo::physics::SimbodyJoint** (p.969), **gazebo::physics::SimbodyBallJoint** (p.941), **gazebo::physics::DARTJoint** (p.336), **gazebo::physics::DARTBallJoint** (p.307), and **gazebo::physics::SimbodyScrewJoint** (p.1008).

10.107.4.63 `virtual void gazebo::physics::Joint::SetMaxForce (unsigned int _index, double _force)` `[pure virtual]`

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_force</code>	Maximum force that can be applied to the axis.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p.1008), **gazebo::physics::DARTScrewJoint** (p.370), **gazebo::physics::DARTHinge2Joint** (p.322), **gazebo::physics::DARTHingeJoint** (p.327), **gazebo::physics::SimbodyUniversalJoint** (p.1020), **gazebo::physics::SimbodyHinge2Joint** (p.954), **gazebo::physics::DARTSliderJoint** (p.375), **gazebo::physics::DARTUniversalJoint** (p.382), **gazebo::physics::SimbodyBallJoint** (p.941), **gazebo::physics::DARTBallJoint** (p.307), **gazebo::physics::SimbodyHingeJoint** (p.959), and **gazebo::physics::SimbodySliderJoint** (p.1013).

10.107.4.64 `void gazebo::physics::Joint::SetModel (ModelPtr _model)`

Set the model this joint belongs too.

Parameters

in	<code>_model</code>	Pointer to a model.
----	---------------------	---------------------

10.107.4.65 `virtual bool gazebo::physics::Joint::SetParam (const std::string & _key, unsigned int _index, const boost::any & _value) [pure virtual]`

Set a non-generic parameter for the joint.

replaces SetAttribute(Attribute, int, double)

Parameters

in	<code>_key</code>	String key.
in	<code>_index</code>	Index of the axis.
in	<code>_value</code>	Value of the attribute.

Implemented in `gazebo::physics::DARTJoint` (p.337), `gazebo::physics::SimbodyScrewJoint` (p.1009), and `gazebo::physics::SimbodyJoint` (p.969).

10.107.4.66 `virtual void gazebo::physics::Joint::SetProvideFeedback (bool _enable) [virtual]`

Set whether the joint should generate feedback.

Parameters

in	<code>_enable</code>	True to enable joint feedback.
----	----------------------	--------------------------------

10.107.4.67 `void gazebo::physics::Joint::SetState (const JointState & _state)`

Set the joint state.

Parameters

in	<code>_state</code>	Joint (p.541) state
----	---------------------	----------------------------

10.107.4.68 `virtual void gazebo::physics::Joint::SetStiffness (unsigned int _index, double _stiffness) [pure virtual]`

Set the joint spring stiffness.

Parameters

in	<code>_index</code>	Index of the axis to set, currently ignored, to be implemented.
in	<code>_stiffness</code>	Spring stiffness value for the axis. : rename to SetSpringStiffness()

Implemented in `gazebo::physics::DARTJoint` (p.337), and `gazebo::physics::SimbodyJoint` (p.970).

10.107.4.69 `virtual void gazebo::physics::Joint::SetStiffnessDamping (unsigned int _index, double _stiffness, double _damping, double _reference = 0) [pure virtual]`

Set the joint spring stiffness.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to set, currently ignored, to be implemented.
<code>in</code>	<code>_stiffness</code>	Stiffness value for the axis.
<code>in</code>	<code>_reference</code>	Spring zero load reference position. : rename to SetSpringStiffnessDamping()

Implemented in `gazebo::physics::DARTJoint` (p. 337), and `gazebo::physics::SimbodyJoint` (p. 970).

10.107.4.70 `void gazebo::physics::Joint::SetStopDissipation (unsigned int _index, double _dissipation)`

Set joint stop dissipation.

Parameters

<code>in</code>	<code>_index</code>	joint axis index.
<code>in</code>	<code>_dissipation</code>	joint stop dissipation coefficient.

10.107.4.71 `void gazebo::physics::Joint::SetStopStiffness (unsigned int _index, double _stiffness)`

Set joint stop stiffness.

Parameters

<code>in</code>	<code>_index</code>	joint axis index.
<code>in</code>	<code>_stiffness</code>	joint stop stiffness coefficient.

10.107.4.72 `void gazebo::physics::Joint::SetUpperLimit (unsigned int _index, math::Angle _limit)`

: set the joint lower limit (replacee GetLowStop and GetHighStop)

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_limit</code>	Upper limit of the axis.

10.107.4.73 `virtual void gazebo::physics::Joint::SetVelocity (unsigned int _index, double _vel) [pure virtual]`

Set the velocity of an axis(index).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_vel</code>	Velocity.

Implemented in `gazebo::physics::SimbodyScrewJoint` (p.1009), `gazebo::physics::DARTScrewJoint` (p.370), `gazebo::physics::DARTHinge2Joint` (p.322), `gazebo::physics::SimbodyHinge2Joint` (p.955), `gazebo::physics::DARTHingeJoint` (p.327), `gazebo::physics::DARTUniversalJoint` (p.382), `gazebo::physics::SimbodyUniversalJoint` (p.1020), `gazebo::physics::DARTSliderJoint` (p.375), `gazebo::physics::SimbodyBallJoint` (p.941), `gazebo::physics::SimbodyHingeJoint` (p.960), `gazebo::physics::SimbodySliderJoint` (p.1014), and `gazebo::physics::DARTBallJoint` (p.307).

10.107.4.74 `void gazebo::physics::Joint::Update () [virtual]`

Update the joint.

Reimplemented from `gazebo::physics::Base` (p.180).

10.107.4.75 `virtual void gazebo::physics::Joint::UpdateParameters (sdf::ElementPtr _sdf) [virtual]`

Update the parameters using new sdf values.

Parameters

<code>in</code>	<code>_sdf</code>	SDF values to update from.
-----------------	-------------------	----------------------------

Reimplemented from `gazebo::physics::Base` (p.180).

10.107.5 Member Data Documentation

10.107.5.1 `LinkPtr gazebo::physics::Joint::anchorLink [protected]`

Anchor link.

10.107.5.2 `math::Vector3 gazebo::physics::Joint::anchorPos [protected]`

Anchor pose.

This is the xyz offset of the joint frame from child frame specified in the parent link frame

10.107.5.3 `math::Pose gazebo::physics::Joint::anchorPose [protected]`

Anchor pose specified in SDF `<joint><pose>` tag.

AnchorPose is the transform from child link frame to joint frame specified in the child link frame. AnchorPos is more relevant in normal usage, but sometimes, we do need this (e.g. GetForceTorque and joint visualization).

10.107.5.4 `gazebo::event::ConnectionPtr gazebo::physics::Joint::applyDamping [protected]`

apply damping for adding viscous damping forces on updates

10.107.5.5 `bool gazebo::physics::Joint::axisParentModelFrame[2] [protected]`

Flags that are set to true if an axis value is expressed in the parent model frame.

Otherwise use the joint frame. See issue #494.

10.107.5.6 `LinkPtr gazebo::physics::Joint::childLink` [protected]

The first link this joint connects to.

10.107.5.7 `double gazebo::physics::Joint::dampingCoefficient` [protected]

joint dissipationCoefficient Deprecated: not used, replaced by dissipationCoefficient array

10.107.5.8 `double gazebo::physics::Joint::dissipationCoefficient[2]` [protected]

joint viscous damping coefficient Replaces dampingCoefficient

10.107.5.9 `double gazebo::physics::Joint::effortLimit[2]` [protected]

Store **Joint** (p. 541) effort limit as specified in SDF.

10.107.5.10 `math::Angle gazebo::physics::Joint::lowerLimit[2]` [protected]

Store **Joint** (p. 541) position lower limit as specified in SDF.

10.107.5.11 `ModelPtr gazebo::physics::Joint::model` [protected]

Pointer to the parent model.

10.107.5.12 `math::Pose gazebo::physics::Joint::parentAnchorPose` [protected]

Anchor pose relative to parent link frame.

10.107.5.13 `LinkPtr gazebo::physics::Joint::parentLink` [protected]

The second link this joint connects to.

10.107.5.14 `bool gazebo::physics::Joint::provideFeedback` [protected]

Provide Feedback data for contact forces.

10.107.5.15 `double gazebo::physics::Joint::springReferencePosition[2]` [protected]

joint spring reference (zero load) position

10.107.5.16 `double gazebo::physics::Joint::stiffnessCoefficient[2]` [protected]

joint stiffnessCoefficient

10.107.5.17 **math::Angle** gazebo::physics::Joint::upperLimit[2] [protected]

Store **Joint** (p. 541) position upper limit as specified in SDF.

10.107.5.18 **bool** gazebo::physics::Joint::useCFMDamping [protected]

option to use implicit damping Deprecated, pushing this flag into individual physics engine, for example: ODEJoint::useImplicitSpringDamper.

10.107.5.19 **double** gazebo::physics::Joint::velocityLimit[2] [protected]

Store **Joint** (p. 541) velocity limit as specified in SDF.

10.107.5.20 **JointWrench** gazebo::physics::Joint::wrench [protected]

Cache **Joint** (p. 541) force torque values in case physics engine clears them at the end of update step.

The documentation for this class was generated from the following file:

- **Joint.hh**

10.108 gazebo::physics::JointController Class Reference

A class for manipulating **physics::Joint** (p. 541).

```
#include <physics/physics.hh>
```

Public Member Functions

- **JointController** (**ModelPtr** _model)
Constructor.
- virtual **~JointController** ()
Destructor.
- void **AddJoint** (**JointPtr** _joint)
Add a joint to control.
- **std::map**< **std::string**, **double** > **GetForces** () const
Get all the applied forces.
- **std::map**< **std::string**, **JointPtr** > **GetJoints** () const
Get all the joints.
- **common::Time** **GetLastUpdateTime** () const
Get the last time the controller was updated.
- **std::map**< **std::string**, **common::PID** > **GetPositionPIDs** () const
Get all the position PID controllers.
- **std::map**< **std::string**, **double** > **GetPositions** () const
Get all the position PID set points.
- **std::map**< **std::string**, **double** > **GetVelocities** () const

Get all the velocity PID set points.

- `std::map< std::string, common::PID > GetVelocityPIDs ()` const

Get all the velocity PID controllers.

- void **Reset** ()

Reset all commands.

- void **SetJointPosition** (const std::string &_name, double _position, int _index=0)

*Set the positions of a **Joint** (p. 541) by name.*

- void **SetJointPosition** (**JointPtr** _joint, double _position, int _index=0)

*Set the positions of a **Joint** (p. 541) by name. The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 1044) and radians for **HingeJoint** (p. 513), etc.*

- void **SetJointPositions** (const std::map< std::string, double > &_jointPositions)

*Set the positions of a set of **Joint** (p. 541)'s.*

- bool **SetPositionTarget** (const std::string &_jointName, double _target)

Set the target position for the position PID controller.

- bool **SetVelocityTarget** (const std::string &_jointName, double _target)

Set the target velocity for the velocity PID controller.

- void **Update** ()

Update the joint control.

10.108.1 Detailed Description

A class for manipulating **physics::Joint** (p. 541).

10.108.2 Constructor & Destructor Documentation

10.108.2.1 gazebo::physics::JointController::JointController (**ModelPtr** _model) [explicit]

Constructor.

Parameters

<code>in</code>	<code>_model</code>	Model (p. 678) that uses this joint controller.
-----------------	---------------------	--

10.108.2.2 virtual gazebo::physics::JointController::~~JointController () [virtual]

Destructor.

10.108.3 Member Function Documentation

10.108.3.1 void gazebo::physics::JointController::AddJoint (**JointPtr** _joint)

Add a joint to control.

Parameters

<code>in</code>	<code>_joint</code>	Joint (p. 541) to control.
-----------------	---------------------	-----------------------------------

10.108.3.2 `std::map<std::string, double> gazebo::physics::JointController::GetForces () const`

Get all the applied forces.

Returns

A map<joint_name, force> that contains force values set by the user of the **JointController** (p. 568).

10.108.3.3 `std::map<std::string, JointPtr> gazebo::physics::JointController::GetJoints () const`

Get all the joints.

Returns

A map<joint_name, joint_ptr> to all the joints that can be controlled.

10.108.3.4 `common::Time gazebo::physics::JointController::GetLastUpdateTime () const`

Get the last time the controller was updated.

Returns

Last time the controller was updated.

10.108.3.5 `std::map<std::string, common::PID> gazebo::physics::JointController::GetPositionPIDs () const`

Get all the position PID controllers.

Returns

A map<joint_name, PID> for all the position PID controllers.

10.108.3.6 `std::map<std::string, double> gazebo::physics::JointController::GetPositions () const`

Get all the position PID set points.

Returns

A map<joint_name, position> that contains position values set by the user of the **JointController** (p. 568).

10.108.3.7 `std::map<std::string, double> gazebo::physics::JointController::GetVelocities () const`

Get all the velocity PID set points.

Returns

A map<joint_name, position> that contains velocity values set by the user of the **JointController** (p. 568).

10.108.3.8 `std::map<std::string, common::PID> gazebo::physics::JointController::GetVelocityPIDs () const`

Get all the velocity PID controllers.

Returns

A map<joint_name, PID> for all the velocity PID controllers.

10.108.3.9 `void gazebo::physics::JointController::Reset ()`

Reset all commands.

10.108.3.10 `void gazebo::physics::JointController::SetJointPosition (const std::string & _name, double _position, int _index = 0)`

Set the positions of a **Joint** (p. 541) by name.

Warning: This function is disabled since collisions are not updated correctly. See issue #1138

See Also

`JointController::SetJointPosition(JointPtr, double)`

10.108.3.11 `void gazebo::physics::JointController::SetJointPosition (JointPtr _joint, double _position, int _index = 0)`

Set the positions of a **Joint** (p. 541) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 1044) and radians for **HingeJoint** (p. 513), etc.

Implementation: In order to change the position of a **Joint** (p. 541) inside a **Model** (p. 678), this call must recursively crawl through all the connected children **Link** (p. 595)'s in this **Model** (p. 678), and update each **Link** (p. 595) Pose affected by this **Joint** (p. 541) angle update. Warning: There is no constraint satisfaction being done here, traversal through the kinematic graph has unexpected behavior if you try to set the joint position of a link inside a loop structure. Warning: This function is disabled since collisions are not updated correctly. See issue #1138

Parameters

<code>in</code>	<code>_joint</code>	Joint (p. 541) to set.
<code>in</code>	<code>_position</code>	Position of the joint.

10.108.3.12 `void gazebo::physics::JointController::SetJointPositions (const std::map< std::string, double > & _jointPositions)`

Set the positions of a set of **Joint** (p. 541)'s.

Warning: This function is disabled since collisions are not updated correctly. See issue #1138

See Also

`JointController::SetJointPosition(JointPtr, double)`

10.108.3.13 `bool gazebo::physics::JointController::SetPositionTarget (const std::string & _jointName, double _target)`

Set the target position for the position PID controller.

Parameters

in	<code>_jointName</code>	Scoped name of the joint.
in	<code>_target</code>	Position target.

Returns

False if the joint was not found.

10.108.3.14 `bool gazebo::physics::JointController::SetVelocityTarget (const std::string & _jointName, double _target)`

Set the target velocity for the velocity PID controller.

Parameters

in	<code>_jointName</code>	Scoped name of the joint.
in	<code>_target</code>	Velocity target.

Returns

False if the joint was not found.

10.108.3.15 `void gazebo::physics::JointController::Update ()`

Update the joint control.

The documentation for this class was generated from the following file:

- **JointController.hh**

10.109 gazebo::physics::JointControllerPrivate Class Reference

```
#include <JointControllerPrivate.hh>
```

Public Attributes

- `std::map< std::string, double > forces`
Forces applied to joints.
- `transport::SubscriberPtr jointCmdSub`
Subscribe to joint command.
- `std::map< std::string, JointPtr > joints`
Map of joint names to the joint pointer.
- **ModelPtr model**
Model (p. 678) to control.
- `transport::NodePtr node`
Node for communication.
- `std::map< std::string, double > positions`
Joint (p. 541) positions.

- `std::map< std::string, common::PID > posPids`
Position PID controllers.
- `common::Time prevUpdateTime`
Last time the controller was updated.
- `Link_V updatedLinks`
List of links that have been updated.
- `std::map< std::string, double > velocities`
Joint (p. 541) velocities.
- `std::map< std::string, common::PID > velPids`
Velocity PID controllers.

10.109.1 Member Data Documentation

10.109.1.1 `std::map<std::string, double> gazebo::physics::JointControllerPrivate::forces`

Forces applied to joints.

10.109.1.2 `transport::SubscriberPtr gazebo::physics::JointControllerPrivate::jointCmdSub`

Subscribe to joint command.

10.109.1.3 `std::map<std::string, JointPtr> gazebo::physics::JointControllerPrivate::joints`

Map of joint names to the joint pointer.

10.109.1.4 `ModelPtr gazebo::physics::JointControllerPrivate::model`

Model (p. 678) to control.

10.109.1.5 `transport::NodePtr gazebo::physics::JointControllerPrivate::node`

Node for communication.

10.109.1.6 `std::map<std::string, double> gazebo::physics::JointControllerPrivate::positions`

Joint (p. 541) positions.

10.109.1.7 `std::map<std::string, common::PID> gazebo::physics::JointControllerPrivate::posPids`

Position PID controllers.

10.109.1.8 `common::Time gazebo::physics::JointControllerPrivate::prevUpdateTime`

Last time the controller was updated.

10.109.1.9 Link_V gazebo::physics::JointControllerPrivate::updatedLinks

List of links that have been updated.

10.109.1.10 std::map<std::string, double> gazebo::physics::JointControllerPrivate::velocities

Joint (p. 541) velocities.

10.109.1.11 std::map<std::string, common::PID> gazebo::physics::JointControllerPrivate::velPids

Velocity PID controllers.

The documentation for this class was generated from the following file:

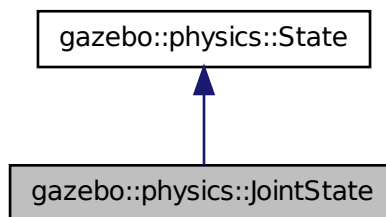
- **JointControllerPrivate.hh**

10.110 gazebo::physics::JointState Class Reference

keeps track of state of a **physics::Joint** (p. 541)

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::JointState:



Public Member Functions

- **JointState** ()
Default constructor.
- **JointState** (**JointPtr** _joint, const **common::Time** &_realTime, const **common::Time** &_simTime)
Constructor.
- **JointState** (**JointPtr** _joint)
Constructor.
- **JointState** (const sdf::ElementPtr _sdf)
Constructor.
- virtual ~**JointState** ()

Destructor.

- void **FillSDF** (sdf::ElementPtr _sdf)
Populate a state SDF element with data from the object.
- **math::Angle GetAngle** (unsigned int _axis) const
Get the joint angle.
- unsigned int **GetAngleCount** () const
Get the number of angles.
- const std::vector< **math::Angle** > & **GetAngles** () const
Get the angles.
- bool **IsZero** () const
Return true if the values in the state are zero.
- void **Load** (**JointPtr** _joint, const **common::Time** &_realTime, const **common::Time** &_simTime)
Load.
- virtual void **Load** (const sdf::ElementPtr _elem)
Load state from SDF element.
- **JointState operator+** (const **JointState** &_state) const
Addition operator.
- **JointState operator-** (const **JointState** &_state) const
Subtraction operator.
- **JointState & operator=** (const **JointState** &_state)
Assignment operator.

Friends

- std::ostream & **operator<<** (std::ostream &_out, const **gazebo::physics::JointState** &_state)
Stream insertion operator.

Additional Inherited Members

10.110.1 Detailed Description

keeps track of state of a **physics::Joint** (p. 541)

10.110.2 Constructor & Destructor Documentation

10.110.2.1 gazebo::physics::JointState::JointState ()

Default constructor.

10.110.2.2 gazebo::physics::JointState::JointState (**JointPtr** _joint, const **common::Time** &_realTime, const **common::Time** &_simTime)

Constructor.

Parameters

in	<i>_joint</i>	Joint (p. 541) to get the state of.
in	<i>_realTime</i>	Real time stamp.
in	<i>_simTime</i>	Sim time stamp.

10.110.2.3 gazebo::physics::JointState::JointState (JointPtr *_joint*) [explicit]

Constructor.

Parameters

in	<i>_joint</i>	Joint (p. 541) to get the state of.
----	---------------	-------------------------------------

10.110.2.4 gazebo::physics::JointState::JointState (const sdf::ElementPtr *_sdf*) [explicit]

Constructor.

Build a **JointState** (p. 574) from SDF data

Parameters

in	<i>_sdf</i>	SDF data to load a joint state from.
----	-------------	--------------------------------------

10.110.2.5 virtual gazebo::physics::JointState::~~JointState () [virtual]

Destructor.

10.110.3 Member Function Documentation

10.110.3.1 void gazebo::physics::JointState::FillSDF (sdf::ElementPtr *_sdf*)

Populate a state SDF element with data from the object.

Parameters

out	<i>_sdf</i>	SDF element to populate.
-----	-------------	--------------------------

10.110.3.2 math::Angle gazebo::physics::JointState::GetAngle (unsigned int *_axis*) const

Get the joint angle.

Parameters

in	<i>_axis</i>	The axis index.
----	--------------	-----------------

Returns

Angle of the axis.

Exceptions

<i>common::Exception</i> (p. 444)	When <i>_axis</i> is invalid.
---	-------------------------------

10.110.3.3 `unsigned int gazebo::physics::JointState::GetAngleCount () const`

Get the number of angles.

Returns

The number of angles.

10.110.3.4 `const std::vector<math::Angle>& gazebo::physics::JointState::GetAngles () const`

Get the angles.

Returns

Vector of angles.

10.110.3.5 `bool gazebo::physics::JointState::IsZero () const`

Return true if the values in the state are zero.

Returns

True if the values in the state are zero.

10.110.3.6 `void gazebo::physics::JointState::Load (JointPtr _joint, const common::Time & _realTime, const common::Time & _simTime)`

Load.

Parameters

<code>in</code>	<code>_joint</code>	Joint (p. 541) to get the state of.
<code>in</code>	<code>_realTime</code>	Real time stamp.
<code>in</code>	<code>_simTime</code>	Sim time stamp.

10.110.3.7 `virtual void gazebo::physics::JointState::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Parameters

<code>in</code>	<code>_elem</code>	SDF values to load from.
-----------------	--------------------	--------------------------

Reimplemented from **gazebo::physics::State** (p. 1070).

10.110.3.8 `JointState gazebo::physics::JointState::operator+ (const JointState & _state) const`

Addition operator.

Parameters

in	<code>_pt</code>	A state to add.
----	------------------	-----------------

Returns

The resulting state.

10.110.3.9 `JointState gazebo::physics::JointState::operator- (const JointState & _state) const`

Subtraction operator.

Parameters

in	<code>_pt</code>	A state to subtract.
----	------------------	----------------------

Returns

The resulting state.

10.110.3.10 `JointState& gazebo::physics::JointState::operator= (const JointState & _state)`

Assignment operator.

Parameters

in	<code>_state</code>	State (p. 1068) value
----	---------------------	------------------------------

Returns

this

10.110.4 Friends And Related Function Documentation

10.110.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::physics::JointState & _state) [friend]`

Stream insertion operator.

Parameters

in	<code>_out</code>	output stream.
in	<code>_state</code>	Joint (p. 541) state to output.

Returns

The stream.

The documentation for this class was generated from the following file:

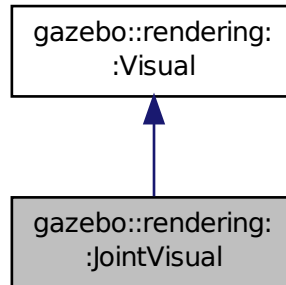
- **JointState.hh**

10.111 gazebo::rendering::JointVisual Class Reference

Visualization for joints.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::JointVisual:



Public Member Functions

- **JointVisual** (const std::string &_name, **VisualPtr** _vis)
Constructor.
- virtual ~**JointVisual** ()
Destructor.
- void **Load** (ConstJointPtr &_msg)
Load the visual based on a message.

Additional Inherited Members

10.111.1 Detailed Description

Visualization for joints.

10.111.2 Constructor & Destructor Documentation

10.111.2.1 gazebo::rendering::JointVisual::JointVisual (const std::string & _name, VisualPtr _vis)

Constructor.

Parameters

in	<code>_name</code>	Name of the visual
in	<code>_vis</code>	Pointer to the parent visual

10.111.2.2 virtual gazebo::rendering::JointVisual::~~JointVisual () [virtual]

Destructor.

10.111.3 Member Function Documentation

10.111.3.1 void gazebo::rendering::JointVisual::Load (ConstJointPtr & _msg)

Load the visual based on a message.

Parameters

in	<code>_msg</code>	Joint message
----	-------------------	---------------

The documentation for this class was generated from the following file:

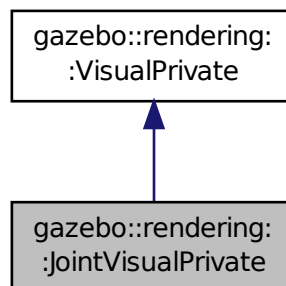
- **JointVisual.hh**

10.112 gazebo::rendering::JointVisualPrivate Class Reference

Private data for the Joint **Visual** (p. 1196) class.

```
#include <JointVisualPrivate.hh>
```

Inheritance diagram for gazebo::rendering::JointVisualPrivate:



Public Attributes

- **AxisVisualPtr axisVisual**

The visual used to draw the joint.

Additional Inherited Members

10.112.1 Detailed Description

Private data for the Joint **Visual** (p. 1196) class.

10.112.2 Member Data Documentation

10.112.2.1 AxisVisualPtr gazebo::rendering::JointVisualPrivate::axisVisual

The visual used to draw the joint.

The documentation for this class was generated from the following file:

- **JointVisualPrivate.hh**

10.113 gazebo::physics::JointWrench Class Reference

Wrench information from a joint.

```
#include <physics/physics.hh>
```

Public Member Functions

- **JointWrench & operator+** (const **JointWrench** &_wrench)
Operator +.
- **JointWrench & operator-** (const **JointWrench** &_wrench)
Operator -.
- **JointWrench & operator=** (const **JointWrench** &_wrench)
Operator =.

Public Attributes

- **math::Vector3 body1Force**
Force on the first link.
- **math::Vector3 body1Torque**
Torque on the first link.
- **math::Vector3 body2Force**
Force on the second link.
- **math::Vector3 body2Torque**
Torque on the second link.

10.113.1 Detailed Description

Wrench information from a joint.

These are forces and torques on parent and child Links, relative to the **Joint** (p. 541) frame immediately after rotation.

10.113.2 Member Function Documentation

10.113.2.1 `JointWrench& gazebo::physics::JointWrench::operator+ (const JointWrench & _wrench) [inline]`

Operator +.

Parameters

in	_wrench	Joint (p. 541) wrench to add
----	---------	-------------------------------------

Returns

*this

References body1Force, body1Torque, body2Force, and body2Torque.

10.113.2.2 `JointWrench& gazebo::physics::JointWrench::operator- (const JointWrench & _wrench) [inline]`

Operator -.

Parameters

in	_wrench	Joint (p. 541) wrench to subtract
----	---------	--

Returns

*this

References body1Force, body1Torque, body2Force, and body2Torque.

10.113.2.3 `JointWrench& gazebo::physics::JointWrench::operator= (const JointWrench & _wrench) [inline]`

Operator =.

Parameters

in	_wrench	Joint (p. 541) wrench to set from.
----	---------	---

Returns

*this

References body1Force, body1Torque, body2Force, and body2Torque.

10.113.3 Member Data Documentation

10.113.3.1 `math::Vector3 gazebo::physics::JointWrench::body1Force`

Force on the first link.

Referenced by operator+(), operator-(), and operator=().

10.113.3.2 `math::Vector3 gazebo::physics::JointWrench::body1Torque`

Torque on the first link.

Referenced by `operator+()`, `operator-()`, and `operator=()`.

10.113.3.3 `math::Vector3 gazebo::physics::JointWrench::body2Force`

Force on the second link.

Referenced by `operator+()`, `operator-()`, and `operator=()`.

10.113.3.4 `math::Vector3 gazebo::physics::JointWrench::body2Torque`

Torque on the second link.

Referenced by `operator+()`, `operator-()`, and `operator=()`.

The documentation for this class was generated from the following file:

- **JointWrench.hh**

10.114 gazebo::common::KeyEvent Class Reference

Generic description of a keyboard event.

```
#include <common/common.hh>
```

Public Types

- enum **EventType** { **NO_EVENT**, **PRESS**, **RELEASE** }
Key event types enumeration.

Public Member Functions

- **KeyEvent** ()
Constructor.

Public Attributes

- int **key**
- **EventType** **type**
Event type.

10.114.1 Detailed Description

Generic description of a keyboard event.

10.114.2 Member Enumeration Documentation

10.114.2.1 enum gazebo::common::KeyEvent::EventType

Key event types enumeration.

Enumerator:

NO_EVENT

PRESS

RELEASE

10.114.3 Constructor & Destructor Documentation

10.114.3.1 gazebo::common::KeyEvent::KeyEvent () [inline]

Constructor.

10.114.4 Member Data Documentation

10.114.4.1 int gazebo::common::KeyEvent::key

10.114.4.2 EventType gazebo::common::KeyEvent::type

Event type.

The documentation for this class was generated from the following file:

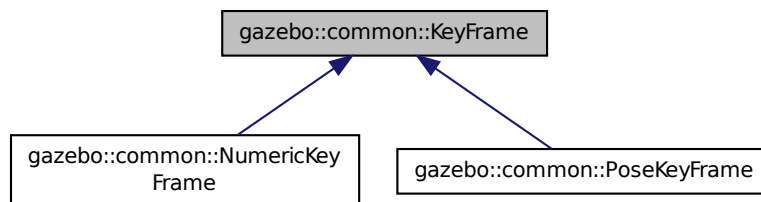
- **KeyEvent.hh**

10.115 gazebo::common::KeyFrame Class Reference

A key frame in an animation.

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::common::KeyFrame:



Public Member Functions

- **KeyFrame** (double *_time*)
Constructor.
- virtual **~KeyFrame** ()
Destructor.
- double **GetTime** () const
Get the time of the keyframe.

Protected Attributes

- double **time**
time of key frame

10.115.1 Detailed Description

A key frame in an animation.

10.115.2 Constructor & Destructor Documentation

10.115.2.1 gazebo::common::KeyFrame::KeyFrame (double *_time*)

Constructor.

Parameters

in	<i>_time</i>	Time (p. 1099) of the keyframe in seconds
----	--------------	--

10.115.2.2 virtual gazebo::common::KeyFrame::~~KeyFrame () [virtual]

Destructor.

10.115.3 Member Function Documentation

10.115.3.1 double gazebo::common::KeyFrame::GetTime () const

Get the time of the keyframe.

Returns

- the time

10.115.4 Member Data Documentation

10.115.4.1 double gazebo::common::KeyFrame::time [protected]

time of key frame

The documentation for this class was generated from the following file:

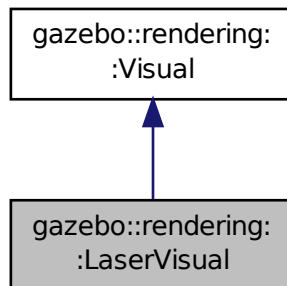
- [KeyFrame.hh](#)

10.116 gazebo::rendering::LaserVisual Class Reference

Visualization for laser data.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::LaserVisual:



Public Member Functions

- **LaserVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)
Constructor.
- virtual \sim **LaserVisual** ()
Destructor.
- virtual void **SetEmissive** (const **common::Color** &_color)
Documentation inherited from parent.

Additional Inherited Members

10.116.1 Detailed Description

Visualization for laser data.

10.116.2 Constructor & Destructor Documentation

10.116.2.1 `gazebo::rendering::LaserVisual::LaserVisual (const std::string & _name, VisualIPtr _vis, const std::string & _topicName)`

Constructor.

Parameters

in	<code>_name</code>	Name of the visual.
in	<code>_vis</code>	Pointer to the parent Visual (p. 1196).
in	<code>_topicName</code>	Name of the topic that has laser data.

10.116.2.2 `virtual gazebo::rendering::LaserVisual::~~LaserVisual () [virtual]`

Destructor.

10.116.3 Member Function Documentation

10.116.3.1 `virtual void gazebo::rendering::LaserVisual::SetEmissive (const common::Color & _color) [virtual]`

Documentation inherited from parent.

Reimplemented from `gazebo::rendering::Visual` (p. 1212).

The documentation for this class was generated from the following file:

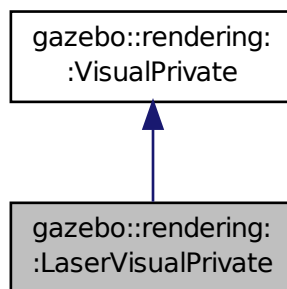
- `LaserVisual.hh`

10.117 gazebo::rendering::LaserVisualPrivate Class Reference

Private data for the Laser **Visual** (p. 1196) class.

```
#include <LaserVisualPrivate.hh>
```

Inheritance diagram for `gazebo::rendering::LaserVisualPrivate`:



Public Attributes

- `event::ConnectionPtr connection`
Pre render connection.

- `boost::shared_ptr`
`< msgs::LaserScanStamped const >` **laserMsg**
The current contact message.
- **transport::SubscriberPtr laserScanSub**
Subscription to the laser data.
- `boost::mutex` **mutex**
Mutex to protect the contact message.
- **transport::NodePtr node**
Pointer to a node that handles communication.
- `std::vector< DynamicLines * >` **rayFans**
Renders the laser data.
- `bool` **receivedMsg**
True if we have received a message.

Additional Inherited Members

10.117.1 Detailed Description

Private data for the Laser **Visual** (p. 1196) class.

10.117.2 Member Data Documentation

10.117.2.1 `event::ConnectionPtr gazebo::rendering::LaserVisualPrivate::connection`

Pre render connection.

10.117.2.2 `boost::shared_ptr<msgs::LaserScanStamped const> gazebo::rendering::LaserVisualPrivate::laserMsg`

The current contact message.

10.117.2.3 `transport::SubscriberPtr gazebo::rendering::LaserVisualPrivate::laserScanSub`

Subscription to the laser data.

10.117.2.4 `boost::mutex gazebo::rendering::LaserVisualPrivate::mutex`

Mutex to protect the contact message.

10.117.2.5 `transport::NodePtr gazebo::rendering::LaserVisualPrivate::node`

Pointer to a node that handles communication.

10.117.2.6 `std::vector<DynamicLines *> gazebo::rendering::LaserVisualPrivate::rayFans`

Renders the laser data.

10.117.2.7 bool gazebo::rendering::LaserVisualPrivate::receivedMsg

True if we have received a message.

The documentation for this class was generated from the following file:

- **LaserVisualPrivate.hh**

10.118 gazebo::rendering::Light Class Reference

A light source.

```
#include <rendering/rendering.hh>
```

Public Member Functions

- **Light (ScenePtr _scene)**
Constructor.
- virtual **~Light ()**
Destructor.
- void **FillMsg** (msgs::Light &_msg) const
Fill the contents of a light message.
- **common::Color GetDiffuseColor ()** const
Get the diffuse color.
- **math::Vector3 GetDirection ()** const
Get the direction.
- std::string **GetName ()** const
Get the name of the visual.
- **math::Vector3 GetPosition ()** const
Get the position of the light.
- **common::Color GetSpecularColor ()** const
Get the specular color.
- std::string **GetType ()** const
Get the type of the light.
- bool **GetVisible ()** const
Get whether the light is visible.
- void **Load** (sdf::ElementPtr _sdf)
Load the light using a set of SDF parameters.
- void **Load ()**
Load the light using default parameters.
- void **LoadFromMsg** (ConstLightPtr &_msg)
Load from a light message.
- void **LoadFromMsg** (const msgs::Light &_msg)
Load from a light message.
- void **SetAttenuation** (double _constant, double _linear, double _quadratic)
Set the attenuation.
- void **SetCastShadows** (const bool &_cast)
Set cast shadows.

- void **SetDiffuseColor** (const **common::Color** &_color)
Set the diffuse color.
- void **SetDirection** (const **math::Vector3** &_dir)
Set the direction.
- void **SetLightType** (const std::string &_type)
Set the light type.
- void **SetName** (const std::string &_name)
Set the name of the visual.
- void **SetPosition** (const **math::Vector3** &_p)
Set the position of the light.
- void **SetRange** (const double &_range)
Set the range.
- virtual bool **SetSelected** (bool _s)
Set whether this entity has been selected by the user through the gui.
- void **SetSpecularColor** (const **common::Color** &_color)
Set the specular color.
- void **SetSpotFalloff** (const double &_value)
Set the spot light falloff.
- void **SetSpotInnerAngle** (const double &_angle)
Set the spot light inner angle.
- void **SetSpotOuterAngle** (const double &_angle)
Set the spot light outer angle.
- void **ShowVisual** (bool _s)
Set whether to show the visual.
- void **ToggleShowVisual** ()
- void **UpdateFromMsg** (ConstLightPtr &_msg)
Update a light source from a message.

Protected Member Functions

- virtual void **OnPoseChange** ()
On pose change callback.

10.118.1 Detailed Description

A light source.

There are three types of lights: Point, Spot, and Directional. This class encapsulates all three. Point lights are light light bulbs, spot lights project a cone of light, and directional lights are light sun light.

10.118.2 Constructor & Destructor Documentation

10.118.2.1 gazebo::rendering::Light::Light (ScenePtr _scene)

Constructor.

Parameters

in	_scene	Pointer to the scene that contains the Light (p. 589).
----	--------	---

10.118.2.2 `virtual gazebo::rendering::Light::~~Light () [virtual]`

Destructor.

10.118.3 Member Function Documentation

10.118.3.1 `void gazebo::rendering::Light::FillMsg (msgs::Light & _msg) const`

Fill the contents of a light message.

Parameters

out	<code>_msg</code>	Message to fill.
-----	-------------------	------------------

10.118.3.2 `common::Color gazebo::rendering::Light::GetDiffuseColor () const`

Get the diffuse color.

Returns

The light's diffuse color.

10.118.3.3 `math::Vector3 gazebo::rendering::Light::GetDirection () const`

Get the direction.

Returns

The light's direction.

10.118.3.4 `std::string gazebo::rendering::Light::GetName () const`

Get the name of the visual.

Returns

The light's name.

10.118.3.5 `math::Vector3 gazebo::rendering::Light::GetPosition () const`

Get the position of the light.

Returns

The position of the light

10.118.3.6 `common::Color gazebo::rendering::Light::GetSpecularColor () const`

Get the specular color.

Returns

The specular color

10.118.3.7 `std::string gazebo::rendering::Light::GetType () const`

Get the type of the light.

Returns

The light type: "point", "spot", "directional".

10.118.3.8 `bool gazebo::rendering::Light::GetVisible () const`

Get whether the light is visible.

Returns

True if the light is visible.

10.118.3.9 `void gazebo::rendering::Light::Load (sdf::ElementPtr _sdf)`

Load the light using a set of SDF parameters.

Parameters

<code>in</code>	<code>_sdf</code>	Pointer to the SDF containing the Light (p. 589) description.
-----------------	-------------------	--

10.118.3.10 `void gazebo::rendering::Light::Load ()`

Load the light using default parameters.

10.118.3.11 `void gazebo::rendering::Light::LoadFromMsg (ConstLightPtr & _msg)`

Load from a light message.

Parameters

<code>in</code>	<code>_msg</code>	Containing the light information.
-----------------	-------------------	-----------------------------------

10.118.3.12 `void gazebo::rendering::Light::LoadFromMsg (const msgs::Light & _msg)`

Load from a light message.

Parameters

in	<code>_msg</code>	Message containing the light information.
----	-------------------	---

10.118.3.13 `virtual void gazebo::rendering::Light::OnPoseChange () [inline], [protected], [virtual]`

On pose change callback.

10.118.3.14 `void gazebo::rendering::Light::SetAttenuation (double _constant, double _linear, double _quadratic)`

Set the attenuation.

Parameters

in	<code>_constant</code>	Constant attenuation
in	<code>_linear</code>	Linear attenuation
in	<code>_quadratic</code>	Quadratic attenuation

10.118.3.15 `void gazebo::rendering::Light::SetCastShadows (const bool & _cast)`

Set cast shadows.

Parameters

in	<code>_cast</code>	Set to true to cast shadows.
----	--------------------	------------------------------

10.118.3.16 `void gazebo::rendering::Light::SetDiffuseColor (const common::Color & _color)`

Set the diffuse color.

Parameters

in	<code>_color</code>	Light (p. 589) diffuse color.
----	---------------------	--------------------------------------

10.118.3.17 `void gazebo::rendering::Light::SetDirection (const math::Vector3 & _dir)`

Set the direction.

Parameters

in	<code>_dir</code>	Set the light's direction. Only applicable to spot and directional lights.
----	-------------------	--

10.118.3.18 `void gazebo::rendering::Light::SetLightType (const std::string & _type)`

Set the light type.

Parameters

in	_type	The light type: "point", "spot", "directional"
----	-------	--

10.118.3.19 void gazebo::rendering::Light::SetName (const std::string & *_name*)

Set the name of the visual.

Parameters

in	_name	Name of the light source.
----	-------	---------------------------

10.118.3.20 void gazebo::rendering::Light::SetPosition (const math::Vector3 & *_p*)

Set the position of the light.

Parameters

in	_p	New position for the light
----	----	----------------------------

10.118.3.21 void gazebo::rendering::Light::SetRange (const double & *_range*)

Set the range.

Parameters

in	_range	Rage of the light in meters.
----	--------	------------------------------

10.118.3.22 virtual bool gazebo::rendering::Light::SetSelected (bool *_s*) [virtual]

Set whether this entity has been selected by the user through the gui.

Parameters

in	_s	Set to True when the light is selected by the user.
----	----	---

10.118.3.23 void gazebo::rendering::Light::SetSpecularColor (const common::Color & *_color*)

Set the specular color.

Parameters

in	_color	The specular color
----	--------	--------------------

10.118.3.24 void gazebo::rendering::Light::SetSpotFalloff (const double & *_value*)

Set the spot light falloff.

Parameters

in	<code>_value</code>	Falloff value
----	---------------------	---------------

10.118.3.25 `void gazebo::rendering::Light::SetSpotInnerAngle (const double & _angle)`

Set the spot light inner angle.

Parameters

in	<code>_angle</code>	Inner angle in radians
----	---------------------	------------------------

10.118.3.26 `void gazebo::rendering::Light::SetSpotOuterAngle (const double & _angle)`

Set the spot light outer angle.

Parameters

in	<code>_angle</code>	Outer angle in radians
----	---------------------	------------------------

10.118.3.27 `void gazebo::rendering::Light::ShowVisual (bool _s)`

Set whether to show the visual.

Parameters

in	<code>_s</code>	Set to true to draw a representation of the light.
----	-----------------	--

10.118.3.28 `void gazebo::rendering::Light::ToggleShowVisual ()`

10.118.3.29 `void gazebo::rendering::Light::UpdateFromMsg (ConstLightPtr & _msg)`

Update a light source from a message.

Parameters

in	<code>_msg</code>	Light (p. 589) message to update from
----	-------------------	--

The documentation for this class was generated from the following file:

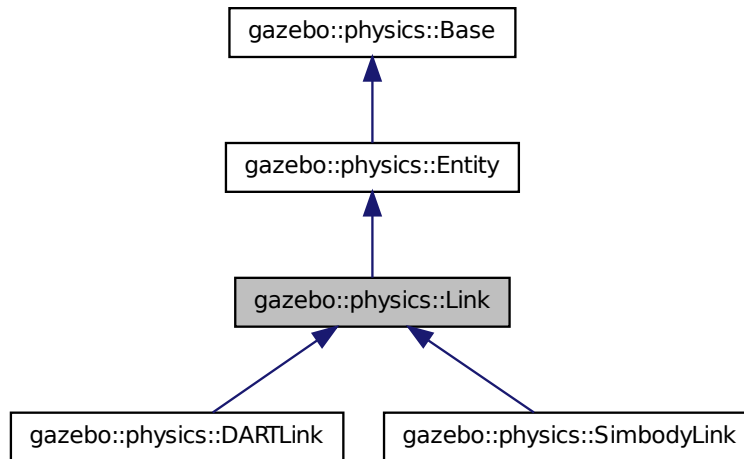
- **Light.hh**

10.119 gazebo::physics::Link Class Reference

Link (p. 595) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::Link:



Public Member Functions

- **Link** (**EntityPtr** _parent)
Constructor.
- virtual **~Link** ()
Destructor.
- void **AddChildJoint** (**JointPtr** _joint)
*Joints that have this **Link** (p. 595) as a parent **Link** (p. 595).*
- virtual void **AddForce** (const **math::Vector3** &_force)=0
Add a force to the body.
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relPos)=0
Add a force to the body at position expressed to the body's own frame of reference.
- virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)=0
Add a force to the body using a global position.
- void **AddParentJoint** (**JointPtr** _joint)
*Joints that have this **Link** (p. 595) as a child **Link** (p. 595).*
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)=0
Add a force to the body, components are relative to the body's own frame of reference.
- virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)=0
Add a torque to the body, components are relative to the body's own frame of reference.
- virtual void **AddTorque** (const **math::Vector3** &_torque)=0
Add a torque to the body.
- void **AttachStaticModel** (**ModelPtr** &_model, const **math::Pose** &_offset)
Attach a static model to this link.
- template<typename T >
event::ConnectionPtr ConnectEnabled (T _subscriber)

- Connect to the add entity signal.*

 - void **DetachAllStaticModels** ()

Detach all static models from this link.
- void **DetachStaticModel** (const std::string &_modelName)
- Detach a static model from this link.*
- void **DisconnectEnabled** (event::ConnectionPtr &_conn)
- Disconnect to the add entity signal.*
- void **FillMsg** (msgs::Link &_msg)
- Fill a link message.*
- void **Fini** ()
- Finalize the body.*
- double **GetAngularDamping** () const
- Get the angular damping factor.*
- virtual **math::Box GetBoundingBox** () const
- Get the bounding box for the link and all the child elements.*
- **Joint_V GetChildJoints** () const
- Get the child joints.*
- **Link_V GetChildJointsLinks** () const
- Returns a vector of children Links connected by joints.*
- **CollisionPtr GetCollision** (const std::string &_name)
- Get a child collision by name.*
- **CollisionPtr GetCollision** (unsigned int _index) const
- Get a child collision by index.*
- **CollisionPtr GetCollisionById** (unsigned int _id) const
- This is an internal function*
- **Collision_V GetCollisions** () const
- Get all the child collisions.*
- virtual bool **GetEnabled** () const =0
- Get whether this body is enabled in the physics engine.*
- virtual bool **GetGravityMode** () const =0
- Get the gravity mode.*
- **InertialPtr GetInertial** () const
- Get the inertia of the link.*
- virtual bool **GetKinematic** () const
- Implement this function.*
- double **GetLinearDamping** () const
- Get the linear damping factor.*
- **ModelPtr GetModel** () const
- Get the model that this body belongs to.*
- **Joint_V GetParentJoints** () const
- Get the parent joints.*
- **Link_V GetParentJointsLinks** () const
- Returns a vector of parent Links connected by joints.*
- **math::Vector3 GetRelativeAngularAccel** () const
- Get the angular acceleration of the body.*
- **math::Vector3 GetRelativeAngularVel** () const
- Get the angular velocity of the body.*

- **math::Vector3 GetRelativeForce** () const
Get the force applied to the body.
- **math::Vector3 GetRelativeLinearAccel** () const
Get the linear acceleration of the body.
- **math::Vector3 GetRelativeLinearVel** () const
Get the linear velocity of the body.
- **math::Vector3 GetRelativeTorque** () const
Get the torque applied to the body.
- bool **GetSelfCollide** () const
Get Self-Collision Flag, if this is true, this body will collide with other bodies even if they share the same parent.
- unsigned int **GetSensorCount** () const
Get sensor count.
- std::string **GetSensorName** (unsigned int _index) const
Get sensor name.
- **math::Vector3 GetWorldAngularAccel** () const
Get the angular acceleration of the body in the world frame.
- virtual **math::Vector3 GetWorldCoGLinearVel** () const =0
Get the linear velocity at the body's center of gravity in the world frame.
- **math::Pose GetWorldCoGPose** () const
Get the pose of the body's center of gravity in the world coordinate frame.
- double **GetWorldEnergy** () const
*Returns this link's total energy, or sum of **Link::GetWorldEnergyPotential**() (p. 609) and **Link::GetWorldEnergyKinetic**() (p. 609).*
- double **GetWorldEnergyKinetic** () const
Returns this link's kinetic energy computed using link's CoG velocity in the inertial (world) frame.
- double **GetWorldEnergyPotential** () const
Returns this link's potential energy, based on position in world frame and gravity.
- virtual **math::Vector3 GetWorldForce** () const =0
Get the force applied to the body in the world frame.
- **math::Pose GetWorldInertialPose** () const
Get the world pose of the link inertia (cog position and Moment of Inertia frame).
- **math::Matrix3 GetWorldInertiaMatrix** () const
Get the inertia matrix in the world frame.
- **math::Vector3 GetWorldLinearAccel** () const
Get the linear acceleration of the body in the world frame.
- virtual **math::Vector3 GetWorldLinearVel** () const
Get the linear velocity of the origin of the link frame, expressed in the world frame.
- virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset) const =0
Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame.
- virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset, const **math::Quaternion** &q) const =0
Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.
- virtual **math::Vector3 GetWorldTorque** () const =0
Get the torque applied to the body in the world frame.
- virtual void **Init** ()
Initialize the body.
- virtual void **Load** (sdf::ElementPtr _sdf)

Load the body based on an SDF element.

- virtual void **OnPoseChange** ()
 - This function is called when the entity's (or one of its parents) pose of the parent has changed.*
- void **ProcessMsg** (const msgs::Link &_msg)
 - Update parameters from a message.*
- virtual void **RemoveChild** (EntityPtr _child)
- void **RemoveChildJoint** (const std::string &_jointName)
 - Remove Joints that have this **Link** (p. 595) as a parent **Link** (p. 595).*
- void **RemoveCollision** (const std::string &_name)
 - Remove a collision from the link.*
- void **RemoveParentJoint** (const std::string &_jointName)
 - Remove Joints that have this **Link** (p. 595) as a child **Link** (p. 595).*
- void **Reset** ()
 - Reset the link.*
- void **ResetPhysicsStates** ()
 - Reset the link.*
- void **SetAngularAccel** (const math::Vector3 &_accel)
 - Set the angular acceleration of the body.*
- virtual void **SetAngularDamping** (double _damping)=0
 - Set the angular damping factor.*
- virtual void **SetAngularVel** (const math::Vector3 &_vel)=0
 - Set the angular velocity of the body.*
- virtual void **SetAutoDisable** (bool _disable)=0
 - Allow the link to auto disable.*
- void **SetCollideMode** (const std::string &_mode)
 - Set the collide mode of the body.*
- virtual void **SetEnabled** (bool _enable) const =0
 - Set whether this body is enabled.*
- virtual void **SetForce** (const math::Vector3 &_force)=0
 - Set the force applied to the body.*
- virtual void **SetGravityMode** (bool _mode)=0
 - Set whether gravity affects this body.*
- void **SetInertial** (const InertialPtr &_inertial)
 - Set the mass of the link.*
- virtual void **SetKinematic** (const bool &_kinematic)
 - Implement this function.*
- void **SetLaserRetro** (float _retro)
 - Set the laser retro reflectiveness.*
- void **SetLinearAccel** (const math::Vector3 &_accel)
 - Set the linear acceleration of the body.*
- virtual void **SetLinearDamping** (double _damping)=0
 - Set the linear damping factor.*
- virtual void **SetLinearVel** (const math::Vector3 &_vel)=0
 - Set the linear velocity of the body.*
- virtual void **SetLinkStatic** (bool _static)=0
 - Freeze link to ground (inertial frame).*
- void **SetPublishData** (bool _enable)

- Enable/Disable link data publishing.*

 - void **SetScale** (const **math::Vector3** &_scale)
 - Set the scale of the link.*
 - virtual bool **SetSelected** (bool _set)
 - Set whether this entity has been selected by the user through the gui.*
 - virtual void **SetSelfCollide** (bool _collide)=0
 - Set whether this body will collide with others in the model.*
 - void **SetState** (const **LinkState** &_state)
 - Set the current link state.*
 - virtual void **SetTorque** (const **math::Vector3** &_torque)=0
 - Set the torque applied to the body.*
 - void **Update** (const **common::UpdateInfo** &_info)
 - Update the collision.*
 - virtual void **UpdateMass** ()
 - Update the mass matrix.*
 - virtual void **UpdateParameters** (sdf::ElementPtr _sdf)
 - Update the parameters using new sdf values.*
 - virtual void **UpdateSurface** ()
 - Update surface parameters.*

Protected Types

- typedef std::map< uint32_t, msgs::Visual > **Visuals_M**

Protected Attributes

- **math::Vector3 angularAccel**
 - Angular acceleration.*
- std::vector< **math::Pose** > **attachedModelsOffset**
 - Offsets for the attached models.*
- std::vector< std::string > **cgVisuals**
 - Center of gravity visual elements.*
- **InertialPtr inertial**
 - Inertial** (p. 529) properties.*
- bool **initialized**
 - This flag is set to true when the link is initialized.*
- **math::Vector3 linearAccel**
 - Linear acceleration.*
- **Visuals_M visuals**
 - Link** (p. 595) visual elements.*

Additional Inherited Members

10.119.1 Detailed Description

Link (p. 595) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

10.119.2 Member Typedef Documentation

10.119.2.1 `typedef std::map<uint32_t, msgs::Visual> gazebo::physics::Link::Visuals_M` [protected]

10.119.3 Constructor & Destructor Documentation

10.119.3.1 `gazebo::physics::Link::Link (EntityPtr _parent)` [explicit]

Constructor.

Parameters

in	<code>_parent</code>	Parent of this link.
----	----------------------	----------------------

10.119.3.2 `virtual gazebo::physics::Link::~~Link ()` [virtual]

Destructor.

10.119.4 Member Function Documentation

10.119.4.1 `void gazebo::physics::Link::AddChildJoint (JointPtr _joint)`

Joints that have this **Link** (p. 595) as a parent **Link** (p. 595).

Parameters

in	<code>_joint</code>	Joint (p. 541) that is a child of this link.
----	---------------------	---

10.119.4.2 `virtual void gazebo::physics::Link::AddForce (const math::Vector3 & _force)` [pure virtual]

Add a force to the body.

Parameters

in	<code>_force</code>	Force to add.
----	---------------------	---------------

Implemented in **gazebo::physics::SimbodyLink** (p. 974), and **gazebo::physics::DARTLink** (p. 341).

10.119.4.3 `virtual void gazebo::physics::Link::AddForceAtRelativePosition (const math::Vector3 & _force, const math::Vector3 & _relPos)` [pure virtual]

Add a force to the body at position expressed to the body's own frame of reference.

Parameters

in	<code>_force</code>	Force to add.
in	<code>_relPos</code>	Position on the link to add the force.

Implemented in **gazebo::physics::SimbodyLink** (p. 974), and **gazebo::physics::DARTLink** (p. 341).

10.119.4.4 `virtual void gazebo::physics::Link::AddForceAtWorldPosition (const math::Vector3 & _force, const math::Vector3 & _pos) [pure virtual]`

Add a force to the body using a global position.

Parameters

<code>in</code>	<code>_force</code>	Force to add.
<code>in</code>	<code>_pos</code>	Position in global coord frame to add the force.

Implemented in `gazebo::physics::SimbodyLink` (p. 975), and `gazebo::physics::DARTLink` (p. 341).

10.119.4.5 `void gazebo::physics::Link::AddParentJoint (JointPtr _joint)`

Joints that have this `Link` (p. 595) as a child `Link` (p. 595).

Parameters

<code>in</code>	<code>_joint</code>	<code>Joint</code> (p. 541) that is a parent of this link.
-----------------	---------------------	--

10.119.4.6 `virtual void gazebo::physics::Link::AddRelativeForce (const math::Vector3 & _force) [pure virtual]`

Add a force to the body, components are relative to the body's own frame of reference.

Parameters

<code>in</code>	<code>_force</code>	Force to add.
-----------------	---------------------	---------------

Implemented in `gazebo::physics::SimbodyLink` (p. 975), and `gazebo::physics::DARTLink` (p. 341).

10.119.4.7 `virtual void gazebo::physics::Link::AddRelativeTorque (const math::Vector3 & _torque) [pure virtual]`

Add a torque to the body, components are relative to the body's own frame of reference.

Parameters

<code>in</code>	<code>_torque</code>	Torque value to add.
-----------------	----------------------	----------------------

Implemented in `gazebo::physics::SimbodyLink` (p. 975), and `gazebo::physics::DARTLink` (p. 342).

10.119.4.8 `virtual void gazebo::physics::Link::AddTorque (const math::Vector3 & _torque) [pure virtual]`

Add a torque to the body.

Parameters

<code>in</code>	<code>_torque</code>	Torque value to add to the link.
-----------------	----------------------	----------------------------------

Implemented in `gazebo::physics::SimbodyLink` (p. 975), and `gazebo::physics::DARTLink` (p. 342).

10.119.4.9 `void gazebo::physics::Link::AttachStaticModel (ModelPtr & _model, const math::Pose & _offset)`

Attach a static model to this link.

Parameters

<code>in</code>	<code><i>_model</i></code>	Pointer to a static model.
<code>in</code>	<code><i>_offset</i></code>	Pose relative to this link to place the model.

10.119.4.10 `template<typename T> event::ConnectionPtr gazebo::physics::Link::ConnectEnabled (T _subscriber)`
`[inline]`

Connect to the add entity signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	Subscriber callback function.
-----------------	---------------------------------	-------------------------------

Returns

Pointer to the connection, which must be kept in scope.

10.119.4.11 `void gazebo::physics::Link::DetachAllStaticModels ()`

Detach all static models from this link.

10.119.4.12 `void gazebo::physics::Link::DetachStaticModel (const std::string & _modelName)`

Detach a static model from this link.

Parameters

<code>in</code>	<code><i>_modelName</i></code>	Name of an attached model to detach.
-----------------	--------------------------------	--------------------------------------

10.119.4.13 `void gazebo::physics::Link::DisconnectEnabled (event::ConnectionPtr & _conn)` `[inline]`

Disconnect to the add entity signal.

Parameters

<code>in</code>	<code><i>_conn</i></code>	Connection pointer to disconnect.
-----------------	---------------------------	-----------------------------------

10.119.4.14 `void gazebo::physics::Link::FillMsg (msgs::Link & _msg)`

Fill a link message.

Parameters

out	<code>_msg</code>	Message to fill
-----	-------------------	-----------------

10.119.4.15 `void gazebo::physics::Link::Fini () [virtual]`

Finalize the body.

Reimplemented from `gazebo::physics::Entity` (p. 407).

Reimplemented in `gazebo::physics::DARTLink` (p. 342), and `gazebo::physics::SimbodyLink` (p. 976).

10.119.4.16 `double gazebo::physics::Link::GetAngularDamping () const`

Get the angular damping factor.

Returns

Angular damping.

10.119.4.17 `virtual math::Box gazebo::physics::Link::GetBoundingBox () const [virtual]`

Get the bounding box for the link and all the child elements.

Returns

The link's bounding box.

Reimplemented from `gazebo::physics::Entity` (p. 407).

10.119.4.18 `Joint_V gazebo::physics::Link::GetChildJoints () const`

Get the child joints.

10.119.4.19 `Link_V gazebo::physics::Link::GetChildJointsLinks () const`

Returns a vector of children Links connected by joints.

Returns

A vector of children Links connected by joints.

10.119.4.20 `CollisionPtr gazebo::physics::Link::GetCollision (const std::string & _name)`

Get a child collision by name.

Parameters

in	<code>_name</code>	Name of the collision object.
----	--------------------	-------------------------------

Returns

Pointer to the collision, NULL if the name was not found.

10.119.4.21 CollisionPtr gazebo::physics::Link::GetCollision (unsigned int *_index*) const

Get a child collision by index.

Parameters

<i>in</i>	<i>_index</i>	Index of the collision object.
-----------	---------------	--------------------------------

Returns

Pointer to the collision, NULL if the name was not found.

10.119.4.22 CollisionPtr gazebo::physics::Link::GetCollisionById (unsigned int *_id*) const

This is an internal function

Get a collision by id.

Parameters

<i>in</i>	<i>_id</i>	Id of the collision object to find.
-----------	------------	-------------------------------------

Returns

Pointer to the collision, NULL if the id is invalid.

10.119.4.23 Collision_V gazebo::physics::Link::GetCollisions () const

Get all the child collisions.

Returns

A std::vector of all the child collisions.

10.119.4.24 virtual bool gazebo::physics::Link::GetEnabled () const [pure virtual]

Get whether this body is enabled in the physics engine.

Returns

True if the link is enabled.

Implemented in **gazebo::physics::DARTLink** (p. 343), and **gazebo::physics::SimbodyLink** (p. 976).

10.119.4.25 `virtual bool gazebo::physics::Link::GetGravityMode () const [pure virtual]`

Get the gravity mode.

Returns

True if gravity is enabled.

Implemented in **gazebo::physics::DARTLink** (p. 343), and **gazebo::physics::SimbodyLink** (p. 976).

10.119.4.26 `InertiaIPtr gazebo::physics::Link::GetInertial () const [inline]`

Get the inertia of the link.

Returns

Inertia of the link.

10.119.4.27 `virtual bool gazebo::physics::Link::GetKinematic () const [inline],[virtual]`

Implement this function.

Get whether this body is in the kinematic state.

Returns

True if the link is kinematic only.

Reimplemented in **gazebo::physics::DARTLink** (p. 343).

10.119.4.28 `double gazebo::physics::Link::GetLinearDamping () const`

Get the linear damping factor.

Returns

Linear damping.

10.119.4.29 `ModelPtr gazebo::physics::Link::GetModel () const`

Get the model that this body belongs to.

Returns

Model (p. 678) that this body belongs to.

10.119.4.30 `Joint_V gazebo::physics::Link::GetParentJoints () const`

Get the parent joints.

10.119.4.31 `Link_V gazebo::physics::Link::GetParentJointsLinks () const`

Returns a vector of parent Links connected by joints.

Returns

Vector of parent Links connected by joints.

10.119.4.32 `math::Vector3 gazebo::physics::Link::GetRelativeAngularAccel () const [virtual]`

Get the angular acceleration of the body.

Returns

Angular acceleration of the body.

Reimplemented from `gazebo::physics::Entity` (p. 409).

10.119.4.33 `math::Vector3 gazebo::physics::Link::GetRelativeAngularVel () const [virtual]`

Get the angular velocity of the body.

Returns

Angular velocity of the body.

Reimplemented from `gazebo::physics::Entity` (p. 409).

10.119.4.34 `math::Vector3 gazebo::physics::Link::GetRelativeForce () const`

Get the force applied to the body.

Returns

Force applied to the body.

10.119.4.35 `math::Vector3 gazebo::physics::Link::GetRelativeLinearAccel () const [virtual]`

Get the linear acceleration of the body.

Returns

Linear acceleration of the body.

Reimplemented from `gazebo::physics::Entity` (p. 409).

10.119.4.36 `math::Vector3 gazebo::physics::Link::GetRelativeLinearVel () const [virtual]`

Get the linear velocity of the body.

Returns

Linear velocity of the body.

Reimplemented from `gazebo::physics::Entity` (p. 410).

10.119.4.37 `math::Vector3 gazebo::physics::Link::GetRelativeTorque () const`

Get the torque applied to the body.

Returns

Torque applied to the body.

10.119.4.38 `bool gazebo::physics::Link::GetSelfCollide () const`

Get Self-Collision Flag, if this is true, this body will collide with other bodies even if they share the same parent.

Returns

True if self collision is enabled.

10.119.4.39 `unsigned int gazebo::physics::Link::GetSensorCount () const`

Get sensor count.

This will return the number of sensors created by the link when it was loaded. This function is commonly used with **Link::GetSensorName** (p. 608).

Returns

The number of sensors created by the link.

10.119.4.40 `std::string gazebo::physics::Link::GetSensorName (unsigned int _index) const`

Get sensor name.

Get the name of a sensor based on an index. The index should be in the range of 0...**Link::GetSensorCount()** (p. 608).

Note

A **Link** (p. 595) does not manage or maintain a pointer to a **sensors::Sensor** (p. 907). Access to a Sensor object is accomplished through the **sensors::SensorManager** (p. 920). This was done to separate the physics engine from the sensor engine.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the sensor name.
-----------------	----------------------------	---------------------------

Returns

The name of the sensor, or empty string if the index is out of bounds.

10.119.4.41 `math::Vector3 gazebo::physics::Link::GetWorldAngularAccel () const [virtual]`

Get the angular acceleration of the body in the world frame.

Returns

Angular acceleration of the body in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 410).

10.119.4.42 `virtual math::Vector3 gazebo::physics::Link::GetWorldCoGLinearVel () const` [pure virtual]

Get the linear velocity at the body's center of gravity in the world frame.

Returns

Linear velocity at the body's center of gravity in the world frame.

Implemented in **gazebo::physics::DARTLink** (p. 344), and **gazebo::physics::SimbodyLink** (p. 976).

10.119.4.43 `math::Pose gazebo::physics::Link::GetWorldCoGPose () const`

Get the pose of the body's center of gravity in the world coordinate frame.

Returns

Pose of the body's center of gravity in the world coordinate frame.

10.119.4.44 `double gazebo::physics::Link::GetWorldEnergy () const`

Returns this link's total energy, or sum of **Link::GetWorldEnergyPotential()** (p. 609) and **Link::GetWorldEnergyKinetic()** (p. 609).

Returns

this link's total energy

10.119.4.45 `double gazebo::physics::Link::GetWorldEnergyKinetic () const`

Returns this link's kinetic energy computed using link's CoG velocity in the inertial (world) frame.

Returns

this link's kinetic energy

10.119.4.46 `double gazebo::physics::Link::GetWorldEnergyPotential () const`

Returns this link's potential energy, based on position in world frame and gravity.

Returns

this link's potential energy,

10.119.4.47 `virtual math::Vector3 gazebo::physics::Link::GetWorldForce () const` [pure virtual]

Get the force applied to the body in the world frame.

Returns

Force applied to the body in the world frame.

Implemented in `gazebo::physics::DARTLink` (p. 344), and `gazebo::physics::SimbodyLink` (p. 977).

10.119.4.48 `math::Pose gazebo::physics::Link::GetWorldInertialPose () const`

Get the world pose of the link inertia (cog position and Moment of Inertia frame).

This differs from `GetWorldCoGPose()` (p. 609), which returns the cog position in the link frame (not the Moment of Inertia frame).

Returns

Inertial (p. 529) pose in world frame.

10.119.4.49 `math::Matrix3 gazebo::physics::Link::GetWorldInertiaMatrix () const`

Get the inertia matrix in the world frame.

Returns

Inertia matrix in world frame, returns matrix of zeros if link has no inertia.

10.119.4.50 `math::Vector3 gazebo::physics::Link::GetWorldLinearAccel () const` [virtual]

Get the linear acceleration of the body in the world frame.

Returns

Linear acceleration of the body in the world frame.

Reimplemented from `gazebo::physics::Entity` (p. 411).

10.119.4.51 `virtual math::Vector3 gazebo::physics::Link::GetWorldLinearVel () const` [inline],[virtual]

Get the linear velocity of the origin of the link frame, expressed in the world frame.

Returns

Linear velocity of the link frame.

Reimplemented from `gazebo::physics::Entity` (p. 411).

References `GetWorldLinearVel()`, and `gazebo::math::Vector3::Zero`.

Referenced by `GetWorldLinearVel()`.

10.119.4.52 `virtual math::Vector3 gazebo::physics::Link::GetWorldLinearVel (const math::Vector3 & _offset) const` [pure virtual]

Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame. If no offset is given, the velocity at the origin of the **Link** (p. 595) frame will be returned.

Parameters

<code>in</code>	<code>_offset</code>	Offset of the point from the origin of the Link (p. 595) frame, expressed in the body-fixed frame.
-----------------	----------------------	---

Returns

Linear velocity of the point on the body

Implemented in **gazebo::physics::DARTLink** (p. 344), and **gazebo::physics::SimbodyLink** (p. 977).

10.119.4.53 `virtual math::Vector3 gazebo::physics::Link::GetWorldLinearVel (const math::Vector3 & _offset, const math::Quaternion & _q) const` [pure virtual]

Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.

Parameters

<code>in</code>	<code>_offset</code>	Offset from the origin of the link frame expressed in a frame defined by <code>_q</code> .
<code>in</code>	<code>_q</code>	Describes the rotation of a reference frame relative to the world reference frame.

Returns

Linear velocity of the point on the body in the world frame.

Implemented in **gazebo::physics::DARTLink** (p. 344), and **gazebo::physics::SimbodyLink** (p. 977).

10.119.4.54 `virtual math::Vector3 gazebo::physics::Link::GetWorldTorque () const` [pure virtual]

Get the torque applied to the body in the world frame.

Returns

Torque applied to the body in the world frame.

Implemented in **gazebo::physics::DARTLink** (p. 345), and **gazebo::physics::SimbodyLink** (p. 977).

10.119.4.55 `virtual void gazebo::physics::Link::Init ()` [virtual]

Initialize the body.

Reimplemented from **gazebo::physics::Base** (p. 177).

Reimplemented in **gazebo::physics::DARTLink** (p. 345), and **gazebo::physics::SimbodyLink** (p. 978).

10.119.4.56 `virtual void gazebo::physics::Link::Load (sdf::ElementPtr _sdf) [virtual]`

Load the body based on an SDF element.

Parameters

in	_sdf	SDF parameters.
----	------	-----------------

Reimplemented from `gazebo::physics::Entity` (p. 411).

Reimplemented in `gazebo::physics::DARTLink` (p. 345), and `gazebo::physics::SimbodyLink` (p. 978).

10.119.4.57 `virtual void gazebo::physics::Link::OnPoseChange () [virtual]`

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implements `gazebo::physics::Entity` (p. 412).

Reimplemented in `gazebo::physics::DARTLink` (p. 345), and `gazebo::physics::SimbodyLink` (p. 978).

10.119.4.58 `void gazebo::physics::Link::ProcessMsg (const msgs::Link & _msg)`

Update parameters from a message.

Parameters

in	_msg	Message to read.
----	------	------------------

10.119.4.59 `virtual void gazebo::physics::Link::RemoveChild (EntityPtr _child) [virtual]`

10.119.4.60 `void gazebo::physics::Link::RemoveChildJoint (const std::string & _jointName)`

Remove Joints that have this `Link` (p. 595) as a parent `Link` (p. 595).

Parameters

in	_jointName	Child <code>Joint</code> (p. 541) name.
----	------------	---

10.119.4.61 `void gazebo::physics::Link::RemoveCollision (const std::string & _name)`

Remove a collision from the link.

Parameters

int]	_name	Name of the collision to remove.
------	-------	----------------------------------

10.119.4.62 `void gazebo::physics::Link::RemoveParentJoint (const std::string & _jointName)`

Remove Joints that have this `Link` (p. 595) as a child `Link` (p. 595).

Parameters

in	<code>_jointName</code>	Parent Joint (p. 541) name.
----	-------------------------	------------------------------------

10.119.4.63 `void gazebo::physics::Link::Reset ()` [virtual]

Reset the link.

Reimplemented from **gazebo::physics::Entity** (p. 412).

10.119.4.64 `void gazebo::physics::Link::ResetPhysicsStates ()`

Reset the link.

10.119.4.65 `void gazebo::physics::Link::SetAngularAccel (const math::Vector3 & _accel)`

Set the angular acceleration of the body.

Parameters

in	<code>_accel</code>	Angular acceleration.
----	---------------------	-----------------------

10.119.4.66 `virtual void gazebo::physics::Link::SetAngularDamping (double _damping)` [pure virtual]

Set the angular damping factor.

Parameters

in	<code>_damping</code>	Angular damping factor.
----	-----------------------	-------------------------

Implemented in **gazebo::physics::DARTLink** (p. 345), and **gazebo::physics::SimbodyLink** (p. 978).

10.119.4.67 `virtual void gazebo::physics::Link::SetAngularVel (const math::Vector3 & _vel)` [pure virtual]

Set the angular velocity of the body.

Parameters

in	<code>_vel</code>	Angular velocity.
----	-------------------	-------------------

Implemented in **gazebo::physics::DARTLink** (p. 345), and **gazebo::physics::SimbodyLink** (p. 978).

10.119.4.68 `virtual void gazebo::physics::Link::SetAutoDisable (bool _disable)` [pure virtual]

Allow the link to auto disable.

Parameters

in	<code>_disable</code>	If true, the link is allowed to auto disable.
----	-----------------------	---

Implemented in **gazebo::physics::DARTLink** (p. 346), and **gazebo::physics::SimbodyLink** (p. 978).

10.119.4.69 `void gazebo::physics::Link::SetCollideMode (const std::string & _mode)`

Set the collide mode of the body.

Parameters

<code>in</code>	<code>_mode</code>	Collision (p. 235) Mode, this can be: [all none sensors fixed ghost] all: collides with everything none: collides with nothing sensors: collides with everything else but other sensors fixed: collides with everything else but other fixed ghost: collides with everything else but other ghost
-----------------	--------------------	--

10.119.4.70 `virtual void gazebo::physics::Link::SetEnabled (bool _enable) const` [pure virtual]

Set whether this body is enabled.

Parameters

<code>in</code>	<code>_enable</code>	True to enable the link in the physics engine.
-----------------	----------------------	--

Implemented in **gazebo::physics::DARTLink** (p. 346), and **gazebo::physics::SimbodyLink** (p. 979).

10.119.4.71 `virtual void gazebo::physics::Link::SetForce (const math::Vector3 & _force)` [pure virtual]

Set the force applied to the body.

Parameters

<code>in</code>	<code>_force</code>	Force value.
-----------------	---------------------	--------------

Implemented in **gazebo::physics::DARTLink** (p. 346), and **gazebo::physics::SimbodyLink** (p. 979).

10.119.4.72 `virtual void gazebo::physics::Link::SetGravityMode (bool _mode)` [pure virtual]

Set whether gravity affects this body.

Parameters

<code>in</code>	<code>_mode</code>	True to enable gravity.
-----------------	--------------------	-------------------------

Implemented in **gazebo::physics::DARTLink** (p. 346), and **gazebo::physics::SimbodyLink** (p. 979).

10.119.4.73 `void gazebo::physics::Link::SetInertial (const InertialPtr & _inertial)`

Set the mass of the link.

[in] `_inertial` **Inertial** (p. 529) value for the link.

10.119.4.74 `virtual void gazebo::physics::Link::SetKinematic (const bool & _kinematic) [virtual]`

Implement this function.

Set whether this body is in the kinematic state.

Parameters

<code>in</code>	<code><i>_kinematic</i></code>	True to make the link kinematic only.
-----------------	--------------------------------	---------------------------------------

Reimplemented in **`gazebo::physics::DARTLink`** (p. 346).

10.119.4.75 `void gazebo::physics::Link::SetLaserRetro (float _retro)`

Set the laser retro reflectiveness.

Parameters

<code>in</code>	<code><i>_retro</i></code>	Retro value for all child collisions.
-----------------	----------------------------	---------------------------------------

10.119.4.76 `void gazebo::physics::Link::SetLinearAccel (const math::Vector3 & _accel)`

Set the linear acceleration of the body.

Parameters

<code>in</code>	<code><i>_accel</i></code>	Linear acceleration.
-----------------	----------------------------	----------------------

10.119.4.77 `virtual void gazebo::physics::Link::SetLinearDamping (double _damping) [pure virtual]`

Set the linear damping factor.

Parameters

<code>in</code>	<code><i>_damping</i></code>	Linear damping factor.
-----------------	------------------------------	------------------------

Implemented in **`gazebo::physics::DARTLink`** (p. 347), and **`gazebo::physics::SimbodyLink`** (p. 979).

10.119.4.78 `virtual void gazebo::physics::Link::SetLinearVel (const math::Vector3 & _vel) [pure virtual]`

Set the linear velocity of the body.

Parameters

<code>in</code>	<code><i>_vel</i></code>	Linear velocity.
-----------------	--------------------------	------------------

Implemented in **`gazebo::physics::DARTLink`** (p. 347), and **`gazebo::physics::SimbodyLink`** (p. 979).

10.119.4.79 `virtual void gazebo::physics::Link::SetLinkStatic (bool _static) [pure virtual]`

Freeze link to ground (inertial frame).

Parameters

<code>in</code>	<code>_static</code>	if true, freeze link to ground. Otherwise unfreeze link.
-----------------	----------------------	--

Implemented in **gazebo::physics::SimbodyLink** (p. 980), and **gazebo::physics::DARTLink** (p. 347).

10.119.4.80 `void gazebo::physics::Link::SetPublishData (bool _enable)`

Enable/Disable link data publishing.

Parameters

<code>in</code>	<code>_enable</code>	True to enable publishing, false to stop publishing
-----------------	----------------------	---

10.119.4.81 `void gazebo::physics::Link::SetScale (const math::Vector3 & _scale)`

Set the scale of the link.

Parameters

<code>in</code>	<code>_scale</code>	Scale to set the link to.
-----------------	---------------------	---------------------------

10.119.4.82 `virtual bool gazebo::physics::Link::SetSelected (bool _set) [virtual]`

Set whether this entity has been selected by the user through the gui.

Parameters

<code>in</code>	<code>_set</code>	True to set the link as selected.
-----------------	-------------------	-----------------------------------

Reimplemented from **gazebo::physics::Base** (p. 179).

10.119.4.83 `virtual void gazebo::physics::Link::SetSelfCollide (bool _collide) [pure virtual]`

Set whether this body will collide with others in the model.

Parameters

<code>in</code>	<code>_collid</code>	True to enable collisions.
-----------------	----------------------	----------------------------

Implemented in **gazebo::physics::DARTLink** (p. 347), and **gazebo::physics::SimbodyLink** (p. 980).

10.119.4.84 `void gazebo::physics::Link::SetState (const LinkState & _state)`

Set the current link state.

Parameters

<code>in</code>	<code>_state</code>	The state to set the link to.
-----------------	---------------------	-------------------------------

10.119.4.85 `virtual void gazebo::physics::Link::SetTorque (const math::Vector3 & _torque)` [pure virtual]

Set the torque applied to the body.

Parameters

in	_torque	Torque value.
----	---------	---------------

Implemented in `gazebo::physics::DARTLink` (p. 347), and `gazebo::physics::SimbodyLink` (p. 980).

10.119.4.86 `void gazebo::physics::Link::Update (const common::UpdateInfo & _info)`

Update the collision.

Parameters

in	_info	Update information.
----	-------	---------------------

10.119.4.87 `virtual void gazebo::physics::Link::UpdateMass ()` [inline],[virtual]

Update the mass matrix.

10.119.4.88 `virtual void gazebo::physics::Link::UpdateParameters (sdf::ElementPtr _sdf)` [virtual]

Update the parameters using new sdf values.

Parameters

in	_sdf	SDF values to load from.
----	------	--------------------------

Reimplemented from `gazebo::physics::Entity` (p. 414).

10.119.4.89 `virtual void gazebo::physics::Link::UpdateSurface ()` [inline],[virtual]

Update surface parameters.

10.119.5 Member Data Documentation

10.119.5.1 `math::Vector3 gazebo::physics::Link::angularAccel` [protected]

Angular acceleration.

10.119.5.2 `std::vector<math::Pose> gazebo::physics::Link::attachedModelsOffset` [protected]

Offsets for the attached models.

10.119.5.3 `std::vector<std::string> gazebo::physics::Link::cgVisuals` [protected]

Center of gravity visual elements.

10.119.5.4 **InertialPtr** gazebo::physics::Link::inertial [protected]

Inertial (p. 529) properties.

10.119.5.5 **bool** gazebo::physics::Link::initialized [protected]

This flag is set to true when the link is initialized.

10.119.5.6 **math::Vector3** gazebo::physics::Link::linearAccel [protected]

Linear acceleration.

10.119.5.7 **Visuals_M** gazebo::physics::Link::visuals [protected]

Link (p. 595) visual elements.

The documentation for this class was generated from the following file:

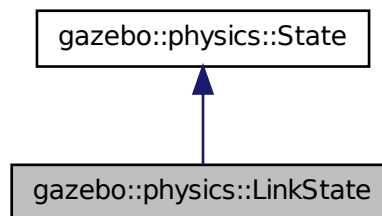
- **Link.hh**

10.120 gazebo::physics::LinkState Class Reference

Store state information of a **physics::Link** (p. 595) object.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::LinkState:



Public Member Functions

- **LinkState** ()
Default constructor.
- **LinkState** (const **LinkPtr** _link, const **common::Time** &_realTime, const **common::Time** &_simTime)
Constructor.
- **LinkState** (const **LinkPtr** _link)

- Constructor.
- **LinkState** (const sdf::ElementPtr _sdf)
 - Constructor.
- virtual **~LinkState** ()
 - Destructor.
- void **FillSDF** (sdf::ElementPtr _sdf)
 - Populate a state SDF element with data from the object.
- const **math::Pose** & **GetAcceleration** () const
 - Get the link acceleration.
- **CollisionState** **GetCollisionState** (unsigned int _index) const
 - Get a collision state.
- **CollisionState** **GetCollisionState** (const std::string &_collisionName) const
 - Get a link state by link name.
- unsigned int **GetCollisionStateCount** () const
 - Get the number of link states.
- const std::vector
 - < **CollisionState** > & **GetCollisionStates** () const
 - Get the collision states.
- const **math::Pose** & **GetPose** () const
 - Get the link pose.
- const **math::Pose** & **GetVelocity** () const
 - Get the link velocity.
- const **math::Pose** & **GetWrench** () const
 - Get the force applied to the **Link** (p. 595).
- bool **IsZero** () const
 - Return true if the values in the state are zero.
- void **Load** (const **LinkPtr** _link, const **common::Time** &_realTime, const **common::Time** &_simTime)
 - Load a **LinkState** (p. 618) from a **Link** (p. 595) pointer.
- virtual void **Load** (const sdf::ElementPtr _elem)
 - Load state from SDF element.
- **LinkState** **operator+** (const **LinkState** &_state) const
 - Addition operator.
- **LinkState** **operator-** (const **LinkState** &_state) const
 - Subtraction operator.
- **LinkState** & **operator=** (const **LinkState** &_state)
 - Assignment operator.
- virtual void **SetRealTime** (const **common::Time** &_time)
 - Set the real time when this state was generated.
- virtual void **SetSimTime** (const **common::Time** &_time)
 - Set the sim time when this state was generated.
- virtual void **SetWallTime** (const **common::Time** &_time)
 - Set the wall time when this state was generated.

Friends

- std::ostream & **operator<<** (std::ostream &_out, const **gazebo::physics::LinkState** &_state)
 - Stream insertion operator.

Additional Inherited Members

10.120.1 Detailed Description

Store state information of a **physics::Link** (p. 595) object.

This class captures the entire state of a **Link** (p. 595) at one specific time during a simulation run.

State (p. 1068) of a **Link** (p. 595) includes the state of itself all its child **Collision** (p. 235) entities.

10.120.2 Constructor & Destructor Documentation

10.120.2.1 gazebo::physics::LinkState::LinkState ()

Default constructor.

10.120.2.2 gazebo::physics::LinkState::LinkState (const LinkPtr *_link*, const common::Time & *_realTime*, const common::Time & *_simTime*)

Constructor.

Build a **LinkState** (p. 618) from an existing **Link** (p. 595).

Parameters

in	<i>_model</i>	Pointer to the Link (p. 595) from which to gather state info.
in	<i>_realTime</i>	Real time stamp.
in	<i>_simTime</i>	Sim time stamp

10.120.2.3 gazebo::physics::LinkState::LinkState (const LinkPtr *_link*) [explicit]

Constructor.

Build a **LinkState** (p. 618) from an existing **Link** (p. 595).

Parameters

in	<i>_model</i>	Pointer to the Link (p. 595) from which to gather state info.
----	---------------	--

10.120.2.4 gazebo::physics::LinkState::LinkState (const sdf::ElementPtr *_sdf*) [explicit]

Constructor.

Build a **LinkState** (p. 618) from SDF data

Parameters

in	<i>_sdf</i>	SDF data to load a link state from.
----	-------------	-------------------------------------

10.120.2.5 virtual gazebo::physics::LinkState::~~LinkState () [virtual]

Destructor.

10.120.3 Member Function Documentation

10.120.3.1 void gazebo::physics::LinkState::FillSDF (sdf::ElementPtr _sdf)

Populate a state SDF element with data from the object.

Parameters

out	<code>_sdf</code>	SDF element to populate.
-----	-------------------	--------------------------

10.120.3.2 const math::Pose& gazebo::physics::LinkState::GetAcceleration () const

Get the link acceleration.

Returns

The acceleration represented as a **math::Pose** (p. 797).

10.120.3.3 CollisionState gazebo::physics::LinkState::GetCollisionState (unsigned int _index) const

Get a collision state.

Get a **Collision** (p. 235) **State** (p. 1068) based on an index, where index is in the range of 0...**LinkState::GetCollisionStateCount** (p. 622).

Parameters

in	<code>_index</code>	Index of the CollisionState (p. 245).
----	---------------------	--

Returns

State (p. 1068) of the **Collision** (p. 235).

Exceptions

common::Exception (p. 444)	When <code>_index</code> is invalid.
--------------------------------------	--------------------------------------

10.120.3.4 CollisionState gazebo::physics::LinkState::GetCollisionState (const std::string & _collisionName) const

Get a link state by link name.

Searches through all CollisionStates. Returns the **CollisionState** (p. 245) with the matching name, if any.

Parameters

in	<code>_collisionName</code>	Name of the CollisionState (p. 245)
----	-----------------------------	--

Returns

State (p. 1068) of the **Collision** (p. 235).

Exceptions

<i>common::Exception</i> (p. 444)	When <code>_collisionName</code> is invalid
---	---

10.120.3.5 `unsigned int gazebo::physics::LinkState::GetCollisionStateCount () const`

Get the number of link states.

This returns the number of Collisions recorded.

Returns

Number of **CollisionState** (p. 245) recorded.

10.120.3.6 `const std::vector<CollisionState>& gazebo::physics::LinkState::GetCollisionStates () const`

Get the collision states.

Returns

A vector of collision states.

10.120.3.7 `const math::Pose& gazebo::physics::LinkState::GetPose () const`

Get the link pose.

Returns

The **math::Pose** (p. 797) of the **Link** (p. 595).

10.120.3.8 `const math::Pose& gazebo::physics::LinkState::GetVelocity () const`

Get the link velocity.

Returns

The velocity represented as a **math::Pose** (p. 797).

10.120.3.9 `const math::Pose& gazebo::physics::LinkState::GetWrench () const`

Get the force applied to the **Link** (p. 595).

Returns

Magnitude of the force.

10.120.3.10 `bool gazebo::physics::LinkState::IsZero () const`

Return true if the values in the state are zero.

Returns

True if the values in the state are zero.

10.120.3.11 `void gazebo::physics::LinkState::Load (const LinkPtr _link, const common::Time & _realTime, const common::Time & _simTime)`

Load a **LinkState** (p. 618) from a **Link** (p. 595) pointer.

Build a **LinkState** (p. 618) from an existing **Link** (p. 595).

Parameters

<code>in</code>	<code>_model</code>	Pointer to the Link (p. 595) from which to gather state info.
<code>in</code>	<code>_realTime</code>	Real time stamp.
<code>in</code>	<code>_simTime</code>	Sim time stamp

10.120.3.12 `virtual void gazebo::physics::LinkState::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Load **LinkState** (p. 618) information from stored data in and SDF::Element.

Parameters

<code>in</code>	<code>_elem</code>	Pointer to the SDF::Element containing state info.
-----------------	--------------------	--

Reimplemented from **gazebo::physics::State** (p. 1070).

10.120.3.13 `LinkState gazebo::physics::LinkState::operator+ (const LinkState & _state) const`

Addition operator.

Parameters

<code>in</code>	<code>_pt</code>	A state to add.
-----------------	------------------	-----------------

Returns

The resulting state.

10.120.3.14 LinkState gazebo::physics::LinkState::operator- (const LinkState & _state) const

Subtraction operator.

Parameters

in	_pt	A state to subtract.
----	-----	----------------------

Returns

The resulting state.

10.120.3.15 LinkState& gazebo::physics::LinkState::operator= (const LinkState & _state)

Assignment operator.

Parameters

in	_state	State (p. 1068) value
----	--------	------------------------------

Returns

this

10.120.3.16 virtual void gazebo::physics::LinkState::SetRealTime (const common::Time & _time) [virtual]

Set the real time when this state was generated.

Parameters

in	_time	Clock time since simulation was stated.
----	-------	---

Reimplemented from **gazebo::physics::State** (p. 1071).

10.120.3.17 virtual void gazebo::physics::LinkState::SetSimTime (const common::Time & _time) [virtual]

Set the sim time when this state was generated.

Parameters

in	_time	Simulation time when the data was recorded.
----	-------	---

Reimplemented from **gazebo::physics::State** (p. 1071).

10.120.3.18 virtual void gazebo::physics::LinkState::SetWallTime (const common::Time & *_time*) [virtual]

Set the wall time when this state was generated.

Parameters

in	<i>_time</i>	The absolute clock time when the State (p. 1068) data was recorded.
----	--------------	--

Reimplemented from **gazebo::physics::State** (p. 1072).

10.120.4 Friends And Related Function Documentation

10.120.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::physics::LinkState & *_state*) [friend]

Stream insertion operator.

Parameters

in	<i>_out</i>	output stream
in	<i>_state</i>	Link (p. 595) state to output

Returns

the stream

Disabling this for efficiency.

Disabling this for efficiency.

The documentation for this class was generated from the following file:

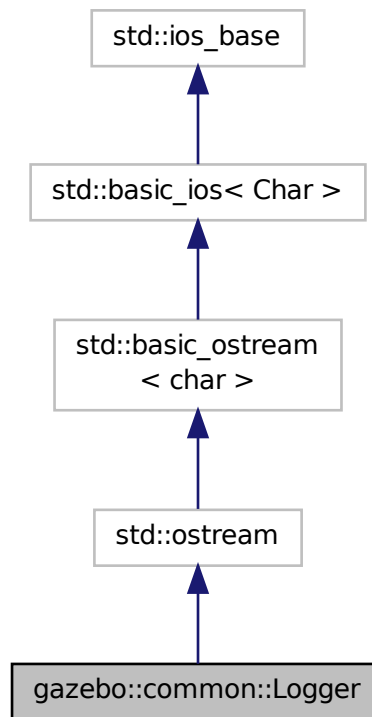
- **LinkState.hh**

10.121 gazebo::common::Logger Class Reference

Terminal logger.

```
#include <Console.hh>
```

Inheritance diagram for gazebo::common::Logger:



Classes

- class **Buffer**
String buffer for the base logger.

Public Types

- enum **LogType** { **STDOUT**, **STDERR** }

Public Member Functions

- **Logger** (const std::string &_prefix, int _color, **LogType** _type)
Constructor.
- virtual **~Logger** ()
Destructor.
- virtual **Logger & operator()** ()

Access operator.

- virtual **Logger & operator()** (const std::string &_file, int _line)
Output a filename and line number, then return a reference to the logger.

Public Attributes

- int **color**
Color (p. 249) for the output.

10.121.1 Detailed Description

Terminal logger.

10.121.2 Member Enumeration Documentation

10.121.2.1 enum gazebo::common::Logger::LogType

Output destination type.

Enumerator:

- STDOUT** Output to stdout.
- STDERR** Output to stderr.

10.121.3 Constructor & Destructor Documentation

10.121.3.1 gazebo::common::Logger::Logger (const std::string & _prefix, int _color, LogType _type)

Constructor.

Parameters

in	<i>_prefix</i>	String to use as prefix when logging to file.
in	<i>_color</i>	Color (p. 249) of the output stream.
in	<i>_type</i>	Output destination type (STDOUT, or STDERR)

10.121.3.2 virtual gazebo::common::Logger::~Logger () [virtual]

Destructor.

10.121.4 Member Function Documentation

10.121.4.1 virtual Logger& gazebo::common::Logger::operator()() [virtual]

Access operator.

Returns

Reference to this logger.

10.121.4.2 `virtual Logger& gazebo::common::Logger::operator() (const std::string & file, int line)` [virtual]

Output a filename and line number, then return a reference to the logger.

Parameters

in	<code><i>_file</i></code>	Filename to output.
in	<code><i>_line</i></code>	Line number in the <code><i>_file</i></code> .

Returns

Reference to this logger.

10.121.5 Member Data Documentation

10.121.5.1 `int gazebo::common::Logger::color`

Color (p. 249) for the output.

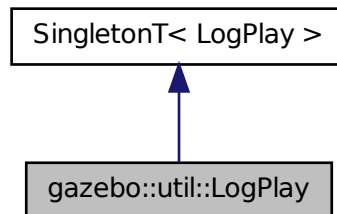
The documentation for this class was generated from the following file:

- **Console.hh**

10.122 gazebo::util::LogPlay Class Reference

```
#include <LogPlay.hh>
```

Inheritance diagram for gazebo::util::LogPlay:

**Public Member Functions**

- bool **GetChunk** (unsigned int `_index`, std::string & `_data`)

- Get data for a particular chunk index.*

 - unsigned int **GetChunkCount** () const

Get the number of chunks (steps) in the open log file.
- std::string **GetEncoding** () const

Get the type of encoding used for current chunk in the open log file.
- std::string **GetGazeboVersion** () const

Get the Gazebo version number of the open log file.
- std::string **GetHeader** () const

Get the header that was read from a log file.
- std::string **GetLogVersion** () const

Get the log version number of the open log file.
- uint32_t **GetRandSeed** () const

Get the random number seed of the open log file.
- bool **IsOpen** () const

Return true if a file is open.
- void **Open** (const std::string &_logFile)

Open a log file for reading.
- bool **Step** (std::string &_data)

Step through the open log file.

Additional Inherited Members

10.122.1 Member Function Documentation

10.122.1.1 bool gazebo::util::LogPlay::GetChunk (unsigned int _index, std::string & _data)

Get data for a particular chunk index.

Parameters

in	<code>_index</code>	Index of the chunk.
out	<code>_data</code>	Storage for the chunk's data.

Returns

True if the `_index` was valid.

10.122.1.2 unsigned int gazebo::util::LogPlay::GetChunkCount () const

Get the number of chunks (steps) in the open log file.

Returns

The number of recorded states in the log file.

10.122.1.3 std::string gazebo::util::LogPlay::GetEncoding () const

Get the type of encoding used for current chunk in the open log file.

Returns

The type of encoding. An empty string will be returned if **LogPlay::Step** (p. 631) has not been called at least once.

10.122.1.4 `std::string gazebo::util::LogPlay::GetGazeboVersion () const`

Get the Gazebo version number of the open log file.

Returns

The Gazebo version of the open log file. Empty string if a log file is not open.

10.122.1.5 `std::string gazebo::util::LogPlay::GetHeader () const`

Get the header that was read from a log file.

Should call **LogPlay::Open** (p. 631) first.

Returns

Header of the open log file.

10.122.1.6 `std::string gazebo::util::LogPlay::GetLogVersion () const`

Get the log version number of the open log file.

Returns

The log version of the open log file. Empty string if a log file is not open.

10.122.1.7 `uint32_t gazebo::util::LogPlay::GetRandSeed () const`

Get the random number seed of the open log file.

Returns

The random number seed the open log file. The current random number seed, as defined in **math::Rand::GetSeed** (p. 839).

10.122.1.8 `bool gazebo::util::LogPlay::IsOpen () const`

Return true if a file is open.

Returns

True if a log file is open.

10.122.1.9 void gazebo::util::LogPlay::Open (const std::string & *_logFile*)

Open a log file for reading.

Open a log file that was previously recorded.

Parameters

in	<i>_logFile</i>	The file to load
----	-----------------	------------------

Exceptions

<i>Exception</i>

10.122.1.10 bool gazebo::util::LogPlay::Step (std::string & *_data*)

Step through the open log file.

Parameters

out	<i>_data</i>	Data from next entry in the log file.
-----	--------------	---------------------------------------

The documentation for this class was generated from the following file:

- **LogPlay.hh**

10.123 Logplay Class Reference

Open and playback log files that were recorded using LogRecord.

10.123.1 Detailed Description

Open and playback log files that were recorded using LogRecord.

Use **Logplay** (p. 631) to open a log file (Logplay::Open), and access the recorded state information. Iterators are available to step through the state information. It is also possible to replay the data in a World using the Play functions. Replay involves reading and applying state information to a World.

See Also

LogRecord, State

The documentation for this class was generated from the following file:

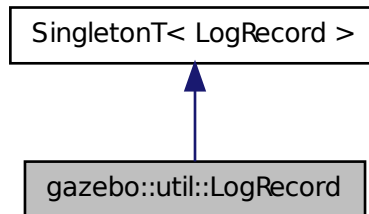
- **LogPlay.hh**

10.124 gazebo::util::LogRecord Class Reference

addtogroup gazebo_util

```
#include <util/util.hh>
```

Inheritance diagram for gazebo::util::LogRecord:



Public Member Functions

- void **Add** (const std::string &_name, const std::string &_filename, boost::function< bool(std::ostream &)> _logCallback)
Add an object to a log file.
- void **Fini** ()
Finalize, and shutdown.
- std::string **GetBasePath** () const
Get the base path for a log recording.
- unsigned int **GetBufferSize** () const
Get the size of the buffer.
- const std::string & **GetEncoding** () const
Get the encoding used.
- std::string **GetFilename** (const std::string &_name="") const
Get the filename for a log object.
- unsigned int **GetFileSize** (const std::string &_name="") const
Get the file size for a log object.
- bool **GetFirstUpdate** () const
Return true if an Update has not yet been completed.
- bool **GetPaused** () const
Get whether logging is paused.
- bool **GetRunning** () const
Get whether logging is running.
- **common::Time GetRunTime** () const
Get the run time in sim time.
- bool **Init** (const std::string &_subdir)
Initialize logging into a subdirectory.
- bool **IsReadyToStart** () const
Get whether the logger is ready to start, which implies that any previous runs have finished.
- void **Notify** ()

Tell the recorder that an update should occur.

- bool **Remove** (const std::string &_name)
Remove an entity from a log.
- void **SetBasePath** (const std::string &_path)
Set the base path.
- void **SetPaused** (bool _paused)
Set whether logging should pause.
- bool **Start** (const std::string &_encoding="zlib", const std::string &_path="")
Start the logger.
- void **Stop** ()
Stop the logger.
- void **Write** (bool _force=false)
Write all logs.

Additional Inherited Members

10.124.1 Detailed Description

addtogroup gazebo_util

Handles logging of data to disk

The **LogRecord** (p. 631) class is a Singleton that manages data logging of any entity within a running simulation. An entity may be a World, Model, or any of their child entities. This class only writes log files, see **LogPlay** (p. 628) for playback functionality.

State information for an entity may be logged through the **LogRecord::Add** (p. 633) function, and stopped through the **LogRecord::Remove** (p. 636) function. Data may be logged into a single file, or split into many separate files by specifying different filenames for the **LogRecord::Add** (p. 633) function.

The **LogRecord** (p. 631) is updated at the start of each simulation step. This guarantees that all data is stored.

See Also

Logplay (p. 631), State

10.124.2 Member Function Documentation

10.124.2.1 void gazebo::util::LogRecord::Add (const std::string & _name, const std::string & _filename, boost::function< bool(std::ostream &)> _logCallback)

Add an object to a log file.

Add a new object to a log. An object can be any valid named object in simulation, including the world itself. Duplicate additions are ignored. Objects can be added to the same file by specifying the same _filename.

Parameters

in	<i>_name</i>	Name of the object to log.
in	<i>_filename</i>	Filename of the log file.
in	<i>_logCallback</i>	Function used to log data for the object. Typically an object will have a log function that outputs data to the provided ostream.

Exceptions

<i>Exception</i>

10.124.2.2 void gazebo::util::LogRecord::Fini ()

Finalize, and shutdown.

10.124.2.3 std::string gazebo::util::LogRecord::GetBasePath () const

Get the base path for a log recording.

Returns

Path for log recording.

10.124.2.4 unsigned int gazebo::util::LogRecord::GetBufferSize () const

Get the size of the buffer.

Returns

Size of the buffer, in bytes.

10.124.2.5 const std::string& gazebo::util::LogRecord::GetEncoding () const

Get the encoding used.

Returns

Either [txt, zlib, or bz2], where txt is plain txt and bz2 and zlib are compressed data with Base64 encoding.

10.124.2.6 std::string gazebo::util::LogRecord::GetFilename (const std::string & _name = " ") const

Get the filename for a log object.

Parameters

in	_name	Name of the log object.
----	-------	-------------------------

Returns

Filename, empty string if not found.

10.124.2.7 unsigned int gazebo::util::LogRecord::GetFileSize (const std::string & _name = " ") const

Get the file size for a log object.

Parameters

in	<i>_name</i>	Name of the log object.
----	--------------	-------------------------

Returns

Size in bytes.

10.124.2.8 bool gazebo::util::LogRecord::GetFirstUpdate () const

Return true if an Update has not yet been completed.

Returns

True if an Update has not yet been completed.

10.124.2.9 bool gazebo::util::LogRecord::GetPaused () const

Get whether logging is paused.

Returns

True if logging is paused.

See Also

LogRecord::SetPaused (p. 636)

10.124.2.10 bool gazebo::util::LogRecord::GetRunning () const

Get whether logging is running.

Returns

True if logging has been started.

10.124.2.11 common::Time gazebo::util::LogRecord::GetRunTime () const

Get the run time in sim time.

Returns

Run sim time.

10.124.2.12 bool gazebo::util::LogRecord::Init (const std::string & *_subdir*)

Initialize logging into a subdirectory.

Init may only be called once, False will be returned if called multiple times.

Parameters

<code>in</code>	<code>_subdir</code>	Directory to record to
-----------------	----------------------	------------------------

Returns

True if successful.

10.124.2.13 `bool gazebo::util::LogRecord::IsReadyToStart () const`

Get whether the logger is ready to start, which implies that any previous runs have finished.

10.124.2.14 `void gazebo::util::LogRecord::Notify ()`

Tell the recorder that an update should occur.

10.124.2.15 `bool gazebo::util::LogRecord::Remove (const std::string & _name)`

Remove an entity from a log.

Removes an entity from the logger. The stops data recording for the entity and all its children. For example, specifying a world will stop all data logging.

Parameters

<code>in</code>	<code>_name</code>	Name of the log
-----------------	--------------------	-----------------

Returns

True if the entity existed and was removed. False if the entity was not registered with the logger.

10.124.2.16 `void gazebo::util::LogRecord::SetBasePath (const std::string & _path)`

Set the base path.

Parameters

<code>in</code>	<code>_path</code>	Path to the new logging location.
-----------------	--------------------	-----------------------------------

10.124.2.17 `void gazebo::util::LogRecord::SetPaused (bool _paused)`

Set whether logging should pause.

A paused state means the log file is still open, but data is not written to it.

Parameters

<code>in</code>	<code>_paused</code>	True to pause data logging.
-----------------	----------------------	-----------------------------

See Also

LogRecord::GetPaused (p. 635)

10.124.2.18 `bool gazebo::util::LogRecord::Start (const std::string & _encoding = "zlib", const std::string & _path = " ")`

Start the logger.

Parameters

in	<i>_encoding</i>	The type of encoding (txt, zlib, or bz2).
in	<i>_path</i>	Path in which to store log files.

10.124.2.19 `void gazebo::util::LogRecord::Stop ()`

Stop the logger.

10.124.2.20 `void gazebo::util::LogRecord::Write (bool _force = false)`

Write all logs.

Parameters

in	<i>_force</i>	True to skip waiting on dataAvailableCondition.
----	---------------	---

The documentation for this class was generated from the following file:

- **LogRecord.hh**

10.125 gazebo::Master Class Reference

A manager that directs topic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

```
#include <gazebo_core.hh>
```

Public Member Functions

- **Master** ()
Constructor.
- virtual **~Master** ()
Destructor.
- void **Fini** ()
Finalize the master.
- void **Init** (uint16_t *_port*)
Initialize.
- void **Run** ()
Run the master.

- void **RunOnce** ()
Run the master one iteration.
- void **RunThread** ()
Run the master in a new thread.
- void **Stop** ()
Stop the master.

10.125.1 Detailed Description

A manager that directs topic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

Base class for simulation server that handles commandline options, starts a **Master** (p.637), runs World update and sensor generation loops.

10.125.2 Constructor & Destructor Documentation

10.125.2.1 gazebo::Master::Master ()

Constructor.

10.125.2.2 virtual gazebo::Master::~Master () [virtual]

Destructor.

10.125.3 Member Function Documentation

10.125.3.1 void gazebo::Master::Fini ()

Finalize the master.

10.125.3.2 void gazebo::Master::Init (uint16_t *_port*)

Initialize.

Parameters

<i>in</i>	<i>_port</i>	The master's port
-----------	--------------	-------------------

10.125.3.3 void gazebo::Master::Run ()

Run the master.

10.125.3.4 void gazebo::Master::RunOnce ()

Run the master one iteration.

10.125.3.5 void gazebo::Master::RunThread ()

Run the master in a new thread.

10.125.3.6 void gazebo::Master::Stop ()

Stop the master.

The documentation for this class was generated from the following file:

- **Master.hh**

10.126 gazebo::common::Material Class Reference

Encapsulates description of a material.

```
#include <common/common.hh>
```

Public Types

- enum **BlendMode** { **ADD**, **MODULATE**, **REPLACE**, **BLEND_COUNT** }
- enum **ShadeMode** { **FLAT**, **GOURAUD**, **PHONG**, **BLINN**, **SHADE_COUNT** }

Public Member Functions

- **Material** ()
Constructor.
- **Material** (const **Color** &_clr)
Create a material with a default color.
- virtual ~**Material** ()
Destructor.
- **Color GetAmbient** () const
Get the ambient color.
- void **GetBlendFactors** (double &_srcFactor, double &_dstFactor)
Get the blend factors.
- **BlendMode GetBlendMode** () const
Get the blending mode.
- bool **GetDepthWrite** () const
Get depth write.
- **Color GetDiffuse** () const
Get the diffuse color.
- **Color GetEmissive** () const
Get the emissive color.
- bool **GetLighting** () const
Get lighting enabled.
- std::string **GetName** () const

- Get the name of the material.*

 - double **GetPointSize** () const

Get the point size.
- **ShadeMode GetShadeMode** () const
- Get the shading mode.*

 - double **GetShininess** () const

Get the shininess.
- **Color GetSpecular** () const
- Get the specular color.*

 - std::string **GetTextureImage** () const

Get a texture image.
- double **GetTransparency** () const
- Get the transparency percentage (0..1)*

 - void **SetAmbient** (const **Color** &_clr)

Set the ambient color.
- void **SetBlendFactors** (double _srcFactor, double _dstFactor)
- Set the blende factors.*

 - void **SetBlendMode** (**BlendMode** _b)

Set the blending mode.
- void **SetDepthWrite** (bool _value)
- Set depth write.*

 - void **SetDiffuse** (const **Color** &_clr)

Set the diffuse color.
- void **SetEmissive** (const **Color** &_clr)
- Set the emissive color.*

 - void **SetLighting** (bool _value)

Set lighting enabled.
- void **SetPointSize** (double _size)
- Set the point size.*

 - void **SetShadeMode** (**ShadeMode** _b)

Set the shading mode param[in] the shading mode.
- void **SetShininess** (double _t)
- Set the shininess.*

 - void **SetSpecular** (const **Color** &_clr)

Set the specular color.
- void **SetTextureImage** (const std::string &_tex)
- Set a texture image.*

 - void **SetTextureImage** (const std::string &_tex, const std::string &_resourcePath)

Set a texture image.
- void **SetTransparency** (double _t)
- Set the transparency percentage (0..1)*

Static Public Attributes

- static std::string **BlendModeStr** [**BLEND_COUNT**]
- static std::string **ShadeModeStr** [**SHADE_COUNT**]

Protected Attributes

- **Color ambient**
the ambient light color
- **BlendMode blendMode**
blend mode
- **Color diffuse**
the diffuse lighth color
- **Color emissive**
the emissive light color
- **std::string name**
the name of the material
- **double pointSize**
point size
- **ShadeMode shadeMode**
the shade mode
- **double shininess**
shininess value (0 to 1)
- **Color specular**
the specular light color
- **std::string texImage**
the texture image file name
- **double transparency**
transparency value in the range 0 to 1

Friends

- **std::ostream & operator<<** (std::ostream &_out, const **gazebo::common::Material** &_m)
Stream insertion operator param[in] _out the output stream to extract from param[out] _m the material information.

10.126.1 Detailed Description

Encapsulates description of a material.

10.126.2 Member Enumeration Documentation

10.126.2.1 enum gazebo::common::Material::BlendMode

Enumerator:

ADD
MODULATE
REPLACE
BLEND_COUNT

10.126.2.2 enum gazebo::common::Material::ShadeMode

Enumerator:

FLAT
GOURAUD
PHONG
BLINN
SHADE_COUNT

10.126.3 Constructor & Destructor Documentation

10.126.3.1 gazebo::common::Material::Material ()

Constructor.

10.126.3.2 virtual gazebo::common::Material::~~Material () [virtual]

Destructor.

10.126.3.3 gazebo::common::Material::Material (const Color & _clr)

Create a material with a default color.

Parameters

in	_clr	Color (p. 249) of the material
----	------	---------------------------------------

10.126.4 Member Function Documentation

10.126.4.1 Color gazebo::common::Material::GetAmbient () const

Get the ambient color.

Returns

The ambient color

10.126.4.2 void gazebo::common::Material::GetBlendFactors (double & _srcFactor, double & _dstFactor)

Get the blend factors.

Parameters

in	_srcFactor	Source factor is returned in this variable
in	_dstFactor	Destination factor is returned in this variable

10.126.4.3 BlendMode gazebo::common::Material::GetBlendMode () const

Get the blending mode.

Returns

the blend mode

10.126.4.4 bool gazebo::common::Material::GetDepthWrite () const

Get depth write.

Returns

the depth write enabled state

10.126.4.5 Color gazebo::common::Material::GetDiffuse () const

Get the diffuse color.

Returns

The diffuse color

10.126.4.6 Color gazebo::common::Material::GetEmissive () const

Get the emissive color.

Returns

The emissive color

10.126.4.7 bool gazebo::common::Material::GetLighting () const

Get lighting enabled.

Returns

the lighting enabled state

10.126.4.8 std::string gazebo::common::Material::GetName () const

Get the name of the material.

Returns

The name of the material

10.126.4.9 `double gazebo::common::Material::GetPointSize () const`

Get the point size.

Returns

the point size

10.126.4.10 `ShadeMode gazebo::common::Material::GetShadeMode () const`

Get the shading mode.

Returns

the shading mode

10.126.4.11 `double gazebo::common::Material::GetShininess () const`

Get the shininess.

Returns

The shininess value

10.126.4.12 `Color gazebo::common::Material::GetSpecular () const`

Get the specular color.

Returns

The specular color

10.126.4.13 `std::string gazebo::common::Material::GetTextureImage () const`

Get a texture image.

Returns

The name of the texture image (if one exists) or an empty string

10.126.4.14 `double gazebo::common::Material::GetTransparency () const`

Get the transparency percentage (0..1)

Returns

The transparency percentage

10.126.4.15 void gazebo::common::Material::SetAmbient (const Color & *_clr*)

Set the ambient color.

Parameters

in	<i>_clr</i>	The ambient color
----	-------------	-------------------

10.126.4.16 void gazebo::common::Material::SetBlendFactors (double *_srcFactor*, double *_dstFactor*)

Set the blende factors.

Will be interpreted as: (texture * *_srcFactor*) + (scene_pixel * *_dstFactor*)

Parameters

in	<i>_srcFactor</i>	The source factor
in	<i>_dstFactor</i>	The destination factor

10.126.4.17 void gazebo::common::Material::SetBlendMode (BlendMode *_b*)

Set the blending mode.

Parameters

in	<i>_b</i>	the blend mode
----	-----------	----------------

10.126.4.18 void gazebo::common::Material::SetDepthWrite (bool *_value*)

Set depth write.

Parameters

in	<i>_value</i>	the depth write enabled state
----	---------------	-------------------------------

10.126.4.19 void gazebo::common::Material::SetDiffuse (const Color & *_clr*)

Set the diffuse color.

Parameters

in	<i>_clr</i>	The diffuse color
----	-------------	-------------------

10.126.4.20 void gazebo::common::Material::SetEmissive (const Color & *_clr*)

Set the emissive color.

Parameters

in	<code>_clr</code>	The emissive color
----	-------------------	--------------------

10.126.4.21 `void gazebo::common::Material::SetLighting (bool _value)`

Set lighting enabled.

Parameters

in	<code>_value</code>	the lighting enabled state
----	---------------------	----------------------------

10.126.4.22 `void gazebo::common::Material::SetPointSize (double _size)`

Set the point size.

Parameters

in	<code>_size</code>	the size
----	--------------------	----------

10.126.4.23 `void gazebo::common::Material::SetShadeMode (ShadeMode _b)`

Set the shading mode param[in] the shading mode.

10.126.4.24 `void gazebo::common::Material::SetShininess (double _t)`

Set the shininess.

Parameters

in	<code>_t</code>	The shininess value
----	-----------------	---------------------

10.126.4.25 `void gazebo::common::Material::SetSpecular (const Color & _clr)`

Set the specular color.

Parameters

in	<code>_clr</code>	The specular color
----	-------------------	--------------------

10.126.4.26 `void gazebo::common::Material::SetTextureImage (const std::string & _tex)`

Set a texture image.

Parameters

in	<code>_tex</code>	The name of the texture, which must be in Gazebo's resource path
----	-------------------	--

10.126.4.27 `void gazebo::common::Material::SetTextureImage (const std::string & _tex, const std::string & _resourcePath)`

Set a texture image.

Parameters

<code>in</code>	<code>_tex</code>	The name of the texture
<code>in</code>	<code>_resourcePath</code>	Path which contains <code>_tex</code>

10.126.4.28 `void gazebo::common::Material::SetTransparency (double _t)`

Set the transparency percentage (0..1)

Parameters

<code>in</code>	<code>_t</code>	The amount of transparency (0..1)
-----------------	-----------------	-----------------------------------

10.126.5 Friends And Related Function Documentation

10.126.5.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::common::Material & _m)` [`friend`]

Stream insertion operator param[in] `_out` the output stream to extract from param[out] `_m` the material information.

10.126.6 Member Data Documentation

10.126.6.1 **Color** `gazebo::common::Material::ambient` [`protected`]

the ambient light color

10.126.6.2 **BlendMode** `gazebo::common::Material::blendMode` [`protected`]

blend mode

10.126.6.3 `std::string gazebo::common::Material::BlendModeStr[BLEND_COUNT]` [`static`]

10.126.6.4 **Color** `gazebo::common::Material::diffuse` [`protected`]

the diffuse lighth color

10.126.6.5 **Color** `gazebo::common::Material::emissive` [`protected`]

the emissive light color

10.126.6.6 `std::string gazebo::common::Material::name` [`protected`]

the name of the material

10.126.6.7 `double gazebo::common::Material::pointSize` [protected]

point size

10.126.6.8 `ShadeMode gazebo::common::Material::shadeMode` [protected]

the shade mode

10.126.6.9 `std::string gazebo::common::Material::ShadeModeStr[SHADE_COUNT]` [static]

10.126.6.10 `double gazebo::common::Material::shininess` [protected]

shininess value (0 to 1)

10.126.6.11 `Color gazebo::common::Material::specular` [protected]

the specular light color

10.126.6.12 `std::string gazebo::common::Material::texImage` [protected]

the texture image file name

10.126.6.13 `double gazebo::common::Material::transparency` [protected]

transparency value in the range 0 to 1

The documentation for this class was generated from the following file:

- `common/Material.hh`

10.127 gazebo::math::Matrix3 Class Reference

A 3x3 matrix class.

```
#include <Matrix3.hh>
```

Public Member Functions

- **Matrix3** ()
Constructor.
- **Matrix3** (const **Matrix3** &_m)
Copy constructor.
- **Matrix3** (double _v00, double _v01, double _v02, double _v10, double _v11, double _v12, double _v20, double _v21, double _v22)
Constructor.
- virtual **~Matrix3** ()
Desctructor.

- **Matrix3 operator*** (const double &_s) const
returns the element wise scalar multiplication
- **Matrix3 operator*** (const **Matrix3** &_m) const
Matrix multiplication operator.
- **math::Vector3 operator*** (const **math::Vector3** &_v) const
*Matrix times **Vector3** (p. 1165) operator.*
- **Matrix3 operator+** (const **Matrix3** &_m) const
returns the element wise sum of two matrices
- **Matrix3 operator-** (const **Matrix3** &_m) const
returns the element wise difference of two matrices
- bool **operator==** (const **Matrix3** &_m) const
Equality test operator.
- const double * **operator[]** (size_t _row) const
Array subscript operator.
- double * **operator[]** (size_t _row)
Array subscript operator.
- void **SetCol** (unsigned int _c, const **Vector3** &_v)
Set a column.
- void **SetFromAxes** (const **Vector3** &_xAxis, const **Vector3** &_yAxis, const **Vector3** &_zAxis)
Set the matrix from three axis (1 per column)
- void **SetFromAxis** (const **Vector3** &_axis, double _angle)
Set the matrix from an axis and angle.

Static Public Attributes

- static const **Matrix3 IDENTITY**
Identity matrix.
- static const **Matrix3 ZERO**
Zero matrix.

Protected Attributes

- double **m** [3][3]
the 3x3 matrix

Friends

- **Matrix3 operator*** (double _s, const **Matrix3** &_m)
Multiplication operators.
- std::ostream & **operator<<** (std::ostream &_out, const **gazebo::math::Matrix3** &_m)
Stream insertion operator.

10.127.1 Detailed Description

A 3x3 matrix class.

10.127.2 Constructor & Destructor Documentation

10.127.2.1 gazebo::math::Matrix3::Matrix3 ()

Constructor.

10.127.2.2 gazebo::math::Matrix3::Matrix3 (const Matrix3 & _m)

Copy constructor.

Parameters

_m	Matrix to copy
----	----------------

10.127.2.3 gazebo::math::Matrix3::Matrix3 (double _v00, double _v01, double _v02, double _v10, double _v11, double _v12, double _v20, double _v21, double _v22)

Constructor.

Parameters

in	_v00	Row 0, Col 0 value
in	_v01	Row 0, Col 1 value
in	_v02	Row 0, Col 2 value
in	_v10	Row 1, Col 0 value
in	_v11	Row 1, Col 1 value
in	_v12	Row 1, Col 2 value
in	_v20	Row 2, Col 0 value
in	_v21	Row 2, Col 1 value
in	_v22	Row 2, Col 2 value

10.127.2.4 virtual gazebo::math::Matrix3::~~Matrix3 () [virtual]

Destructor.

10.127.3 Member Function Documentation

10.127.3.1 Matrix3 gazebo::math::Matrix3::operator*(const double & .s) const [inline]

returns the element wise scalar multiplication

10.127.3.2 Matrix3 gazebo::math::Matrix3::operator*(const Matrix3 & _m) const [inline]

Matrix multiplication operator.

Parameters

in	_m	Matrix3 (p. 648) to multiply
----	----	-------------------------------------

Returns

product of this * `_m`

10.127.3.3 `math::Vector3 gazebo::math::Matrix3::operator*(const math::Vector3 & _v) const` `[inline]`

Matrix times **Vector3** (p. 1165) operator.

Parameters

<code>in</code>	<code>_v</code>	a Vector3 (p. 1165)
-----------------	-----------------	----------------------------

Returns

this * `_v`

References gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

10.127.3.4 `Matrix3 gazebo::math::Matrix3::operator+(const Matrix3 & _m) const` `[inline]`

returns the element wise sum of two matrices

10.127.3.5 `Matrix3 gazebo::math::Matrix3::operator-(const Matrix3 & _m) const` `[inline]`

returns the element wise difference of two matrices

10.127.3.6 `bool gazebo::math::Matrix3::operator==(const Matrix3 & _m) const`

Equality test operator.

Parameters

<code>in</code>	<code>_m</code>	Matrix3 (p. 648) to test
-----------------	-----------------	---------------------------------

Returns

True if equal (using the default tolerance of 1e-6)

10.127.3.7 `const double* gazebo::math::Matrix3::operator[](size_t _row) const` `[inline]`

Array subscript operator.

Parameters

<code>in</code>	<code>_row</code>	row index
-----------------	-------------------	-----------

Returns

a pointer to the row

10.127.3.8 `double* gazebo::math::Matrix3::operator[](size_t _row) [inline]`

Array subscript operator.

Parameters

<code>in</code>	<code>_row</code>	row index
-----------------	-------------------	-----------

Returns

a pointer to the row

10.127.3.9 `void gazebo::math::Matrix3::SetCol (unsigned int _c, const Vector3 & _v)`

Set a column.

Parameters

<code>in</code>	<code>_c</code>	The column index (0, 1, 2)
<code>in</code>	<code>_v</code>	The value to set in each row of the column

10.127.3.10 `void gazebo::math::Matrix3::SetFromAxes (const Vector3 & _xAxis, const Vector3 & _yAxis, const Vector3 & _zAxis)`

Set the matrix from three axis (1 per column)

Parameters

<code>in</code>	<code>_xAxis</code>	The x axis
<code>in</code>	<code>_yAxis</code>	The y axis
<code>in</code>	<code>_zAxis</code>	The z axis

10.127.3.11 `void gazebo::math::Matrix3::SetFromAxis (const Vector3 & _axis, double _angle)`

Set the matrix from an axis and angle.

Parameters

<code>in</code>	<code>_axis</code>	the axis
<code>in</code>	<code>_angle</code>	ccw rotation around the axis in radians

10.127.4 Friends And Related Function Documentation

10.127.4.1 `Matrix3 operator* (double _s, const Matrix3 & _m) [friend]`

Multiplication operators.

Parameters

in	<code>_s</code>	the scaling factor
in	<code>_m</code>	input matrix

Returns

a scaled matrix

10.127.4.2 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Matrix3 & _m)` [*friend*]

Stream insertion operator.

Parameters

in	<code>_out</code>	Output stream
in	<code>_m</code>	Matrix to output

Returns

the stream

10.127.5 Member Data Documentation

10.127.5.1 `const Matrix3 gazebo::math::Matrix3::IDENTITY` [*static*]

Identity matrix.

10.127.5.2 `double gazebo::math::Matrix3::m[3][3]` [*protected*]

the 3x3 matrix

10.127.5.3 `const Matrix3 gazebo::math::Matrix3::ZERO` [*static*]

Zero matrix.

The documentation for this class was generated from the following file:

- **Matrix3.hh**

10.128 gazebo::math::Matrix4 Class Reference

A 3x3 matrix class.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Matrix4 ()**

Constructor.

- **Matrix4** (const **Matrix4** &_m)

Copy constructor.

- **Matrix4** (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

Constructor.

- virtual ~**Matrix4** ()

Destructor.

- **math::Pose GetAsPose** () const

*Get the transformation as **math::Pose** (p. 797).*

- **Vector3 GetEulerRotation** (unsigned int solution_number=1) const

Get the rotation as a Euler angles.

- **Quaternion GetRotation** () const

Get the rotation as a quaternion.

- **Vector3 GetTranslation** () const

*Get the translational values as a **Vector3** (p. 1165).*

- **Matrix4 Inverse** () const

Return the inverse matrix.

- bool **IsAffine** () const

Return true if the matrix is affine.

- **Matrix4 operator*** (const **Matrix4** &_mat) const

Multiplication operator.

- **Matrix4 operator*** (const **Matrix3** &_mat) const

Multiplication operator.

- **Vector3 operator*** (const **Vector3** &_vec) const

Multiplication operator.

- **Matrix4 & operator=** (const **Matrix4** &_mat)

Equal operator.

- const **Matrix4 & operator=** (const **Matrix3** &_mat)

Equal operator for 3x3 matrix.

- bool **operator==** (const **Matrix4** &_m) const

Equality operator.

- double * **operator[]** (size_t _row)

Array subscript operator.

- const double * **operator[]** (size_t _row) const

- void **Set** (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

Change the values.

- void **SetScale** (const **Vector3** &_s)

Set the scale.

- void **SetTranslate** (const **Vector3** &_t)

Set the translational values [(0, 3) (1, 3) (2, 3)].

- **Vector3 TransformAffine** (const **Vector3** &_v) const

Perform an affine transformation.

Static Public Attributes

- static const **Matrix4 IDENTITY**
Identity matrix.
- static const **Matrix4 ZERO**
Zero matrix.

Protected Attributes

- double **m** [4][4]
The 4x4 matrix.

Friends

- std::ostream & **operator<<** (std::ostream &_out, const **gazebo::math::Matrix4** &_m)
Stream insertion operator.

10.128.1 Detailed Description

A 3x3 matrix class.

10.128.2 Constructor & Destructor Documentation

10.128.2.1 gazebo::math::Matrix4::Matrix4 ()

Constructor.

10.128.2.2 gazebo::math::Matrix4::Matrix4 (const Matrix4 & _m)

Copy constructor.

Parameters

<code>_m</code>	Matrix to copy
-----------------	----------------

10.128.2.3 gazebo::math::Matrix4::Matrix4 (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

Constructor.

Parameters

<code>in</code>	<code>_v00</code>	Row 0, Col 0 value
<code>in</code>	<code>_v01</code>	Row 0, Col 1 value
<code>in</code>	<code>_v02</code>	Row 0, Col 2 value
<code>in</code>	<code>_v03</code>	Row 0, Col 3 value
<code>in</code>	<code>_v10</code>	Row 1, Col 0 value

in	<code>_v11</code>	Row 1, Col 1 value
in	<code>_v12</code>	Row 1, Col 2 value
in	<code>_v13</code>	Row 1, Col 3 value
in	<code>_v20</code>	Row 2, Col 0 value
in	<code>_v21</code>	Row 2, Col 1 value
in	<code>_v22</code>	Row 2, Col 2 value
in	<code>_v23</code>	Row 2, Col 3 value
in	<code>_v30</code>	Row 3, Col 0 value
in	<code>_v31</code>	Row 3, Col 1 value
in	<code>_v32</code>	Row 3, Col 2 value
in	<code>_v33</code>	Row 3, Col 3 value

10.128.2.4 `virtual gazebo::math::Matrix4::~~Matrix4 () [virtual]`

Destructor.

10.128.3 Member Function Documentation

10.128.3.1 `math::Pose gazebo::math::Matrix4::GetAsPose () const`

Get the transformation as **math::Pose** (p. 797).

Returns

the pose

10.128.3.2 `Vector3 gazebo::math::Matrix4::GetEulerRotation (unsigned int solution_number = 1) const`

Get the rotation as a Euler angles.

Returns

the rotation

10.128.3.3 `Quaternion gazebo::math::Matrix4::GetRotation () const`

Get the rotation as a quaternion.

Returns

the rotation

10.128.3.4 `Vector3 gazebo::math::Matrix4::GetTranslation () const`

Get the translational values as a **Vector3** (p. 1165).

Returns

x,y,z

10.128.3.5 Matrix4 gazebo::math::Matrix4::Inverse () const

Return the inverse matrix.

10.128.3.6 bool gazebo::math::Matrix4::IsAffine () const

Return true if the matrix is affine.

Returns

true if the matrix is affine, false otherwise

10.128.3.7 Matrix4 gazebo::math::Matrix4::operator* (const Matrix4 & *_mat*) const

Multiplication operator.

Parameters

<i>_mat</i>	Incoming matrix
-------------	-----------------

Returns

This matrix * *_mat*

10.128.3.8 Matrix4 gazebo::math::Matrix4::operator* (const Matrix3 & *_mat*) const

Multiplication operator.

Parameters

<i>_mat</i>	Incoming matrix
-------------	-----------------

Returns

This matrix * *_mat*

10.128.3.9 Vector3 gazebo::math::Matrix4::operator* (const Vector3 & *_vec*) const

Multiplication operator.

Parameters

<i>_vec</i>	Vector3 (p. 1165)
-------------	--------------------------

Returns

Resulting vector from multiplication

10.128.3.10 **Matrix4&** gazebo::math::Matrix4::operator= (const Matrix4 & *_mat*)

Equal operator.

this = *_mat*

Parameters

<i>_mat</i>	Incoming matrix
-------------	-----------------

Returns

itself

10.128.3.11 **const Matrix4&** gazebo::math::Matrix4::operator= (const Matrix3 & *_mat*)

Equal operator for 3x3 matrix.

Parameters

<i>_mat</i>	Incoming matrix
-------------	-----------------

Returns

itself

10.128.3.12 **bool** gazebo::math::Matrix4::operator==(const Matrix4 & *_m*) const

Equality operator.

Parameters

<i>in</i>	<i>_m</i>	Matrix3 (p. 648) to test
-----------	-----------	---------------------------------

Returns

true if the 2 matrices are equal (using the tolerance 1e-6), false otherwise

10.128.3.13 **double*** gazebo::math::Matrix4::operator[] (size_t *_row*) [inline]

Array subscript operator.

Parameters

<i>in</i>	<i>_row</i>	the row index
-----------	-------------	---------------

Returns

the row

10.128.3.14 `const double* gazebo::math::Matrix4::operator[](size_t _row) const [inline]`

Parameters

in	<code>_row</code>	the row index
----	-------------------	---------------

Returns

the row

10.128.3.15 `void gazebo::math::Matrix4::Set (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)`

Change the values.

Parameters

in	<code>_v00</code>	Row 0, Col 0 value
in	<code>_v01</code>	Row 0, Col 1 value
in	<code>_v02</code>	Row 0, Col 2 value
in	<code>_v03</code>	Row 0, Col 3 value
in	<code>_v10</code>	Row 1, Col 0 value
in	<code>_v11</code>	Row 1, Col 1 value
in	<code>_v12</code>	Row 1, Col 2 value
in	<code>_v13</code>	Row 1, Col 3 value
in	<code>_v20</code>	Row 2, Col 0 value
in	<code>_v21</code>	Row 2, Col 1 value
in	<code>_v22</code>	Row 2, Col 2 value
in	<code>_v23</code>	Row 2, Col 3 value
in	<code>_v30</code>	Row 3, Col 0 value
in	<code>_v31</code>	Row 3, Col 1 value
in	<code>_v32</code>	Row 3, Col 2 value
in	<code>_v33</code>	Row 3, Col 3 value

10.128.3.16 `void gazebo::math::Matrix4::SetScale (const Vector3 & _s)`

Set the scale.

Parameters

in	<code>_s</code>	scale
----	-----------------	-------

10.128.3.17 `void gazebo::math::Matrix4::SetTranslate (const Vector3 & _t)`

Set the translational values [(0, 3) (1, 3) (2, 3)].

Parameters

in	<code>_t</code>	Values to set
----	-----------------	---------------

10.128.3.18 **Vector3** gazebo::math::Matrix4::TransformAffine (const Vector3 & _v) const

Perform an affine transformation.

Parameters

<code>_v</code>	Vector3 (p. 1165) value for the transformation
-----------------	---

Returns

The result of the transformation

10.128.4 Friends And Related Function Documentation

10.128.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Matrix4 & _m)` [`friend`]

Stream insertion operator.

Parameters

<code>_out</code>	output stream
<code>_m</code>	Matrix to output

Returns

the stream

10.128.5 Member Data Documentation

10.128.5.1 `const Matrix4 gazebo::math::Matrix4::IDENTITY` [`static`]

Identity matrix.

10.128.5.2 `double gazebo::math::Matrix4::m[4][4]` [`protected`]

The 4x4 matrix.

10.128.5.3 `const Matrix4 gazebo::math::Matrix4::ZERO` [`static`]

Zero matrix.

The documentation for this class was generated from the following file:

- **Matrix4.hh**

10.129 gazebo::common::Mesh Class Reference

A 3D mesh.

```
#include <common/common.hh>
```


Public Member Functions

- **Mesh** ()
Constructor.
- virtual \sim **Mesh** ()
Destructor.
- int **AddMaterial** (**Material** * _mat)
Add a material to the mesh.
- void **AddSubMesh** (**SubMesh** * _child)
Add a submesh mesh.
- void **Center** (const **math::Vector3** &_center=**math::Vector3::Zero**)
Move the center of the mesh to the given coordinate.
- void **FillArrays** (float ** _vertArr, int ** _indArr) const
Put all the data into flat arrays.
- void **GenSphericalTexCoord** (const **math::Vector3** &_center)
Generate texture coordinates using spherical projection from center.
- void **GetAABB** (**math::Vector3** &_center, **math::Vector3** &_min_xyz, **math::Vector3** &_max_xyz) const
Get AABB coordinate.
- unsigned int **GetIndexCount** () const
Return the number of indices.
- const **Material** * **GetMaterial** (int _index) const
Get a material.
- unsigned int **GetMaterialCount** () const
Get the number of materials.
- **math::Vector3** **GetMax** () const
Get the maximum X, Y, Z values.
- **math::Vector3** **GetMin** () const
Get the minimum X, Y, Z values.
- std::string **GetName** () const
Get the name of this mesh.
- unsigned int **GetNormalCount** () const
Return the number of normals.
- std::string **GetPath** () const
Get the path which contains the mesh resource.
- **Skeleton** * **GetSkeleton** () const
Get the skeleton to which this mesh is attached.
- const **SubMesh** * **GetSubMesh** (unsigned int _i) const
Get a child mesh.
- const **SubMesh** * **GetSubMesh** (const std::string &_name) const
Get a child mesh by name.
- unsigned int **GetSubMeshCount** () const
Get the number of children.
- unsigned int **GetTexCoordCount** () const
Return the number of texture coordinates.
- unsigned int **GetVertexCount** () const
Return the number of vertices.
- bool **HasSkeleton** () const

Return true if mesh is attached to a skeleton.

- void **RecalculateNormals** ()
Recalculate all the normals of each face defined by three indices.
- void **Scale** (double *_factor*)
*Scale all vertices by *_factor*.*
- void **SetName** (const std::string &*_n*)
Set the name of this mesh.
- void **SetPath** (const std::string &*_path*)
Set the path which contains the mesh resource.
- void **SetScale** (const **math::Vector3** &*_factor*)
*Scale all vertices by the *_factor* vector.*
- void **SetSkeleton** (**Skeleton** **_skel*)
Set the mesh skeleton.
- void **Translate** (const **math::Vector3** &*_vec*)
*Move all vertices in all submeshes by *_vec*.*

10.129.1 Detailed Description

A 3D mesh.

10.129.2 Constructor & Destructor Documentation

10.129.2.1 gazebo::common::Mesh::Mesh ()

Constructor.

10.129.2.2 virtual gazebo::common::Mesh::~~Mesh () [virtual]

Destructor.

10.129.3 Member Function Documentation

10.129.3.1 int gazebo::common::Mesh::AddMaterial (**Material** * *_mat*)

Add a material to the mesh.

Parameters

<i>in</i>	<i>_mat</i>	the material
-----------	-------------	--------------

Returns

Index of this material

10.129.3.2 void gazebo::common::Mesh::AddSubMesh (**SubMesh** * *_child*)

Add a submesh mesh.

The **Mesh** (p. 660) object takes ownership of the submesh.

Parameters

in	<code>_child</code>	the submesh
----	---------------------	-------------

10.129.3.3 `void gazebo::common::Mesh::Center (const math::Vector3 & _center = math::Vector3::Zero)`

Move the center of the mesh to the given coordinate.

This will move all the vertices in all submeshes.

Parameters

in	<code>_center</code>	Location of the mesh center.
----	----------------------	------------------------------

10.129.3.4 `void gazebo::common::Mesh::FillArrays (float ** _vertArr, int ** _indArr) const`

Put all the data into flat arrays.

Parameters

out	<code>_vertArr</code>	the vertex array
out	<code>_indArr</code>	the index array

10.129.3.5 `void gazebo::common::Mesh::GenSphericalTexCoord (const math::Vector3 & _center)`

Generate texture coordinates using spherical projection from center.

Parameters

in	<code>_center</code>	the center of the projection
----	----------------------	------------------------------

10.129.3.6 `void gazebo::common::Mesh::GetAABB (math::Vector3 & _center, math::Vector3 & _min_xyz, math::Vector3 & _max_xyz) const`

Get AABB coordinate.

Parameters

out	<code>_center</code>	of the bounding box
out	<code>_min_xyz</code>	bounding box minimum values
out	<code>_max_xyz</code>	bounding box maximum values

10.129.3.7 `unsigned int gazebo::common::Mesh::GetIndexCount () const`

Return the number of indices.

Returns

the count

10.129.3.8 `const Material* gazebo::common::Mesh::GetMaterial (int _index) const`

Get a material.

Parameters

<code>in</code>	<code><i>_index</i></code>	the index
-----------------	----------------------------	-----------

Returns

the material or NULL if the index is out of bounds

10.129.3.9 `unsigned int gazebo::common::Mesh::GetMaterialCount () const`

Get the number of materials.

Returns

the count

10.129.3.10 `math::Vector3 gazebo::common::Mesh::GetMax () const`

Get the maximum X, Y, Z values.

Returns

the upper bounds of the bounding box

10.129.3.11 `math::Vector3 gazebo::common::Mesh::GetMin () const`

Get the minimum X, Y, Z values.

Returns

the lower bounds of the bounding box

10.129.3.12 `std::string gazebo::common::Mesh::GetName () const`

Get the name of this mesh.

Returns

the name

10.129.3.13 `unsigned int gazebo::common::Mesh::GetNormalCount () const`

Return the number of normals.

Returns

the count

10.129.3.14 `std::string gazebo::common::Mesh::GetPath () const`

Get the path which contains the mesh resource.

Returns

the path to the mesh resource

10.129.3.15 `Skeleton* gazebo::common::Mesh::GetSkeleton () const`

Get the skeleton to which this mesh is attached.

Returns

pointer to skeleton, or NULL if none is present.

10.129.3.16 `const SubMesh* gazebo::common::Mesh::GetSubMesh (unsigned int i) const`

Get a child mesh.

Parameters

<code><i>i</i></code>	<code><i>i</i></code>	the index
-----------------------	-----------------------	-----------

Returns

the submesh. An exception is thrown if the index is out of bounds

10.129.3.17 `const SubMesh* gazebo::common::Mesh::GetSubMesh (const std::string & _name) const`

Get a child mesh by name.

Parameters

<code><i>i</i></code>	<code><i>_name</i></code>	Name of the submesh.
-----------------------	---------------------------	----------------------

Returns

The submesh, NULL if the `_name` is not found.

10.129.3.18 `unsigned int gazebo::common::Mesh::GetSubMeshCount () const`

Get the number of children.

Returns

the count

10.129.3.19 `unsigned int gazebo::common::Mesh::GetTexCoordCount () const`

Return the number of texture coordinates.

Returns

the count

10.129.3.20 `unsigned int gazebo::common::Mesh::GetVertexCount () const`

Return the number of vertices.

Returns

the count

10.129.3.21 `bool gazebo::common::Mesh::HasSkeleton () const`

Return true if mesh is attached to a skeleton.

10.129.3.22 `void gazebo::common::Mesh::RecalculateNormals ()`

Recalculate all the normals of each face defined by three indices.

10.129.3.23 `void gazebo::common::Mesh::Scale (double _factor)`

Scale all vertices by *_factor*.

Parameters

<i>_factor</i>	Scaling factor
----------------	----------------

10.129.3.24 `void gazebo::common::Mesh::SetName (const std::string & _n)`

Set the name of this mesh.

Parameters

<i>in</i>	<i>_n</i>	the name to set
-----------	-----------	-----------------

10.129.3.25 void gazebo::common::Mesh::SetPath (const std::string & *_path*)

Set the path which contains the mesh resource.

Parameters

in	<i>_path</i>	the file path
----	--------------	---------------

10.129.3.26 void gazebo::common::Mesh::SetScale (const math::Vector3 & *_factor*)

Scale all vertices by the *_factor* vector.

Parameters

in	<i>_factor</i>	Scaling vector
----	----------------	----------------

10.129.3.27 void gazebo::common::Mesh::SetSkeleton (Skeleton * *_skel*)

Set the mesh skeleton.

10.129.3.28 void gazebo::common::Mesh::Translate (const math::Vector3 & *_vec*)

Move all vertices in all submeshes by *_vec*.

Parameters

in	<i>_vec</i>	Amount to translate vertices.
----	-------------	-------------------------------

The documentation for this class was generated from the following file:

- **Mesh.hh**

10.130 gazebo::common::MeshCSG Class Reference

Creates CSG meshes.

```
#include <common/common.hh>
```

Public Types

- enum **BooleanOperation** { **UNION**, **INTERSECTION**, **DIFFERENCE** }
An enumeration of the boolean operations.

Public Member Functions

- **MeshCSG** ()
Constructor.
- virtual **~MeshCSG** ()

Destructor.

- **Mesh * CreateBoolean** (const **Mesh** *_m1, const **Mesh** *_m2, const int _operation, const **math::Pose** & _offset=**math::Pose::Zero**)

Create a boolean mesh from two meshes.

10.130.1 Detailed Description

Creates CSG meshes.

10.130.2 Member Enumeration Documentation

10.130.2.1 enum gazebo::common::MeshCSG::BooleanOperation

An enumeration of the boolean operations.

Enumerator:

UNION

INTERSECTION

DIFFERENCE

10.130.3 Constructor & Destructor Documentation

10.130.3.1 gazebo::common::MeshCSG::MeshCSG ()

Constructor.

10.130.3.2 virtual gazebo::common::MeshCSG::~~MeshCSG () [virtual]

Destructor.

10.130.4 Member Function Documentation

10.130.4.1 Mesh* gazebo::common::MeshCSG::CreateBoolean (const Mesh * _m1, const Mesh * _m2, const int _operation, const math::Pose & _offset = math::Pose::Zero)

Create a boolean mesh from two meshes.

Parameters

in	<i>_m1</i>	the parent mesh in the boolean operation
in	<i>_m2</i>	the child mesh in the boolean operation
in	<i>_operation</i>	the boolean operation applied to the two meshes
in	<i>_offset</i>	_m2's pose offset from _m1

Returns

a pointer to the created mesh

The documentation for this class was generated from the following file:

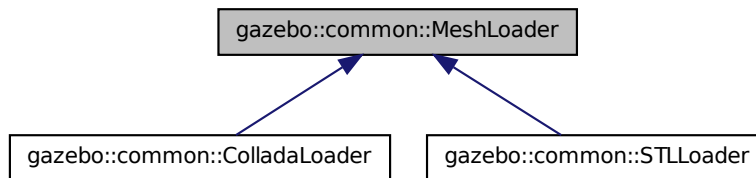
- [MeshCSG.hh](#)

10.131 gazebo::common::MeshLoader Class Reference

Base class for loading meshes.

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::common::MeshLoader:



Public Member Functions

- **MeshLoader** ()
Constructor.
- virtual **~MeshLoader** ()
Destructor.
- virtual **Mesh * Load** (const std::string &_filename)=0
Load a 3D mesh.

10.131.1 Detailed Description

Base class for loading meshes.

10.131.2 Constructor & Destructor Documentation

10.131.2.1 gazebo::common::MeshLoader::MeshLoader ()

Constructor.

10.131.2.2 virtual gazebo::common::MeshLoader::~~MeshLoader () [virtual]

Destructor.

10.131.3 Member Function Documentation

10.131.3.1 `virtual Mesh* gazebo::common::MeshLoader::Load (const std::string & _filename) [pure virtual]`

Load a 3D mesh.

Parameters

in	_filename	the path to the mesh
----	-----------	----------------------

Returns

a pointer to the created mesh

Implemented in `gazebo::common::ColladaLoader` (p. 235), and `gazebo::common::STLLoader` (p. 1073).

The documentation for this class was generated from the following file:

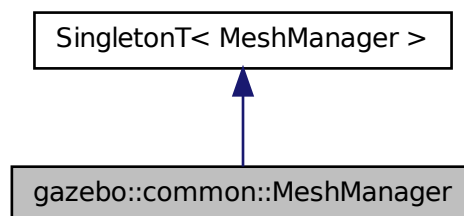
- `MeshLoader.hh`

10.132 gazebo::common::MeshManager Class Reference

Maintains and manages all meshes.

```
#include <common/common.hh>
```

Inheritance diagram for `gazebo::common::MeshManager`:



Public Member Functions

- void **AddMesh** (**Mesh** *_mesh)
Add a mesh to the manager.
- void **CreateBox** (const std::string &_name, const **math::Vector3** &_sides, const **math::Vector2d** &_uvCoords)
Create a Box mesh.
- void **CreateCamera** (const std::string &_name, float _scale)
Create a Camera mesh.
- void **CreateCone** (const std::string &_name, float _radius, float _height, int _rings, int _segments)

Create a cone mesh.

- void **CreateCylinder** (const std::string &_name, float _radius, float _height, int _rings, int _segments)

Create a cylinder mesh.

- void **CreatePlane** (const std::string &_name, const **math::Plane** &_plane, const **math::Vector2d** &_segments, const **math::Vector2d** &_uvTile)

Create mesh for a plane.

- void **CreatePlane** (const std::string &_name, const **math::Vector3** &_normal, double _d, const **math::Vector2d** &_size, const **math::Vector2d** &_segments, const **math::Vector2d** &_uvTile)

Create mesh for a plane.

- void **CreateSphere** (const std::string &_name, float _radius, int _rings, int _segments)

Create a sphere mesh.

- void **CreateTube** (const std::string &_name, float _innerRadius, float _outterRadius, float _height, int _rings, int _segments)

Create a tube mesh.

- void **GenSphericalTexCoord** (const **Mesh** *_mesh, **math::Vector3** _center)

generate spherical texture coordinates

- const **Mesh** * **GetMesh** (const std::string &_name) const

Get a mesh by name.

- void **GetMeshAABB** (const **Mesh** *_mesh, **math::Vector3** &_center, **math::Vector3** &_min_xyz, **math::Vector3** &_max_xyz)

Get mesh aabb and center.

- bool **HasMesh** (const std::string &_name) const

Return true if the mesh exists.

- bool **IsValidFilename** (const std::string &_filename)

Checks a path extension against the list of valid extensions.

- const **Mesh** * **Load** (const std::string &_filename)

Load a mesh from a file.

Additional Inherited Members

10.132.1 Detailed Description

Maintains and manages all meshes.

10.132.2 Member Function Documentation

10.132.2.1 void gazebo::common::MeshManager::AddMesh (**Mesh** * _mesh)

Add a mesh to the manager.

This **MeshManager** (p. 670) takes ownership of the mesh and will destroy it. See `~MeshManager`.

Parameters

<i>in</i>	<i>the</i>	mesh to add.
-----------	------------	--------------

10.132.2.2 void gazebo::common::MeshManager::CreateBox (const std::string & *_name*, const math::Vector3 & *_sides*, const math::Vector2d & *_uvCoords*)

Create a Box mesh.

Parameters

in	<i>_name</i>	the name of the new mesh
in	<i>_sides</i>	the x y x dimentions of eah side in meter
in	<i>_uvCoords</i>	the texture coordinates

10.132.2.3 void gazebo::common::MeshManager::CreateCamera (const std::string & *_name*, float *_scale*)

Create a Camera mesh.

Parameters

in	<i>_name</i>	name of the new mesh
in	<i>_scale</i>	scaling factor for the camera

10.132.2.4 void gazebo::common::MeshManager::CreateCone (const std::string & *_name*, float *_radius*, float *_height*, int *_rings*, int *_segments*)

Create a cone mesh.

Parameters

in	<i>_name</i>	the name of the new mesh
in	<i>_radius</i>	the radius of the cylinder in the x y plane
in	<i>_height</i>	the height along z
in	<i>_rings</i>	the number of circles along the height
in	<i>_segments</i>	the number of segment per circle

10.132.2.5 void gazebo::common::MeshManager::CreateCylinder (const std::string & *_name*, float *_radius*, float *_height*, int *_rings*, int *_segments*)

Create a cylinder mesh.

Parameters

in	<i>_name</i>	the name of the new mesh
in	<i>_radius</i>	the radius of the cylinder in the x y plane
in	<i>_height</i>	the height along z
in	<i>_rings</i>	the number of circles along the height
in	<i>_segments</i>	the number of segment per circle

10.132.2.6 void gazebo::common::MeshManager::CreatePlane (const std::string & *_name*, const math::Plane & *_plane*, const math::Vector2d & *_segments*, const math::Vector2d & *_uvTile*)

Create mesh for a plane.

Parameters

in	<i>_name</i>	
in	<i>_plane</i>	plane parameters
in	<i>_segments</i>	number of segments in x and y
in	<i>_uvTile</i>	the texture tile size in x and y

10.132.2.7 void gazebo::common::MeshManager::CreatePlane (const std::string & *_name*, const math::Vector3 & *_normal*, double *_d*, const math::Vector2d & *_size*, const math::Vector2d & *_segments*, const math::Vector2d & *_uvTile*)

Create mesh for a plane.

Parameters

in	<i>_name</i>	the name of the new mesh
in	<i>_normal</i>	the normal to the plane
in	<i>_d</i>	distance from the origin along normal
in	<i>_size</i>	the size of the plane in x and y
in	<i>_segments</i>	the number of segments in x and y
in	<i>_uvTile</i>	the texture tile size in x and y

10.132.2.8 void gazebo::common::MeshManager::CreateSphere (const std::string & *_name*, float *_radius*, int *_rings*, int *_segments*)

Create a sphere mesh.

Parameters

in	<i>_name</i>	the name of the mesh
in	<i>_radius</i>	radius of the sphere in meter
in	<i>_rings</i>	number of circles on th y axis
in	<i>_segments</i>	number of segment per circle

10.132.2.9 void gazebo::common::MeshManager::CreateTube (const std::string & *_name*, float *_innerRadius*, float *_outerRadius*, float *_height*, int *_rings*, int *_segments*)

Create a tube mesh.

Generates rings inside and outside the cylinder Needs at least two rings and 3 segments

Parameters

in	<i>_name</i>	the name of the new mesh
in	<i>_innerRadius</i>	the inner radius of the tube in the x y plane
in	<i>_outerRadius</i>	the outer radius of the tube in the x y plane
in	<i>_height</i>	the height along z
in	<i>_rings</i>	the number of circles along the height
in	<i>_segments</i>	the number of segment per circle

10.132.2.10 `void gazebo::common::MeshManager::GenSphericalTexCoord (const Mesh * _mesh, math::Vector3 _center)`

generate spherical texture coordinates

10.132.2.11 `const Mesh* gazebo::common::MeshManager::GetMesh (const std::string & _name) const`

Get a mesh by name.

Parameters

in	<code>_name</code>	the name of the mesh to look for
----	--------------------	----------------------------------

Returns

the mesh or NULL if not found

10.132.2.12 `void gazebo::common::MeshManager::GetMeshAABB (const Mesh * _mesh, math::Vector3 & _center, math::Vector3 & _min_xyz, math::Vector3 & _max_xyz)`

Get mesh aabb and center.

Parameters

in	<code>_mesh</code>	the mesh
out	<code>_center</code>	the AAB center position
out	<code>_min_xyz</code>	the bounding box minimum
out	<code>_max_xyz</code>	the bounding box maximum

10.132.2.13 `bool gazebo::common::MeshManager::HasMesh (const std::string & _name) const`

Return true if the mesh exists.

Parameters

in	<code>_name</code>	the name of the mesh
----	--------------------	----------------------

10.132.2.14 `bool gazebo::common::MeshManager::IsValidFilename (const std::string & _filename)`

Checks a path extension against the list of valid extensions.

Returns

true if the file extension is loadable

10.132.2.15 `const Mesh* gazebo::common::MeshManager::Load (const std::string & _filename)`

Load a mesh from a file.

Parameters

in	<code>_filename</code>	the path to the mesh
----	------------------------	----------------------

Returns

a pointer to the created mesh

The documentation for this class was generated from the following file:

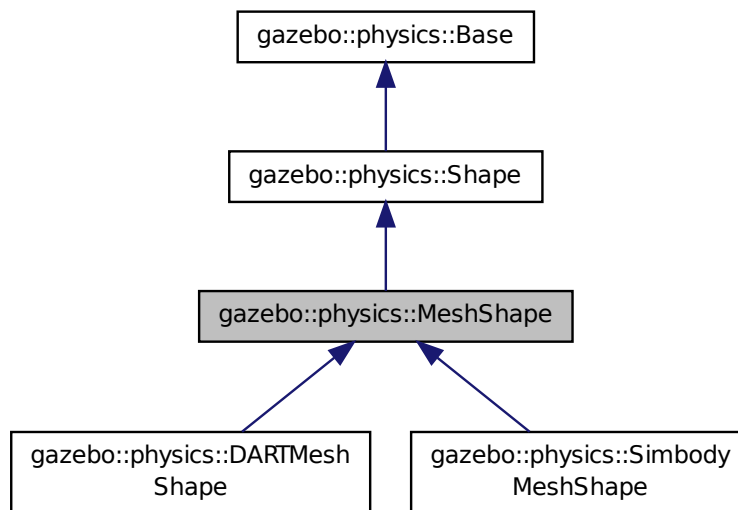
- **MeshManager.hh**

10.133 gazebo::physics::MeshShape Class Reference

Triangle mesh collision shape.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::MeshShape:



Public Member Functions

- **MeshShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~MeshShape** ()
Destructor.
- void **FillMsg** (msgs::Geometry &_msg)
Populate a msgs::Geometry message with data from this shape.

- `std::string GetMeshURI ()` const
Get the URI of the mesh data.
- virtual `math::Vector3 GetSize ()` const
Get the size of the triangle mesh.
- virtual void `Init ()`
Initialize the shape.
- virtual void `ProcessMsg (const msgs::Geometry &_msg)`
Update this shape from a message.
- void `SetMesh (const std::string &_uri, const std::string &_submesh="", bool _center=false)`
Set the mesh uri and submesh name.
- void `SetScale (const math::Vector3 &_scale)`
Set the scaling factor.
- virtual void `Update ()`
Update the tri mesh.

Protected Attributes

- const `common::Mesh * mesh`
Pointer to the mesh data.
- `common::SubMesh * submesh`
The submesh to use from within the parent mesh.

Additional Inherited Members

10.133.1 Detailed Description

Triangle mesh collision shape.

10.133.2 Constructor & Destructor Documentation

10.133.2.1 `gazebo::physics::MeshShape::MeshShape (CollisionPtr _parent)` [explicit]

Constructor.

Parameters

in	<code>_parent</code>	Parent collision.
----	----------------------	-------------------

10.133.2.2 `virtual gazebo::physics::MeshShape::~MeshShape ()` [virtual]

Destructor.

10.133.3 Member Function Documentation

10.133.3.1 `void gazebo::physics::MeshShape::FillMsg (msgs::Geometry & _msg)` [virtual]

Populate a `msgs::Geometry` message with data from this shape.

Parameters

out	<code>_msg</code>	Message to fill.
-----	-------------------	------------------

Implements **gazebo::physics::Shape** (p. 934).

10.133.3.2 `std::string gazebo::physics::MeshShape::GetMeshURI () const`

Get the URI of the mesh data.

Returns

The URI of the mesh data.

10.133.3.3 `virtual math::Vector3 gazebo::physics::MeshShape::GetSize () const [virtual]`

Get the size of the triangle mesh.

Returns

The size of the triangle mesh.

10.133.3.4 `virtual void gazebo::physics::MeshShape::Init () [virtual]`

Initialize the shape.

Implements **gazebo::physics::Shape** (p. 935).

Reimplemented in **gazebo::physics::SimbodyMeshShape** (p. 982), and **gazebo::physics::DARTMeshShape** (p. 349).

10.133.3.5 `virtual void gazebo::physics::MeshShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]`

Update this shape from a message.

Parameters

in	<code>_msg</code>	Message that contains triangle mesh info.
----	-------------------	---

Implements **gazebo::physics::Shape** (p. 935).

10.133.3.6 `void gazebo::physics::MeshShape::SetMesh (const std::string & _uri, const std::string & _submesh = "", bool _center = false)`

Set the mesh uri and submesh name.

Parameters

in	<code>_uri</code>	Filename of the mesh file to load from.
in	<code>_submesh</code>	Name of the submesh to use within the mesh
in	<code>_center</code>	True to center the submesh. specified in the <code>_uri</code> .

10.133.3.7 `void gazebo::physics::MeshShape::SetScale (const math::Vector3 & _scale) [virtual]`

Set the scaling factor.

Parameters

<code>in</code>	<code>_scale</code>	Scaling factor.
-----------------	---------------------	-----------------

Implements `gazebo::physics::Shape` (p. 935).

10.133.3.8 `virtual void gazebo::physics::MeshShape::Update () [inline],[virtual]`

Update the tri mesh.

Reimplemented from `gazebo::physics::Base` (p. 180).

Reimplemented in `gazebo::physics::DARTMeshShape` (p. 350).

10.133.4 Member Data Documentation

10.133.4.1 `const common::Mesh* gazebo::physics::MeshShape::mesh [protected]`

Pointer to the mesh data.

10.133.4.2 `common::SubMesh* gazebo::physics::MeshShape::submesh [protected]`

The submesh to use from within the parent mesh.

The documentation for this class was generated from the following file:

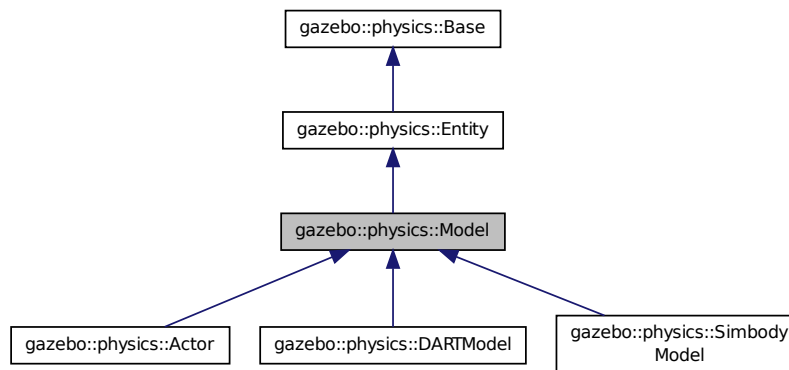
- `MeshShape.hh`

10.134 gazebo::physics::Model Class Reference

A model is a collection of links, joints, and plugins.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::Model:



Public Member Functions

- **Model** (**BasePtr** _parent)
Constructor.
- virtual **~Model** ()
Destructor.
- void **AttachStaticModel** (**ModelPtr** &_model, **math::Pose** _offset)
Attach a static model to this model.
- void **DetachStaticModel** (const std::string &_model)
Detach a static model from this model.
- virtual void **FillMsg** (msgs::Model &_msg)
Fill a model message.
- virtual void **Fini** ()
Finalize the model.
- bool **GetAutoDisable** () const
Return the value of the SDF <allow_auto_disable> element.
- virtual **math::Box** **GetBoundingBox** () const
Get the size of the bounding box.
- **GripperPtr** **GetGripper** (size_t _index) const
Get a gripper based on an index.
- size_t **GetGripperCount** () const
Get the number of grippers in this model.
- **JointPtr** **GetJoint** (const std::string &name)
Get a joint.
- **JointControllerPtr** **GetJointController** ()
Get a handle to the Controller for the joints in this model.
- unsigned int **GetJointCount** () const
Get the number of joints.
- const **Joint_V** & **GetJoints** () const

Get the joints.

- **LinkPtr GetLink** (const std::string &_name="canonical") const

Get a link by name.

- **LinkPtr GetLinkById** (unsigned int _id) const

This is an internal function

- **Link_V GetLinks** () const

*Construct and return a vector of **Link** (p. 595)'s in this model Note this constructs the vector of **Link** (p. 595)'s on the fly, could be costly.*

- unsigned int **GetPluginCount** () const

Get the number of plugins this model has.

- virtual **math::Vector3 GetRelativeAngularAccel** () const

Get the angular acceleration of the entity.

- virtual **math::Vector3 GetRelativeAngularVel** () const

Get the angular velocity of the entity.

- virtual **math::Vector3 GetRelativeLinearAccel** () const

Get the linear acceleration of the entity.

- virtual **math::Vector3 GetRelativeLinearVel** () const

Get the linear velocity of the entity.

- virtual const sdf::ElementPtr **GetSDF** ()

Get the SDF values for the model.

- unsigned int **GetSensorCount** () const

Get the number of sensors attached to this model.

- virtual **math::Vector3 GetWorldAngularAccel** () const

Get the angular acceleration of the entity in the world frame.

- virtual **math::Vector3 GetWorldAngularVel** () const

Get the angular velocity of the entity in the world frame.

- double **GetWorldEnergy** () const

*Returns this model's total energy, or sum of **Model::GetWorldEnergyPotential()** (p. 687) and **Model::GetWorldEnergyKinetic()** (p. 687).*

- double **GetWorldEnergyKinetic** () const

Returns sum of the kinetic energies of all links in this model.

- double **GetWorldEnergyPotential** () const

Returns the potential energy of all links and joint springs in the model.

- virtual **math::Vector3 GetWorldLinearAccel** () const

Get the linear acceleration of the entity in the world frame.

- virtual **math::Vector3 GetWorldLinearVel** () const

Get the linear velocity of the entity in the world frame.

- virtual void **Init** ()

Initialize the model.

- void **Load** (sdf::ElementPtr _sdf)

Load the model.

- void **LoadJoints** ()

Load all the joints.

- void **LoadPlugins** ()

Load all plugins.

- void **ProcessMsg** (const msgs::Model &_msg)

Update parameters from a model message.

- virtual void **RemoveChild** (**EntityPtr** _child)
 - Remove a child.*
- void **Reset** ()
 - Reset the model.*
- void **SetAngularAccel** (const **math::Vector3** &_vel)
 - Set the angular acceleration of the model, and all its links.*
- void **SetAngularVel** (const **math::Vector3** &_vel)
 - Set the angular velocity of the model, and all its links.*
- void **SetAutoDisable** (bool _disable)
 - Allow the model the auto disable.*
- void **SetCollideMode** (const std::string &_mode)
 - This is not implemented in **Link** (p. 595), which means this function doesn't do anything.*
- void **SetEnabled** (bool _enabled)
 - Enable all the links in all the models.*
- void **SetGravityMode** (const bool &_value)
 - Set the gravity mode of the model.*
- void **SetJointAnimation** (const std::map< std::string, **common::NumericAnimationPtr** > &_anims, boost::function< void()> _onComplete=NULL)
 - Joint** (p. 541) Animation.*
- void **SetJointPosition** (const std::string &_jointName, double _position, int _index=0)
 - Set the positions of a **Joint** (p. 541) by name.*
- void **SetJointPositions** (const std::map< std::string, double > &_jointPositions)
 - Set the positions of a set of joints.*
- void **SetLaserRetro** (const float _retro)
 - Set the laser retro reflectiveness of the model.*
- void **SetLinearAccel** (const **math::Vector3** &_vel)
 - Set the linear acceleration of the model, and all its links.*
- void **SetLinearVel** (const **math::Vector3** &_vel)
 - Set the linear velocity of the model, and all its links.*
- void **SetLinkWorldPose** (const **math::Pose** &_pose, std::string _linkName)
 - Set the Pose of the entire **Model** (p. 678) by specifying desired Pose of a **Link** (p. 595) within the **Model** (p. 678).*
- void **SetLinkWorldPose** (const **math::Pose** &_pose, const **LinkPtr** &_link)
 - Set the Pose of the entire **Model** (p. 678) by specifying desired Pose of a **Link** (p. 595) within the **Model** (p. 678).*
- void **SetScale** (const **math::Vector3** &_scale)
 - Set the scale of model.*
- void **SetState** (const **ModelState** &_state)
 - Set the current model state.*
- virtual void **StopAnimation** ()
 - Stop the current animations.*
- void **Update** ()
 - Update the model.*
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)
 - Update the parameters using new sdf values.*

Protected Member Functions

- virtual void **OnPoseChange** ()
 - Callback when the pose of the model has been changed.*

Protected Attributes

- `std::vector< ModelPtr > attachedModels`
used by `Model::AttachStaticModel` (p. 682)
- `std::vector< math::Pose > attachedModelsOffset`
used by `Model::AttachStaticModel` (p. 682)
- `transport::PublisherPtr jointPub`
Publisher for joint info.

Additional Inherited Members

10.134.1 Detailed Description

A model is a collection of links, joints, and plugins.

10.134.2 Constructor & Destructor Documentation

10.134.2.1 `gazebo::physics::Model::Model (BasePtr _parent)` [explicit]

Constructor.

Parameters

in	<code>_parent</code>	Parent object.
----	----------------------	----------------

10.134.2.2 `virtual gazebo::physics::Model::~~Model ()` [virtual]

Destructor.

10.134.3 Member Function Documentation

10.134.3.1 `void gazebo::physics::Model::AttachStaticModel (ModelPtr & _model, math::Pose _offset)`

Attach a static model to this model.

This function takes as input a static **Model** (p. 678), which is a **Model** (p. 678) that has been marked as static (no physics simulation), and attaches it to this **Model** (p. 678) with a given offset.

This function is useful when you want to simulate a grasp of a static object, or move a static object around using a dynamic model.

If you are in doubt, do not use this function.

Parameters

in	<code>_model</code>	Pointer to the static model.
in	<code>_offset</code>	Offset, relative to this Model (p. 678), to place <code>_model</code> .

10.134.3.2 `void gazebo::physics::Model::DetachStaticModel (const std::string & _model)`

Detach a static model from this model.

Parameters

<code>in</code>	<code><i>_model</i></code>	Name of an attached static model to remove.
-----------------	----------------------------	---

See Also

Model::AttachStaticModel (p. 682).

10.134.3.3 `virtual void gazebo::physics::Model::FillMsg (msgs::Model & _msg) [virtual]`

Fill a model message.

Parameters

<code>in</code>	<code><i>_msg</i></code>	Message to fill using this model's data.
-----------------	--------------------------	--

10.134.3.4 `virtual void gazebo::physics::Model::Fini () [virtual]`

Finalize the model.

Reimplemented from **gazebo::physics::Entity** (p. 407).

Reimplemented in **gazebo::physics::Actor** (p. 142), and **gazebo::physics::DARTModel** (p. 351).

10.134.3.5 `bool gazebo::physics::Model::GetAutoDisable () const`

Return the value of the SDF `<allow_auto_disable>` element.

Returns

True if auto disable is allowed for this model.

10.134.3.6 `virtual math::Box gazebo::physics::Model::GetBoundingBox () const [virtual]`

Get the size of the bounding box.

Returns

The bounding box.

Reimplemented from **gazebo::physics::Entity** (p. 407).

10.134.3.7 `GripperPtr gazebo::physics::Model::GetGripper (size_t _index) const`

Get a gripper based on an index.

Returns

A pointer to a **Gripper** (p. 492). Null if the `_index` is invalid.

10.134.3.8 `size_t gazebo::physics::Model::GetGripperCount () const`

Get the number of grippers in this model.

Returns

Size of this->grippers array.

See Also

Model::GetGripper() (p. 683)

10.134.3.9 `JointPtr gazebo::physics::Model::GetJoint (const std::string & name)`

Get a joint.

Parameters

<i>name</i>	The name of the joint, specified in the world file
-------------	--

Returns

Pointer to the joint

10.134.3.10 `JointControllerPtr gazebo::physics::Model::GetJointController ()`

Get a handle to the Controller for the joints in this model.

Returns

A handle to the Controller for the joints in this model.

10.134.3.11 `unsigned int gazebo::physics::Model::GetJointCount () const`

Get the number of joints.

Returns

Get the number of joints.

10.134.3.12 `const Joint_V& gazebo::physics::Model::GetJoints () const`

Get the joints.

Returns

Vector of joints.

10.134.3.13 `LinkPtr gazebo::physics::Model::GetLink (const std::string & _name = "canonical") const`

Get a link by name.

Parameters

<code>in</code>	<code>_name</code>	Name of the link to get.
-----------------	--------------------	--------------------------

Returns

Pointer to the link, NULL if the name is invalid.

10.134.3.14 `LinkPtr gazebo::physics::Model::GetLinkById (unsigned int _id) const`

This is an internal function

Get a link by id.

Returns

Pointer to the link, NULL if the id is invalid.

10.134.3.15 `Link_V gazebo::physics::Model::GetLinks () const`

Construct and return a vector of **Link** (p. 595)'s in this model Note this constructs the vector of **Link** (p. 595)'s on the fly, could be costly.

Returns

a vector of **Link** (p. 595)'s in this model

10.134.3.16 `unsigned int gazebo::physics::Model::GetPluginCount () const`

Get the number of plugins this model has.

Returns

Number of plugins associated with this model.

10.134.3.17 `virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularAccel () const` [virtual]

Get the angular acceleration of the entity.

Returns

math::Vector3 (p. 1165), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 409).

10.134.3.18 `virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularVel () const` [virtual]

Get the angular velocity of the entity.

Returns

math::Vector3 (p. 1165), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 409).

10.134.3.19 `virtual math::Vector3 gazebo::physics::Model::GetRelativeLinearAccel () const` [virtual]

Get the linear acceleration of the entity.

Returns

math::Vector3 (p. 1165), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 409).

10.134.3.20 `virtual math::Vector3 gazebo::physics::Model::GetRelativeLinearVel () const` [virtual]

Get the linear velocity of the entity.

Returns

math::Vector3 (p. 1165), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 410).

10.134.3.21 `virtual const sdf::ElementPtr gazebo::physics::Model::GetSDF ()` [virtual]

Get the SDF values for the model.

Returns

The SDF value for this model.

Reimplemented from **gazebo::physics::Base** (p. 176).

Reimplemented in **gazebo::physics::Actor** (p. 142).

10.134.3.22 `unsigned int gazebo::physics::Model::GetSensorCount () const`

Get the number of sensors attached to this model.

This will count all the sensors attached to all the links.

Returns

Number of sensors.

10.134.3.23 `virtual math::Vector3 gazebo::physics::Model::GetWorldAngularAccel () const [virtual]`

Get the angular acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 1165), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 410).

10.134.3.24 `virtual math::Vector3 gazebo::physics::Model::GetWorldAngularVel () const [virtual]`

Get the angular velocity of the entity in the world frame.

Returns

math::Vector3 (p. 1165), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 410).

10.134.3.25 `double gazebo::physics::Model::GetWorldEnergy () const`

Returns this model's total energy, or sum of **Model::GetWorldEnergyPotential()** (p. 687) and **Model::GetWorldEnergyKinetic()** (p. 687).

Returns

this link's total energy

10.134.3.26 `double gazebo::physics::Model::GetWorldEnergyKinetic () const`

Returns sum of the kinetic energies of all links in this model.

Computed using link's CoG velocity in the inertial (world) frame.

Returns

this link's kinetic energy

10.134.3.27 `double gazebo::physics::Model::GetWorldEnergyPotential () const`

Returns the potential energy of all links and joint springs in the model.

Returns

this link's potential energy,

10.134.3.28 `virtual math::Vector3 gazebo::physics::Model::GetWorldLinearAccel () const [virtual]`

Get the linear acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 1165), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 411).

10.134.3.29 `virtual math::Vector3 gazebo::physics::Model::GetWorldLinearVel () const [virtual]`

Get the linear velocity of the entity in the world frame.

Returns

math::Vector3 (p. 1165), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 411).

10.134.3.30 `virtual void gazebo::physics::Model::Init () [virtual]`

Initialize the model.

Reimplemented from **gazebo::physics::Base** (p. 177).

Reimplemented in **gazebo::physics::Actor** (p. 142), **gazebo::physics::DARTModel** (p. 352), and **gazebo::physics::SimbodyModel** (p. 984).

10.134.3.31 `void gazebo::physics::Model::Load (sdf::ElementPtr _sdf) [virtual]`

Load the model.

Parameters

<code>in</code>	<code>_sdf</code>	SDF parameters to load from.
-----------------	-------------------	------------------------------

Reimplemented from **gazebo::physics::Entity** (p. 411).

Reimplemented in **gazebo::physics::Actor** (p. 142), **gazebo::physics::DARTModel** (p. 352), and **gazebo::physics::SimbodyModel** (p. 984).

10.134.3.32 `void gazebo::physics::Model::LoadJoints ()`

Load all the joints.

10.134.3.33 `void gazebo::physics::Model::LoadPlugins ()`

Load all plugins.

Load all plugins specified in the SDF for the model.

10.134.3.34 `virtual void gazebo::physics::Model::OnPoseChange () [protected],[virtual]`

Callback when the pose of the model has been changed.

Implements **gazebo::physics::Entity** (p. 412).

10.134.3.35 `void gazebo::physics::Model::ProcessMsg (const msgs::Model & _msg)`

Update parameters from a model message.

Parameters

in	_msg	Message to process.
----	------	---------------------

10.134.3.36 `virtual void gazebo::physics::Model::RemoveChild (EntityPtr _child) [virtual]`

Remove a child.

Parameters

in	_child	Remove a child entity.
----	--------	------------------------

10.134.3.37 `void gazebo::physics::Model::Reset () [virtual]`

Reset the model.

Reimplemented from **gazebo::physics::Entity** (p. 412).

10.134.3.38 `void gazebo::physics::Model::SetAngularAccel (const math::Vector3 & _vel)`

Set the angular acceleration of the model, and all its links.

Parameters

in	_vel	The new angular acceleration
----	------	------------------------------

10.134.3.39 `void gazebo::physics::Model::SetAngularVel (const math::Vector3 & _vel)`

Set the angular velocity of the model, and all its links.

Parameters

in	_vel	The new angular velocity.
----	------	---------------------------

10.134.3.40 `void gazebo::physics::Model::SetAutoDisable (bool _disable)`

Allow the model the auto disable.

This is ignored if the model has joints.

Parameters

in	<code>_disable</code>	If true, the model is allowed to auto disable.
----	-----------------------	--

10.134.3.41 `void gazebo::physics::Model::SetCollideMode (const std::string & _mode)`

This is not implemented in **Link** (p. 595), which means this function doesn't do anything.

Set the collide mode of the model.

Parameters

in	<code>_mode</code>	The collision mode
----	--------------------	--------------------

10.134.3.42 `void gazebo::physics::Model::SetEnabled (bool _enabled)`

Enable all the links in all the models.

Parameters

in	<code>_enabled</code>	True to enable all the links.
----	-----------------------	-------------------------------

10.134.3.43 `void gazebo::physics::Model::SetGravityMode (const bool & _value)`

Set the gravity mode of the model.

Parameters

in	<code>_value</code>	False to turn gravity on for the model.
----	---------------------	---

10.134.3.44 `void gazebo::physics::Model::SetJointAnimation (const std::map< std::string, common::NumericAnimationPtr > & _anims, boost::function< void()> _onComplete = NULL)`

Joint (p. 541) Animation.

Parameters

in	<code>_anim</code>	Map of joint names to their position animation.
in	<code>_onComplete</code>	Callback function for when the animation completes.

10.134.3.45 `void gazebo::physics::Model::SetJointPosition (const std::string & _jointName, double _position, int _index = 0)`

Set the positions of a **Joint** (p. 541) by name.

See Also

JointController::SetJointPosition (p. 571)

Parameters

in	<code>_jointName</code>	Name of the joint to set.
in	<code>_position</code>	Position to set the joint to.

10.134.3.46 `void gazebo::physics::Model::SetJointPositions (const std::map< std::string, double > & _jointPositions)`

Set the positions of a set of joints.

See Also

JointController::SetJointPositions (p. 571).

Parameters

in	<code>_jointPositions</code>	Map of joint names to their positions.
----	------------------------------	--

10.134.3.47 `void gazebo::physics::Model::SetLaserRetro (const float _retro)`

Set the laser retro reflectiveness of the model.

Parameters

in	<code>_retro</code>	Retro reflectance value.
----	---------------------	--------------------------

10.134.3.48 `void gazebo::physics::Model::SetLinearAccel (const math::Vector3 & _vel)`

Set the linear acceleration of the model, and all its links.

Parameters

in	<code>_vel</code>	The new linear acceleration.
----	-------------------	------------------------------

10.134.3.49 `void gazebo::physics::Model::SetLinearVel (const math::Vector3 & _vel)`

Set the linear velocity of the model, and all its links.

Parameters

in	<code>_vel</code>	The new linear velocity.
----	-------------------	--------------------------

10.134.3.50 `void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & _pose, std::string _linkName)`

Set the Pose of the entire **Model** (p. 678) by specifying desired Pose of a **Link** (p. 595) within the **Model** (p. 678).

Doing so, keeps the configuration of the **Model** (p. 678) unchanged, i.e. all **Joint** (p. 541) angles are unchanged.

Parameters

in	<code>_pose</code>	Pose to set the link to.
in	<code>_linkName</code>	Name of the link to set.

10.134.3.51 `void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & _pose, const LinkPtr & _link)`

Set the Pose of the entire **Model** (p. 678) by specifying desired Pose of a **Link** (p. 595) within the **Model** (p. 678). Doing so, keeps the configuration of the **Model** (p. 678) unchanged, i.e. all **Joint** (p. 541) angles are unchanged.

Parameters

in	<code>_pose</code>	Pose to set the link to.
in	<code>_link</code>	Pointer to the link to set.

10.134.3.52 `void gazebo::physics::Model::SetScale (const math::Vector3 & _scale)`

Set the scale of model.

Parameters

in	<code>_scale</code>	Scale to set the model to.
----	---------------------	----------------------------

10.134.3.53 `void gazebo::physics::Model::SetState (const ModelState & _state)`

Set the current model state.

Parameters

in	<code>_state</code>	State (p. 1068) to set the model to.
----	---------------------	---

10.134.3.54 `virtual void gazebo::physics::Model::StopAnimation () [virtual]`

Stop the current animations.

Reimplemented from **gazebo::physics::Entity** (p. 414).

10.134.3.55 `void gazebo::physics::Model::Update () [virtual]`

Update the model.

Reimplemented from **gazebo::physics::Base** (p. 180).

Reimplemented in **gazebo::physics::Actor** (p. 143), and **gazebo::physics::DARTModel** (p. 352).

10.134.3.56 `virtual void gazebo::physics::Model::UpdateParameters (sdf::ElementPtr _sdf) [virtual]`

Update the parameters using new sdf values.

Parameters

in	<code>_sdf</code>	SDF values to update from.
----	-------------------	----------------------------

Reimplemented from `gazebo::physics::Entity` (p. 414).

Reimplemented in `gazebo::physics::Actor` (p. 143).

10.134.4 Member Data Documentation

10.134.4.1 `std::vector<ModelPtr> gazebo::physics::Model::attachedModels` [protected]

used by `Model::AttachStaticModel` (p. 682)

10.134.4.2 `std::vector<math::Pose> gazebo::physics::Model::attachedModelsOffset` [protected]

used by `Model::AttachStaticModel` (p. 682)

10.134.4.3 `transport::PublisherPtr gazebo::physics::Model::jointPub` [protected]

Publisher for joint info.

The documentation for this class was generated from the following file:

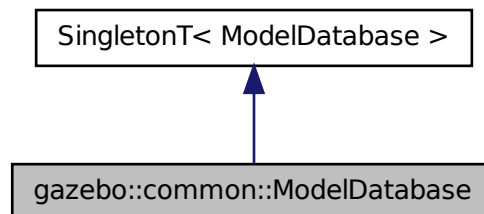
- `Model.hh`

10.135 gazebo::common::ModelDatabase Class Reference

Connects to model database, and has utility functions to find models.

```
#include <common/common.hh>
```

Inheritance diagram for `gazebo::common::ModelDatabase`:



Public Member Functions

- void `DownloadDependencies` (const std::string &_path)

Download all dependencies for a give model path.

- void **Fini** ()
 - Finalize the model database.*
- std::string **GetDBConfig** (const std::string &_uri)
 - Return the database.config file as a string.*
- std::string **GetModelConfig** (const std::string &_uri)
 - Return the model.config file as a string.*
- std::string **GetModelFile** (const std::string &_uri)
 - Get a model's SDF file based on a URI.*
- std::string **GetModelName** (const std::string &_uri)
 - Get the name of a model based on a URI.*
- std::string **GetModelPath** (const std::string &_uri, bool _forceDownload=false)
 - Get the local path to a model.*
- std::map< std::string, std::string > **GetModels** ()
 - Returns the dictionary of all the model names.*
- event::ConnectionPtr **GetModels** (boost::function< void(const std::map< std::string, std::string > &)> _func)
 - Get the dictionary of all model names via a callback.*
- std::string **GetURI** ()
 - Returns the the global model database URI.*
- bool **HasModel** (const std::string &_modelName)
 - Returns true if the model exists on the database.*
- void **Start** (bool _fetchImmediately=false)
 - Start the model database.*

Additional Inherited Members

10.135.1 Detailed Description

Connects to model database, and has utility functions to find models.

The documentation for this class was generated from the following file:

- **ModelDatabase.hh**

10.136 gazebo::common::ModelDatabasePrivate Class Reference

Private class attributes for **ModelDatabase** (p. 693).

```
#include <ModelDatabasePrivate.hh>
```

Public Types

- typedef boost::function< void(const std::map< std::string, std::string > &)> **CallbackFunc**

Public Attributes

- boost::mutex **callbacksMutex**
Protects callback list.
- std::list< **CallbackFunc** > **deprecatedCallbacks**
- std::map< std::string, std::string > **modelCache**
A dictionary of all model names indexed by their uri.
- **event::EventT**< void(std::map< std::string, std::string >)> **modelDBUpdated**
*Triggered when the model data has been updated after calling **ModelDatabase::GetModels()** (p. 44)*
- boost::recursive_mutex **startCacheMutex**
Mutex to protect cache thread status checks.
- bool **stop**
True to stop the background thread.
- boost::condition_variable **updateCacheCompleteCondition**
Condition variable for completion of one cache update.
- boost::condition_variable **updateCacheCondition**
Condition variable for the updateCacheThread.
- boost::thread * **updateCacheThread**
Thread to update the model cache.
- boost::mutex **updateMutex**
Cache update mutex.

10.136.1 Detailed Description

Private class attributes for **ModelDatabase** (p. 693).

10.136.2 Member Typedef Documentation

10.136.2.1 `typedef boost::function< void (const std::map<std::string, std::string> &)> gazebo::common::ModelDatabasePrivate::CallbackFunc`

10.136.3 Member Data Documentation

10.136.3.1 `boost::mutex gazebo::common::ModelDatabasePrivate::callbacksMutex`

Protects callback list.

10.136.3.2 `std::list<CallbackFunc> gazebo::common::ModelDatabasePrivate::deprecatedCallbacks`

10.136.3.3 `std::map<std::string, std::string> gazebo::common::ModelDatabasePrivate::modelCache`

A dictionary of all model names indexed by their uri.

10.136.3.4 `event::EventT< void (std::map<std::string, std::string>)> gazebo::common::ModelDatabasePrivate::modelDBUpdated`

Triggered when the model data has been updated after calling **ModelDatabase::GetModels()** (p. 44)

10.136.3.5 `boost::recursive_mutex gazebo::common::ModelDatabasePrivate::startCacheMutex`

Mutex to protect cache thread status checks.

10.136.3.6 `bool gazebo::common::ModelDatabasePrivate::stop`

True to stop the background thread.

10.136.3.7 `boost::condition_variable gazebo::common::ModelDatabasePrivate::updateCacheCompleteCondition`

Condition variable for completion of one cache update.

10.136.3.8 `boost::condition_variable gazebo::common::ModelDatabasePrivate::updateCacheCondition`

Condition variable for the updateCacheThread.

10.136.3.9 `boost::thread* gazebo::common::ModelDatabasePrivate::updateCacheThread`

Thread to update the model cache.

10.136.3.10 `boost::mutex gazebo::common::ModelDatabasePrivate::updateMutex`

Cache update mutex.

The documentation for this class was generated from the following file:

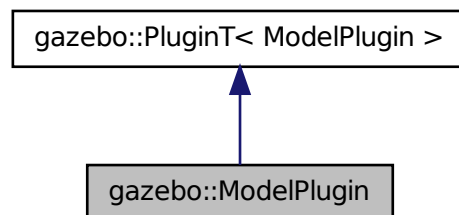
- **ModelDatabasePrivate.hh**

10.137 gazebo::ModelPlugin Class Reference

A plugin with access to `physics::Model` (p. 678).

```
#include <Plugin.hh>
```

Inheritance diagram for gazebo::ModelPlugin:



Public Member Functions

- **ModelPlugin** ()
Constructor.
- virtual **~ModelPlugin** ()
Destructor.
- virtual void **Init** ()
Override this method for custom plugin initialization behavior.
- virtual void **Load** (**physics::ModelPtr** _model, sdf::ElementPtr _sdf)=0
Load function.
- virtual void **Reset** ()
Override this method for custom plugin reset behavior.

Additional Inherited Members

10.137.1 Detailed Description

A plugin with access to **physics::Model** (p. 678).

See [reference](#).

10.137.2 Constructor & Destructor Documentation

10.137.2.1 gazebo::ModelPlugin::ModelPlugin () [inline]

Constructor.

References [gazebo::MODEL_PLUGIN](#).

10.137.2.2 virtual gazebo::ModelPlugin::~~ModelPlugin () [inline],[virtual]

Destructor.

10.137.3 Member Function Documentation

10.137.3.1 virtual void gazebo::ModelPlugin::Init () [inline],[virtual]

Override this method for custom plugin initialization behavior.

10.137.3.2 virtual void gazebo::ModelPlugin::Load (**physics::ModelPtr** _model, sdf::ElementPtr _sdf) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

in	<code>_model</code>	Pointer to the Model
in	<code>_sdf</code>	Pointer to the SDF element of the plugin.

10.137.3.3 `virtual void gazebo::ModelPlugin::Reset () [inline],[virtual]`

Override this method for custom plugin reset behavior.

The documentation for this class was generated from the following file:

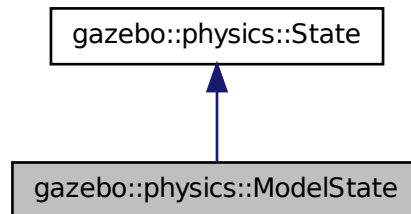
- **Plugin.hh**

10.138 gazebo::physics::ModelState Class Reference

Store state information of a **physics::Model** (p. 678) object.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::ModelState:



Public Member Functions

- **ModelState** ()
Default constructor.
- **ModelState** (const **ModelPtr** _model, const **common::Time** &_realTime, const **common::Time** &_simTime)
Constructor.
- **ModelState** (const **ModelPtr** _model)
Constructor.
- **ModelState** (const sdf::ElementPtr _sdf)
Constructor.
- virtual **~ModelState** ()
Destructor.
- void **FillSDF** (sdf::ElementPtr _sdf)
Populate a state SDF element with data from the object.
- **JointState GetJointState** (unsigned int _index) const
*Get a **Joint** (p. 541) state.*
- **JointState GetJointState** (const std::string &_jointName) const
*Get a **Joint** (p. 541) state by **Joint** (p. 541) name.*
- unsigned int **GetJointStateCount** () const

Get the number of joint states.

- **JointState_M GetJointStates** (const boost::regex &_regex) const
Get joint states based on a regular expression.
- const **JointState_M & GetJointStates** () const
Get the joint states.
- **LinkState GetLinkState** (const std::string &_linkName) const
*Get a link state by **Link** (p. 595) name.*
- unsigned int **GetLinkStateCount** () const
Get the number of link states.
- **LinkState_M GetLinkStates** (const boost::regex &_regex) const
Get link states based on a regular expression.
- const **LinkState_M & GetLinkStates** () const
Get the link states.
- const **math::Pose & GetPose** () const
Get the stored model pose.
- bool **HasJointState** (const std::string &_jointName) const
Return true if there is a joint with the specified name.
- bool **HasLinkState** (const std::string &_linkName) const
Return true if there is a link with the specified name.
- bool **IsZero** () const
Return true if the values in the state are zero.
- void **Load** (const **ModelPtr** _model, const **common::Time** &_realTime, const **common::Time** &_simTime)
*Load state from **Model** (p. 678) pointer.*
- virtual void **Load** (const sdf::ElementPtr _elem)
Load state from SDF element.
- **ModelState operator+** (const **ModelState** &_state) const
Addition operator.
- **ModelState operator-** (const **ModelState** &_state) const
Subtraction operator.
- **ModelState & operator=** (const **ModelState** &_state)
Assignment operator.
- virtual void **SetRealTime** (const **common::Time** &_time)
Set the real time when this state was generated.
- virtual void **SetSimTime** (const **common::Time** &_time)
Set the sim time when this state was generated.
- virtual void **SetWallTime** (const **common::Time** &_time)
Set the wall time when this state was generated.

Friends

- std::ostream & **operator<<** (std::ostream &_out, const **gazebo::physics::ModelState** &_state)
Stream insertion operator.

Additional Inherited Members

10.138.1 Detailed Description

Store state information of a **physics::Model** (p. 678) object.

This class captures the entire state of a **Model** (p. 678) at one specific time during a simulation run.

State (p. 1068) of a **Model** (p. 678) includes the state of all its child Links and Joints.

10.138.2 Constructor & Destructor Documentation

10.138.2.1 gazebo::physics::ModelState::ModelState ()

Default constructor.

10.138.2.2 gazebo::physics::ModelState::ModelState (const ModelPtr _model, const common::Time & _realTime, const common::Time & _simTime)

Constructor.

Build a **ModelState** (p. 698) from an existing **Model** (p. 678).

Parameters

in	<i>_model</i>	Pointer to the model from which to gather state info.
in	<i>_realTime</i>	Real time stamp.
in	<i>_simTime</i>	Sim time stamp.

10.138.2.3 gazebo::physics::ModelState::ModelState (const ModelPtr _model) [explicit]

Constructor.

Build a **ModelState** (p. 698) from an existing **Model** (p. 678).

Parameters

in	<i>_model</i>	Pointer to the model from which to gather state info.
----	---------------	---

10.138.2.4 gazebo::physics::ModelState::ModelState (const sdf::ElementPtr _sdf) [explicit]

Constructor.

Build a **ModelState** (p. 698) from SDF data

Parameters

in	<i>_sdf</i>	SDF data to load a model state from.
----	-------------	--------------------------------------

10.138.2.5 virtual gazebo::physics::ModelState::~~ModelState () [virtual]

Destructor.

10.138.3 Member Function Documentation

10.138.3.1 void gazebo::physics::ModelState::FillSDF (sdf::ElementPtr _sdf)

Populate a state SDF element with data from the object.

Parameters

out	<code>_sdf</code>	SDF element to populate.
-----	-------------------	--------------------------

10.138.3.2 JointState gazebo::physics::ModelState::GetJointState (unsigned int _index) const

Get a **Joint** (p. 541) state.

Return a **JointState** (p. 574) based on a index, where index is between 0...**ModelState::GetJointStateCount()** (p. 702).

Parameters

in	<code>_index</code>	Index of a JointState (p. 574).
----	---------------------	--

Returns

State (p. 1068) of a **Joint** (p. 541).

Exceptions

common::Exception (p. 444)	When <code>_index</code> is out of range.
--------------------------------------	---

10.138.3.3 JointState gazebo::physics::ModelState::GetJointState (const std::string & _jointName) const

Get a **Joint** (p. 541) state by **Joint** (p. 541) name.

Searches through all JointStates. Returns the **JointState** (p. 574) with the matching name, if any.

Parameters

in	<code>_jointName</code>	Name of the JointState (p. 574).
----	-------------------------	---

Returns

State (p. 1068) of the **Joint** (p. 541).

Exceptions

common::Exception (p. 444)	When <code>_jointName</code> is invalid.
--------------------------------------	--

10.138.3.4 `unsigned int gazebo::physics::ModelState::GetJointStateCount () const`

Get the number of joint states.

Returns the number of JointStates recorded.

Returns

Number of JointStates.

10.138.3.5 `JointState_M gazebo::physics::ModelState::GetJointStates (const boost::regex & _regex) const`

Get joint states based on a regular expression.

Parameters

<code>in</code>	<code>_regex</code>	The regular expression.
-----------------	---------------------	-------------------------

Returns

List of joint states whose names match the regular expression.

10.138.3.6 `const JointState_M& gazebo::physics::ModelState::GetJointStates () const`

Get the joint states.

Returns

A map of joint states.

10.138.3.7 `LinkState gazebo::physics::ModelState::GetLinkState (const std::string & _linkName) const`

Get a link state by **Link** (p. 595) name.

Searches through all LinkStates. Returns the **LinkState** (p. 618) with the matching name, if any.

Parameters

<code>in</code>	<code>_linkName</code>	Name of the LinkState (p. 618)
-----------------	------------------------	---------------------------------------

Returns

State (p. 1068) of the **Link** (p. 595).

Exceptions

<i>common::Exception</i> (p. 444)	When <code>_linkName</code> is invalid.
---	---

10.138.3.8 `unsigned int gazebo::physics::ModelState::GetLinkStateCount () const`

Get the number of link states.

This returns the number of Links recorded.

Returns

Number of **LinkState** (p. 618) recorded.

10.138.3.9 `LinkState_M gazebo::physics::ModelState::GetLinkStates (const boost::regex & _regex) const`

Get link states based on a regular expression.

Parameters

<code>in</code>	<code>_regex</code>	The regular expression.
-----------------	---------------------	-------------------------

Returns

List of link states whose names match the regular expression.

10.138.3.10 `const LinkState_M& gazebo::physics::ModelState::GetLinkStates () const`

Get the link states.

Returns

A map of link states.

10.138.3.11 `const math::Pose& gazebo::physics::ModelState::GetPose () const`

Get the stored model pose.

Returns

The **math::Pose** (p. 797) of the **Model** (p. 678).

10.138.3.12 `bool gazebo::physics::ModelState::HasJointState (const std::string & _jointName) const`

Return true if there is a joint with the specified name.

Parameters

<code>in</code>	<code>_jointName</code>	Name of the Jointtate.
-----------------	-------------------------	------------------------

Returns

True if the joint exists in the model.

10.138.3.13 `bool gazebo::physics::ModelState::HasLinkState (const std::string & _linkName) const`

Return true if there is a link with the specified name.

Parameters

<i>in</i>	<i>_linkName</i>	Name of the LinkState (p. 618).
-----------	------------------	--

Returns

True if the link exists in the model.

10.138.3.14 `bool gazebo::physics::ModelState::IsZero () const`

Return true if the values in the state are zero.

Returns

True if the values in the state are zero.

10.138.3.15 `void gazebo::physics::ModelState::Load (const ModelPtr _model, const common::Time & _realTime, const common::Time & _simTime)`

Load state from **Model** (p. 678) pointer.

Build a **ModelState** (p. 698) from an existing **Model** (p. 678).

Parameters

<i>in</i>	<i>_model</i>	Pointer to the model from which to gather state info.
<i>in</i>	<i>_realTime</i>	Real time stamp.
<i>in</i>	<i>_simTime</i>	Sim time stamp.

10.138.3.16 `virtual void gazebo::physics::ModelState::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Load **ModelState** (p. 698) information from stored data in and SDF::Element

Parameters

<i>in</i>	<i>_elem</i>	Pointer to the SDF::Element containing state info.
-----------	--------------	--

Reimplemented from **gazebo::physics::State** (p. 1070).

10.138.3.17 `ModelState gazebo::physics::ModelState::operator+ (const ModelState & _state) const`

Addition operator.

Parameters

in	_pt	A state to subtract.
----	-----	----------------------

Returns

The resulting state.

10.138.3.18 **ModelState** gazebo::physics::ModelState::operator- (const ModelState & _state) const

Subtraction operator.

Parameters

in	_pt	A state to subtract.
----	-----	----------------------

Returns

The resulting state.

10.138.3.19 **ModelState&** gazebo::physics::ModelState::operator= (const ModelState & _state)

Assignment operator.

Parameters

in	_state	State (p. 1068) value
----	--------	------------------------------

Returns

this

10.138.3.20 virtual void gazebo::physics::ModelState::SetRealTime (const common::Time & _time) [virtual]

Set the real time when this state was generated.

Parameters

in	_time	Clock time since simulation was stated.
----	-------	---

Reimplemented from **gazebo::physics::State** (p. 1071).

10.138.3.21 virtual void gazebo::physics::ModelState::SetSimTime (const common::Time & _time) [virtual]

Set the sim time when this state was generated.

Parameters

in	_time	Simulation time when the data was recorded.
----	-------	---

Reimplemented from **gazebo::physics::State** (p. 1071).

10.138.3.22 `virtual void gazebo::physics::ModelState::SetWallTime (const common::Time & _time) [virtual]`

Set the wall time when this state was generated.

Parameters

<code>in</code>	<code>_time</code>	The absolute clock time when the State (p. 1068) data was recorded.
-----------------	--------------------	--

Reimplemented from **gazebo::physics::State** (p. 1072).

10.138.4 Friends And Related Function Documentation

10.138.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::physics::ModelState & _state) [friend]`

Stream insertion operator.

Parameters

<code>in</code>	<code>_out</code>	output stream.
<code>in</code>	<code>_state</code>	Model (p. 678) state to output.

Returns

The stream.

The documentation for this class was generated from the following file:

- **ModelState.hh**

10.139 gazebo::common::MouseEvent Class Reference

Generic description of a mouse event.

```
#include <common/common.hh>
```

Public Types

- enum **Buttons** { **NO_BUTTON** = 0x0, **LEFT** = 0x1, **MIDDLE** = 0x2, **RIGHT** = 0x4 }
Standard mouse buttons enumeration.
- enum **EventType** {
NO_EVENT, **MOVE**, **PRESS**, **RELEASE**,
SCROLL }
Mouse event types enumeration.

Public Member Functions

- **MouseEvent** ()
Constructor.

Public Attributes

- bool **alt**
Alt key press flag.
- unsigned int **button**
The button which caused the event.
- unsigned int **buttons**
State of the buttons when the event was generated.
- bool **control**
Control key press flag.
- bool **dragging**
Flag for mouse drag motion.
- float **moveScale**
Scaling factor.
- **math::Vector2i** **pos**
Mouse pointer position on the screen.
- **math::Vector2i** **pressPos**
Position of button press.
- **math::Vector2i** **prevPos**
Previous position.
- **math::Vector2i** **scroll**
Scroll position.
- bool **shift**
Shift key press flag.
- **EventType** **type**
Event type.

10.139.1 Detailed Description

Generic description of a mouse event.

10.139.2 Member Enumeration Documentation

10.139.2.1 enum gazebo::common::MouseEvent::Buttons

Standard mouse buttons enumeration.

Enumerator:

NO_BUTTON
LEFT
MIDDLE
RIGHT

10.139.2.2 enum gazebo::common::MouseEvent::EventType

Mouse event types enumeration.

Enumerator:

NO_EVENT
MOVE
PRESS
RELEASE
SCROLL

10.139.3 Constructor & Destructor Documentation

10.139.3.1 gazebo::common::MouseEvent::MouseEvent() [inline]

Constructor.

10.139.4 Member Data Documentation

10.139.4.1 bool gazebo::common::MouseEvent::alt

Alt key press flag.

10.139.4.2 unsigned int gazebo::common::MouseEvent::button

The button which caused the event.

10.139.4.3 unsigned int gazebo::common::MouseEvent::buttons

State of the buttons when the event was generated.

10.139.4.4 bool gazebo::common::MouseEvent::control

Control key press flag.

10.139.4.5 bool gazebo::common::MouseEvent::dragging

Flag for mouse drag motion.

10.139.4.6 float gazebo::common::MouseEvent::moveScale

Scaling factor.

10.139.4.7 math::Vector2i gazebo::common::MouseEvent::pos

Mouse pointer position on the screen.

10.139.4.8 `math::Vector2i gazebo::common::MouseEvent::pressPos`

Position of button press.

10.139.4.9 `math::Vector2i gazebo::common::MouseEvent::prevPos`

Previous position.

10.139.4.10 `math::Vector2i gazebo::common::MouseEvent::scroll`

Scroll position.

10.139.4.11 `bool gazebo::common::MouseEvent::shift`

Shift key press flag.

10.139.4.12 `EventType gazebo::common::MouseEvent::type`

Event type.

The documentation for this class was generated from the following file:

- **MouseEvent.hh**

10.140 gazebo::rendering::MovableText Class Reference

Movable text.

```
#include <rendering/rendering.hh>
```

Public Types

- enum **HorizAlign** { **H_LEFT**, **H_CENTER** }
Horizontal alignment.
- enum **VertAlign** { **V_BELOW**, **V_ABOVE** }
vertical alignment

Public Member Functions

- **MovableText** ()
Constructor.
- virtual **~MovableText** ()
Destructor.
- **math::Box GetAABB** ()
Get the axis aligned bounding box of the text.
- float **GetBaseline** () const
Get the baseline height.

- float **GetCharHeight** () const
Set the height of a characters return Height of the characters.
- const **common::Color** & **GetColor** () const
Get the text color.
- const std::string & **GetFont** () const
Get the font.
- bool **GetShowOnTop** () const
True = text is displayed on top.
- float **GetSpaceWidth** () const
Get the width of a space.
- const std::string & **GetText** () const
Get the displayed text.
- void **Load** (const std::string & _name, const std::string & _text, const std::string & _fontName="Arial", float _charHeight=1.0, const **common::Color** & _color=**common::Color::White**)
Loads text and font info.
- void **SetBaseline** (float _height)
Set the baseline height of the text.
- void **SetCharHeight** (float _height)
Set the height of a character.
- void **SetColor** (const **common::Color** & _color)
Set the text color.
- void **SetFontName** (const std::string & _font)
Set the font.
- void **SetShowOnTop** (bool _show)
True = text always is displayed on top.
- void **SetSpaceWidth** (float _width)
Set the width of a space.
- void **SetText** (const std::string & _text)
Set the text to display.
- void **SetTextAlignment** (const **HorizAlign** & _hAlign, const **VertAlign** & _vAlign)
Set the alignment of the text.
- void **Update** ()
Update the text.
- virtual void **visitRenderables** (Ogre::Renderable::Visitor * _visitor, bool _debug=false)

Protected Member Functions

- void **_setupGeometry** ()
- void **_updateColors** ()
- float **getBoundingRadius** () const
- const Ogre::LightList & **getLights** (void) const
- const Ogre::MaterialPtr & **getMaterial** (void) const
- void **getRenderOperation** (Ogre::RenderOperation &op)
- float **getSquaredViewDepth** (const Ogre::Camera *cam) const
- void **getWorldTransforms** (Ogre::Matrix4 *xform) const

10.140.1 Detailed Description

Movable text.

10.140.2 Member Enumeration Documentation

10.140.2.1 enum gazebo::rendering::MovableText::HorizAlign

Horizontal alignment.

Enumerator:

H_LEFT Left alignment.

H_CENTER Center alignment.

10.140.2.2 enum gazebo::rendering::MovableText::VertAlign

vertical alignment

Enumerator:

V_BELOW Align below.

V_ABOVE Align above.

10.140.3 Constructor & Destructor Documentation

10.140.3.1 gazebo::rendering::MovableText::MovableText ()

Constructor.

10.140.3.2 virtual gazebo::rendering::MovableText::~MovableText () [virtual]

Destructor.

10.140.4 Member Function Documentation

10.140.4.1 void gazebo::rendering::MovableText::_setupGeometry () [protected]

10.140.4.2 void gazebo::rendering::MovableText::_updateColors () [protected]

10.140.4.3 math::Box gazebo::rendering::MovableText::GetAABB ()

Get the axis aligned bounding box of the text.

Returns

The axis aligned bounding box.

10.140.4.4 `float gazebo::rendering::MovableText::GetBaseline () const`

Get the baseline height.

Returns

Baseline height

10.140.4.5 `float gazebo::rendering::MovableText::getBoundingRadius () const` [protected]

10.140.4.6 `float gazebo::rendering::MovableText::GetCharHeight () const`

Set the height of a characters return Height of the characters.

10.140.4.7 `const common::Color& gazebo::rendering::MovableText::GetColor () const`

Get the text color.

Returns

Texture color.

10.140.4.8 `const std::string& gazebo::rendering::MovableText::GetFont () const`

Get the font.

Returns

The font name

10.140.4.9 `const Ogre::LightList& gazebo::rendering::MovableText::getLights (void) const` [protected]

10.140.4.10 `const Ogre::MaterialPtr& gazebo::rendering::MovableText::getMaterial (void) const` [protected]

10.140.4.11 `void gazebo::rendering::MovableText::getRenderOperation (Ogre::RenderOperation & op)` [protected]

10.140.4.12 `bool gazebo::rendering::MovableText::GetShowOnTop () const`

True = text is displayed on top.

Returns

True if `MovableText::SetShownOnTop(true)` was called.

10.140.4.13 `float gazebo::rendering::MovableText::GetSpaceWidth () const`

Get the width of a space.

Returns

Space width

10.140.4.14 `float gazebo::rendering::MovableText::getSquaredViewDepth (const Ogre::Camera * cam) const` [protected]

10.140.4.15 `const std::string& gazebo::rendering::MovableText::GetText () const`

Get the displayed text.

Returns

The displayed text.

10.140.4.16 `void gazebo::rendering::MovableText::getWorldTransforms (Ogre::Matrix4 * xform) const` [protected]

10.140.4.17 `void gazebo::rendering::MovableText::Load (const std::string & _name, const std::string & _text, const std::string & _fontName = "Arial", float _charHeight = 1.0, const common::Color & _color = common::Color::White)`

Loads text and font info.

Parameters

in	<i>_name</i>	Name of the text object
in	<i>_text</i>	Text to render
in	<i>_fontName</i>	Font to use
in	<i>_charHeight</i>	Height of the characters
in	<i>_color</i>	Text color

10.140.4.18 `void gazebo::rendering::MovableText::SetBaseline (float _height)`

Set the baseline height of the text.

Parameters

in	<i>_height</i>	Baseline height
----	----------------	-----------------

10.140.4.19 `void gazebo::rendering::MovableText::SetCharHeight (float _height)`

Set the height of a character.

Parameters

in	<i>_height</i>	Height of the characters.
----	----------------	---------------------------

10.140.4.20 `void gazebo::rendering::MovableText::SetColor (const common::Color & _color)`

Set the text color.

Parameters

in	<i>_color</i>	Text color.
----	---------------	-------------

10.140.4.21 `void gazebo::rendering::MovableText::SetFontName (const std::string & _font)`

Set the font.

Parameters

<code>in</code>	<code><i>_font</i></code>	Name of the font
-----------------	---------------------------	------------------

10.140.4.22 `void gazebo::rendering::MovableText::SetShowOnTop (bool _show)`

True = text always is displayed on top.

Parameters

<code>in</code>	<code><i>_show</i></code>	Set to true to render the text on top of all other drawables.
-----------------	---------------------------	---

10.140.4.23 `void gazebo::rendering::MovableText::SetSpaceWidth (float _width)`

Set the width of a space.

Parameters

<code>in</code>	<code><i>_width</i></code>	space width
-----------------	----------------------------	-------------

10.140.4.24 `void gazebo::rendering::MovableText::SetText (const std::string & _text)`

Set the text to display.

Parameters

<code>in</code>	<code><i>_text</i></code>	The text to display.
-----------------	---------------------------	----------------------

10.140.4.25 `void gazebo::rendering::MovableText::SetTextAlignment (const HorizAlign & _hAlign, const VertAlign & _vAlign)`

Set the alignment of the text.

Parameters

<code>in</code>	<code><i>_hAlign</i></code>	Horizontal alignment
<code>in</code>	<code><i>_vAlign</i></code>	Vertical alignment

10.140.4.26 `void gazebo::rendering::MovableText::Update ()`

Update the text.

```
10.140.4.27 virtual void gazebo::rendering::MovableText::visitRenderables ( Ogre::Renderable::Visitor * _visitor, bool _debug =
false ) [virtual]
```

The documentation for this class was generated from the following file:

- **MovableText.hh**

10.141 gazebo::common::MovingWindowFilter< T > Class Template Reference

Base class for **MovingWindowFilter** (p. 715).

```
#include <common/common.hh>
```

Public Member Functions

- **MovingWindowFilter** ()
Constructor.
- virtual **~MovingWindowFilter** ()
Destructor.
- **T Get** ()
Get filtered result.
- bool **GetWindowFilled** () const
Get whether the window has been filled.
- unsigned int **GetWindowSize** () const
Get the window size.
- void **SetWindowSize** (unsigned int _n)
Set window size.
- void **Update** (T _val)
Update value of filter.

Protected Member Functions

- **MovingWindowFilter (MovingWindowFilterPrivate< T > &_d)**
Allow subclasses to initialize their own data pointer.

Protected Attributes

- **MovingWindowFilterPrivate< T > * dataPtr**
Data pointer.

10.141.1 Detailed Description

```
template<typename T>class gazebo::common::MovingWindowFilter< T >
```

Base class for **MovingWindowFilter** (p. 715).

10.141.2 Constructor & Destructor Documentation

10.141.2.1 `template<typename T > gazebo::common::MovingWindowFilter< T >::MovingWindowFilter (MovingWindowFilterPrivate< T > & _d)` [protected]

Allow subclasses to initialize their own data pointer.

Parameters

in	_d	Reference to data pointer.
----	----	----------------------------

10.141.3 Member Data Documentation

10.141.3.1 `template<typename T > MovingWindowFilterPrivate<T>* gazebo::common::MovingWindowFilter< T >::dataPtr` [protected]

Data pointer.

The documentation for this class was generated from the following file:

- **MovingWindowFilter.hh**

10.142 gazebo::common::MovingWindowFilterPrivate< T > Class Template Reference

```
#include <MovingWindowFilter.hh>
```

Public Member Functions

- **MovingWindowFilterPrivate ()**

Public Attributes

- unsigned int **samples**
keep track of number of elements
- T **sum**
keep track of running sum
- std::vector< T > **valHistory**
buffer history of raw values
- std::vector< T >::iterator **valIter**
iterator pointing to current value in buffer
- unsigned int **valWindowSize**
For moving window smoothed value.

10.142.1 Member Data Documentation

10.142.1.1 `template<typename T > unsigned int gazebo::common::MovingWindowFilterPrivate< T >::samples`

keep track of number of elements

10.142.1.2 `template<typename T > T gazebo::common::MovingWindowFilterPrivate< T >::sum`

keep track of running sum

10.142.1.3 `template<typename T > std::vector<T> gazebo::common::MovingWindowFilterPrivate< T >::valHistory`

buffer history of raw values

10.142.1.4 `template<typename T > std::vector<T>::iterator gazebo::common::MovingWindowFilterPrivate< T >::vallter`

iterator pointing to current value in buffer

10.142.1.5 `template<typename T > unsigned int gazebo::common::MovingWindowFilterPrivate< T >::valWindowSize`

For moving window smoothed value.

The documentation for this class was generated from the following file:

- **MovingWindowFilter.hh**

10.143 gazebo::msgs::MsgFactory Class Reference

A factory that generates protobuf message based on a string type.

```
#include <msgs/msgs.hh>
```

Static Public Member Functions

- static void **GetMsgTypes** (std::vector< std::string > &_types)
Get all the message types.
- static boost::shared_ptr
< google::protobuf::Message > **NewMsg** (const std::string &_msgType)
Create a new instance of a message.
- static void **RegisterMsg** (const std::string &_msgType, **MsgFactoryFn** _factoryfn)
Register a message.

10.143.1 Detailed Description

A factory that generates protobuf message based on a string type.

10.143.2 Member Function Documentation

10.143.2.1 `static void gazebo::msgs::MsgFactory::GetMsgTypes (std::vector< std::string > &_types)` [static]

Get all the message types.

Parameters

out	<i>_types</i>	Vector of strings of the message types.
-----	---------------	---

10.143.2.2 `static boost::shared_ptr<google::protobuf::Message> gazebo::msgs::MsgFactory::NewMsg (const std::string & _msgType) [static]`

Create a new instance of a message.

Parameters

in	<i>_msgType</i>	Type of message to create.
----	-----------------	----------------------------

Returns

Pointer to a google protobuf message. Null if the message type could not be handled.

10.143.2.3 `static void gazebo::msgs::MsgFactory::RegisterMsg (const std::string & _msgType, MsgFactoryFn _factoryfn) [static]`

Register a message.

Parameters

in	<i>_msgType</i>	Type of message to register.
in	<i>_factoryfn</i>	Function that generates the message.

The documentation for this class was generated from the following file:

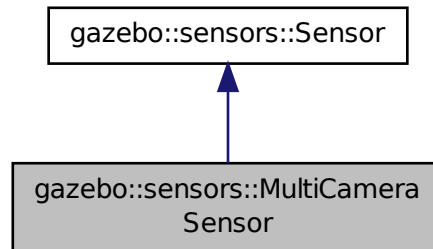
- [MsgFactory.hh](#)

10.144 gazebo::sensors::MultiCameraSensor Class Reference

Multiple camera sensor.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::MultiCameraSensor:



Public Member Functions

- **MultiCameraSensor** ()
Constructor.
- virtual **~MultiCameraSensor** ()
Destructor.
- **rendering::CameraPtr GetCamera** (unsigned int _index) const
*Returns a pointer to a **rendering::Camera** (p. 197).*
- unsigned int **GetCameraCount** () const
Get the number of cameras.
- const unsigned char * **GetImageData** (unsigned int _index)
Gets the raw image data from the sensor.
- unsigned int **GetImageHeight** (unsigned int _index) const
Gets the height of the image in pixels.
- unsigned int **GetImageWidth** (unsigned int _index) const
Gets the width of the image in pixels.
- virtual std::string **GetTopic** () const
Returns the topic name as set in SDF.
- virtual void **Init** ()
Initialize the sensor.
- virtual bool **IsActive** ()
Returns true if sensor generation is active.
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.
- bool **SaveFrame** (const std::vector< std::string > &_filenames)
Saves the camera image(s) to the disk.

Protected Member Functions

- virtual void **Fini** ()
Finalize the sensor.
- virtual bool **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.144.1 Detailed Description

Multiple camera sensor.

This sensor type can create one or more synchronized cameras.

10.144.2 Constructor & Destructor Documentation

10.144.2.1 gazebo::sensors::MultiCameraSensor::MultiCameraSensor ()

Constructor.

10.144.2.2 virtual gazebo::sensors::MultiCameraSensor::~~MultiCameraSensor () [virtual]

Destructor.

10.144.3 Member Function Documentation

10.144.3.1 virtual void gazebo::sensors::MultiCameraSensor::Fini () [protected], [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 912).

10.144.3.2 rendering::CameraPtr gazebo::sensors::MultiCameraSensor::GetCamera (unsigned int _index) const

Returns a pointer to a **rendering::Camera** (p. 197).

Parameters

<code>in</code>	<code>_index</code>	Index of the camera to get
-----------------	---------------------	----------------------------

Returns

The Pointer to the camera sensor.

See Also

MultiCameraSensor::GetCameraCount (p. 721)

10.144.3.3 unsigned int gazebo::sensors::MultiCameraSensor::GetCameraCount () const

Get the number of cameras.

Returns

The number of cameras.

10.144.3.4 const unsigned char* gazebo::sensors::MultiCameraSensor::GetImageData (unsigned int *_index*)

Gets the raw image data from the sensor.

Parameters

in	<i>_index</i>	Index of the camera
----	---------------	---------------------

Returns

The pointer to the image data array.

See Also

MultiCameraSensor::GetCameraCount (p. 721)

10.144.3.5 unsigned int gazebo::sensors::MultiCameraSensor::GetImageHeight (unsigned int *_index*) const

Gets the height of the image in pixels.

Parameters

in	<i>_index</i>	Index of the camera
----	---------------	---------------------

Returns

The image height in pixels.

See Also

MultiCameraSensor::GetCameraCount (p. 721)

10.144.3.6 unsigned int gazebo::sensors::MultiCameraSensor::GetImageWidth (unsigned int *_index*) const

Gets the width of the image in pixels.

Parameters

in	<i>_index</i>	Index of the camera
----	---------------	---------------------

Returns

The image width in pixels.

See Also

MultiCameraSensor::GetCameraCount (p. 721)

10.144.3.7 `virtual std::string gazebo::sensors::MultiCameraSensor::GetTopic () const` [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 914).

10.144.3.8 `virtual void gazebo::sensors::MultiCameraSensor::Init ()` [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.144.3.9 `virtual bool gazebo::sensors::MultiCameraSensor::IsActive ()` [virtual]

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.144.3.10 `virtual void gazebo::sensors::MultiCameraSensor::Load (const std::string & _worldName)` [virtual]

Load the sensor with default parameters.

Parameters

<code>in</code>	<code>_worldName</code>	Name of world to load from.
-----------------	-------------------------	-----------------------------

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.144.3.11 `bool gazebo::sensors::MultiCameraSensor::SaveFrame (const std::vector< std::string > & _filenames)`

Saves the camera image(s) to the disk.

Parameters

in	<code>_filenames</code>	The name of the files for each camera.
----	-------------------------	--

Returns

True if successful, false if unsuccessful.

See Also

MultiCameraSensor::GetCameraCount (p. 721)

10.144.3.12 `virtual bool gazebo::sensors::MultiCameraSensor::UpdateImpl (bool)` [protected],[virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	<code>_force</code>	True if update is forced, false if not
----	---------------------	--

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 917).

The documentation for this class was generated from the following file:

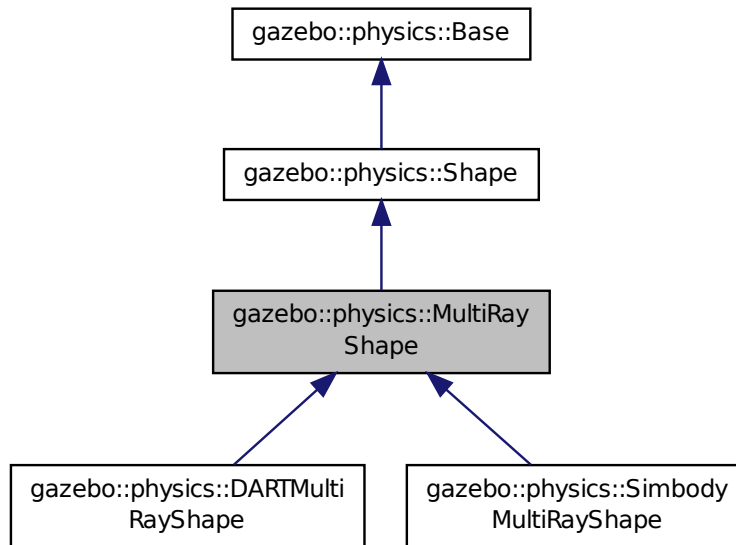
- **MultiCameraSensor.hh**

10.145 gazebo::physics::MultiRayShape Class Reference

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::MultiRayShape:



Public Member Functions

- **MultiRayShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~MultiRayShape** ()
Destructor.
- template<typename T >
event::ConnectionPtr ConnectNewLaserScans (T _subscriber)
Connect a to the new laser scan signal.
- void **DisconnectNewLaserScans** (**event::ConnectionPtr** &_conn)
Disconnect from the new laser scans signal.
- void **FillMsg** (**msgs::Geometry** &_msg)
This function is not implemented.
- int **GetFiducial** (unsigned int _index)
Get detected fiducial value for a ray.
- **math::Angle GetMaxAngle** () const
Get the maximum angle.
- double **GetMaxRange** () const
Get the maximum range.
- **math::Angle GetMinAngle** () const
Get the minimum angle.
- double **GetMinRange** () const
Get the minimum range.

- double **GetRange** (unsigned int _index)
Get detected range for a ray.
- double **GetResRange** () const
Get the range resolution.
- double **GetRetro** (unsigned int _index)
Get detected retro (intensity) value for a ray.
- int **GetSampleCount** () const
Get the horizontal sample count.
- double **GetScanResolution** () const
Get the horizontal resolution.
- **math::Angle GetVerticalMaxAngle** () const
Get the vertical max angle.
- **math::Angle GetVerticalMinAngle** () const
Get the vertical min angle.
- int **GetVerticalSampleCount** () const
Get the vertical sample count.
- double **GetVerticalScanResolution** () const
Get the vertical range resolution.
- virtual void **Init** ()
Init the shape.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
This function is not implemented.
- virtual void **SetScale** (const **math::Vector3** &_scale)
Set the scale of the multi ray shape.
- void **Update** ()
Update the ray collisions.

Protected Member Functions

- virtual void **AddRay** (const **math::Vector3** &_start, const **math::Vector3** &_end)
Add a ray to the collision.
- virtual void **UpdateRays** ()=0
Physics engine specific method for updating the rays.

Protected Attributes

- sdf::ElementPtr **horzElem**
Horizontal SDF element pointer.
- **event::EventT**< void()> **newLaserScans**
New laser scans event.
- **math::Pose** **offset**
Pose offset of all the rays.
- sdf::ElementPtr **rangeElem**
Range SDF element pointer.
- sdf::ElementPtr **rayElem**
Ray SDF element pointer.

- `std::vector< RayShapePtr > rays`
Ray data.
- `sdf::ElementPtr scanElem`
Scan SDF element pointer.
- `sdf::ElementPtr vertElem`
Vertical SDF element pointer.

Additional Inherited Members

10.145.1 Detailed Description

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

10.145.2 Constructor & Destructor Documentation

10.145.2.1 `gazebo::physics::MultiRayShape::MultiRayShape (CollisionPtr _parent) [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Parent collision shape.
----	----------------------	-------------------------

10.145.2.2 `virtual gazebo::physics::MultiRayShape::~MultiRayShape () [virtual]`

Destructor.

10.145.3 Member Function Documentation

10.145.3.1 `virtual void gazebo::physics::MultiRayShape::AddRay (const math::Vector3 & _start, const math::Vector3 & _end) [protected], [virtual]`

Add a ray to the collision.

Parameters

in	<code>_start</code>	Start of the ray.
in	<code>_end</code>	End of the ray.

Reimplemented in `gazebo::physics::DARTMultiRayShape` (p. 80), and `gazebo::physics::SimbodyMultiRayShape` (p. 986).

10.145.3.2 `template<typename T > event::ConnectionPtr gazebo::physics::MultiRayShape::ConnectNewLaserScans (T _subscriber) [inline]`

Connect a to the new laser scan signal.

Parameters

in	<code>_subscriber</code>	Callback function.
----	--------------------------	--------------------

Returns

The connection, which must be kept in scope.

10.145.3.3 `void gazebo::physics::MultiRayShape::DisconnectNewLaserScans (event::ConnectionPtr & _conn) [inline]`

Disconnect from the new laser scans signal.

Parameters

in	<code>_conn</code>	Connection to remove.
----	--------------------	-----------------------

10.145.3.4 `void gazebo::physics::MultiRayShape::FillMsg (msgs::Geometry & _msg) [virtual]`

This function is not implemented.

Fill a message with this shape's values.

Parameters

out	<code>_msg</code>	Message that contains the shape's values.
-----	-------------------	---

Implements `gazebo::physics::Shape` (p. 934).

10.145.3.5 `int gazebo::physics::MultiRayShape::GetFiducial (unsigned int _index)`

Get detected fiducial value for a ray.

Parameters

in	<code>_index</code>	Index of the ray.
----	---------------------	-------------------

Returns

Fiducial value for the ray.

10.145.3.6 `math::Angle gazebo::physics::MultiRayShape::GetMaxAngle () const`

Get the maximum angle.

Returns

Maximum angle of ray scan.

10.145.3.7 `double gazebo::physics::MultiRayShape::GetMaxRange () const`

Get the maximum range.

Returns

Maximum range of all the rays.

10.145.3.8 `math::Angle gazebo::physics::MultiRayShape::GetMinAngle () const`

Get the minimum angle.

Returns

Minimum angle of ray scan.

10.145.3.9 `double gazebo::physics::MultiRayShape::GetMinRange () const`

Get the minimum range.

Returns

Minimum range of all the rays.

10.145.3.10 `double gazebo::physics::MultiRayShape::GetRange (unsigned int _index)`

Get detected range for a ray.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the ray.
-----------------	----------------------------	-------------------

Returns

Returns DBL_MAX for no detection.

10.145.3.11 `double gazebo::physics::MultiRayShape::GetResRange () const`

Get the range resolution.

Returns

Range resolution of all the rays.

10.145.3.12 `double gazebo::physics::MultiRayShape::GetRetro (unsigned int _index)`

Get detected retro (intensity) value for a ray.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the ray.
-----------------	----------------------------	-------------------

Returns

Retro value for the ray.

10.145.3.13 `int gazebo::physics::MultiRayShape::GetSampleCount () const`

Get the horizontal sample count.

Returns

Horizontal sample count.

10.145.3.14 `double gazebo::physics::MultiRayShape::GetScanResolution () const`

Get the horizontal resolution.

Returns

Horizontal resolution.

10.145.3.15 `math::Angle gazebo::physics::MultiRayShape::GetVerticalMaxAngle () const`

Get the vertical max angle.

Returns

Vertical max angle.

10.145.3.16 `math::Angle gazebo::physics::MultiRayShape::GetVerticalMinAngle () const`

Get the vertical min angle.

Returns

Vertical min angle.

10.145.3.17 `int gazebo::physics::MultiRayShape::GetVerticalSampleCount () const`

Get the vertical sample count.

Returns

Verical sample count.

10.145.3.18 `double gazebo::physics::MultiRayShape::GetVerticalScanResolution () const`

Get the vertical range resolution.

Returns

Vertical range resolution.

10.145.3.19 `virtual void gazebo::physics::MultiRayShape::Init () [virtual]`

Init the shape.

Implements **`gazebo::physics::Shape`** (p. 935).

10.145.3.20 `virtual void gazebo::physics::MultiRayShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]`

This function is not implemented.

Update the ray based on a message.

Parameters

<code>in</code>	<code>_msg</code>	Message to update from.
-----------------	-------------------	-------------------------

Implements **`gazebo::physics::Shape`** (p. 935).

10.145.3.21 `virtual void gazebo::physics::MultiRayShape::SetScale (const math::Vector3 & _scale) [virtual]`

Set the scale of the multi ray shape.

Returns

`_scale` Scale to set the multi ray shape to.

Implements **`gazebo::physics::Shape`** (p. 935).

10.145.3.22 `void gazebo::physics::MultiRayShape::Update () [virtual]`

Update the ray collisions.

Reimplemented from **`gazebo::physics::Base`** (p. 180).

10.145.3.23 `virtual void gazebo::physics::MultiRayShape::UpdateRays () [protected],[pure virtual]`

Physics engine specific method for updating the rays.

Implemented in **`gazebo::physics::DARTMultiRayShape`** (p. 80), and **`gazebo::physics::SimbodyMultiRayShape`** (p. 986).

10.145.4 Member Data Documentation

10.145.4.1 `sdf::ElementPtr gazebo::physics::MultiRayShape::horzElem [protected]`

Horizontal SDF element pointer.

10.145.4.2 `event::EventT<void()> gazebo::physics::MultiRayShape::newLaserScans [protected]`

New laser scans event.

10.145.4.3 `math::Pose` `gazebo::physics::MultiRayShape::offset` `[protected]`

Pose offset of all the rays.

10.145.4.4 `sdf::ElementPtr` `gazebo::physics::MultiRayShape::rangeElem` `[protected]`

Range SDF element pointer.

10.145.4.5 `sdf::ElementPtr` `gazebo::physics::MultiRayShape::rayElem` `[protected]`

Ray SDF element pointer.

10.145.4.6 `std::vector<RayShapePtr>` `gazebo::physics::MultiRayShape::rays` `[protected]`

Ray data.

10.145.4.7 `sdf::ElementPtr` `gazebo::physics::MultiRayShape::scanElem` `[protected]`

Scan SDF element pointer.

10.145.4.8 `sdf::ElementPtr` `gazebo::physics::MultiRayShape::vertElem` `[protected]`

Vertical SDF element pointer.

The documentation for this class was generated from the following file:

- **MultiRayShape.hh**

10.146 gazebo::transport::Node Class Reference

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

```
#include <transport/transport.hh>
```

Public Member Functions

- **Node** ()
Constructor.
- virtual `~Node` ()
Destructor.
- `template<typename M >`
transport::PublisherPtr Advertise (const std::string &_topic, unsigned int _queueLimit=1000, double _hzRate=0)
Advertise a topic.
- std::string **DecodeTopicName** (const std::string &_topic)
Decode a topic name.
- std::string **EncodeTopicName** (const std::string &_topic)

- Encode a topic name.*

 - void **Fini** ()

Finalize the node.
- unsigned int **GetId** () const
- Get the unique ID of the node.*

 - std::string **GetMsgType** (const std::string &_topic) const

Get the message type for a topic.
- std::string **GetTopicNamespace** () const
- Get the topic namespace for this node.*

 - bool **HandleData** (const std::string &_topic, const std::string &_msg)

Handle incoming data.
- bool **HandleMessage** (const std::string &_topic, **MessagePtr** _msg)
- Handle incoming msg.*

 - bool **HasLatchedSubscriber** (const std::string &_topic) const

Return true if a subscriber on a specific topic is latched.
- void **Init** (const std::string &_space="")
- Init the node.*

 - void **InsertLatchedMsg** (const std::string &_topic, const std::string &_msg)

Add a latched message to the node for publication.
- void **InsertLatchedMsg** (const std::string &_topic, **MessagePtr** _msg)
- Add a latched message to the node for publication.*

 - void **ProcessIncoming** ()

Process incoming messages.
- void **ProcessPublishers** ()
- Process all publishers, which has each publisher send it's most recent message over the wire.*

 - template<typename M >
 - void **Publish** (const std::string &_topic, const google::protobuf::Message &_message)
 - A convenience function for a one-time publication of a message.*
- void **RemoveCallback** (const std::string &_topic, unsigned int _id)
- template<typename M , typename T >
- SubscriberPtr Subscribe** (const std::string &_topic, void(T::*_fp)(const boost::shared_ptr< M const > &), T *_obj, bool _latching=false)
- Subscribe to a topic using a class method as the callback.*

 - template<typename M >
 - SubscriberPtr Subscribe** (const std::string &_topic, void(*_fp)(const boost::shared_ptr< M const > &), bool _latching=false)
 - Subscribe to a topic using a bare function as the callback.*
- template<typename T >
- SubscriberPtr Subscribe** (const std::string &_topic, void(T::*_fp)(const std::string &), T *_obj, bool _latching=false)
- Subscribe to a topic using a class method as the callback.*

 - **SubscriberPtr Subscribe** (const std::string &_topic, void(*_fp)(const std::string &), bool _latching=false)
 - Subscribe to a topic using a bare function as the callback.*

10.146.1 Detailed Description

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

10.146.2 Constructor & Destructor Documentation

10.146.2.1 gazebo::transport::Node::Node ()

Constructor.

10.146.2.2 virtual gazebo::transport::Node::~~Node () [virtual]

Destructor.

10.146.3 Member Function Documentation

10.146.3.1 template<typename M > transport::PublisherPtr gazebo::transport::Node::Advertise (const std::string & *_topic*, unsigned int *_queueLimit* = 1000, double *_hzRate* = 0) [inline]

Advertise a topic.

Parameters

in	<i>_topic</i>	The topic to advertise
in	<i>_queueLimit</i>	The maximum number of outgoing messages to queue for delivery
in	<i>_hz</i>	Update rate for the publisher. Units are 1.0/seconds.

Returns

Pointer to new publisher object

References SingletonT< T >::Instance().

10.146.3.2 std::string gazebo::transport::Node::DecodeTopicName (const std::string & *_topic*)

Decode a topic name.

Parameters

in	<i>The</i>	encoded name
----	------------	--------------

Returns

The decoded name

10.146.3.3 std::string gazebo::transport::Node::EncodeTopicName (const std::string & *_topic*)

Encode a topic name.

Parameters

in	<i>The</i>	decoded name
----	------------	--------------

Returns

The encoded name

10.146.3.4 `void gazebo::transport::Node::Fini ()`

Finalize the node.

10.146.3.5 `unsigned int gazebo::transport::Node::GetId () const`

Get the unique ID of the node.

Returns

The unique ID of the node

10.146.3.6 `std::string gazebo::transport::Node::GetMsgType (const std::string & _topic) const`

Get the message type for a topic.

Parameters

<code>in</code>	<code>_topic</code>	The topic
-----------------	---------------------	-----------

Returns

The message type

10.146.3.7 `std::string gazebo::transport::Node::GetTopicNamespace () const`

Get the topic namespace for this node.

Returns

The namespace

10.146.3.8 `bool gazebo::transport::Node::HandleData (const std::string & _topic, const std::string & _msg)`

Handle incoming data.

Parameters

<code>in</code>	<code>_topic</code>	Topic for which the data was received
<code>in</code>	<code>_msg</code>	The message that was received

Returns

true if the message was handled successfully, false otherwise

10.146.3.9 `bool gazebo::transport::Node::HandleMessage (const std::string & _topic, MessagePtr _msg)`

Handle incoming msg.

Parameters

<code>in</code>	<code><i>_topic</i></code>	Topic for which the data was received
<code>in</code>	<code><i>_msg</i></code>	The message that was received

Returns

true if the message was handled successfully, false otherwise

10.146.3.10 `bool gazebo::transport::Node::HasLatchedSubscriber (const std::string & _topic) const`

Return true if a subscriber on a specific topic is latched.

Parameters

<code>in</code>	<code><i>_topic</i></code>	Name of the topic to check.
-----------------	----------------------------	-----------------------------

Returns

True if a latched subscriber exists.

10.146.3.11 `void gazebo::transport::Node::Init (const std::string & _space = " ")`

Init the node.

Parameters

<code>in</code>	<code><i>_space</i></code>	Set the global namespace of all topics. If left blank, the topic will initialize to the first namespace on the Master (p. 637)
-----------------	----------------------------	---

10.146.3.12 `void gazebo::transport::Node::InsertLatchedMsg (const std::string & _topic, const std::string & _msg)`

Add a latched message to the node for publication.

This is called when a subscription is connected to a publication.

Parameters

<code>in</code>	<code><i>_topic</i></code>	Name of the topic to publish data on.
<code>in</code>	<code><i>_msg</i></code>	The message to publish.

10.146.3.13 `void gazebo::transport::Node::InsertLatchedMsg (const std::string & _topic, MessagePtr _msg)`

Add a latched message to the node for publication.

This is called when a subscription is connected to a publication.

Parameters

in	<code>_topic</code>	Name of the topic to publish data on.
in	<code>_msg</code>	The message to publish.

10.146.3.14 `void gazebo::transport::Node::ProcessIncoming ()`

Process incoming messages.

10.146.3.15 `void gazebo::transport::Node::ProcessPublishers ()`

Process all publishers, which has each publisher send it's most recent message over the wire.

This is for internal use only

10.146.3.16 `template<typename M > void gazebo::transport::Node::Publish (const std::string & _topic, const google::protobuf::Message & _message) [inline]`

A convenience function for a one-time publication of a message.

This is inefficient, compared to **Node::Advertise** (p. 733) followed by **Publisher::Publish** (p. 822). This function should only be used when sending a message very infrequently.

Parameters

in	<code>_topic</code>	The topic to advertise
in	<code>_message</code>	Message to be published

10.146.3.17 `void gazebo::transport::Node::RemoveCallback (const std::string & _topic, unsigned int _id)`10.146.3.18 `template<typename M , typename T > SubscriberPtr gazebo::transport::Node::Subscribe (const std::string & _topic, void(T::*)(const boost::shared_ptr< M const > &) _fp, T * _obj, bool _latching = false) [inline]`

Subscribe to a topic using a class method as the callback.

Parameters

in	<code>_topic</code>	The topic to subscribe to
in	<code>_fp</code>	Class method to be called on receipt of new message
in	<code>_obj</code>	Class instance to be used on receipt of new message
in	<code>_latching</code>	If true, latch latest incoming message; otherwise don't latch

Returns

Pointer to new **Subscriber** (p. 1086) object

References SingletonT< T >::Instance().

10.146.3.19 `template<typename M > SubscriberPtr gazebo::transport::Node::Subscribe (const std::string & _topic, void(*)(const boost::shared_ptr< M const > &) _fp, bool _latching = false) [inline]`

Subscribe to a topic using a bare function as the callback.

Parameters

in	<code>_topic</code>	The topic to subscribe to
in	<code>_fp</code>	Function to be called on receipt of new message
in	<code>_latching</code>	If true, latch latest incoming message; otherwise don't latch

Returns

Pointer to new **Subscriber** (p. 1086) object

References SingletonT< T >::Instance().

```
10.146.3.20 template<typename T > SubscriberPtr gazebo::transport::Node::Subscribe ( const std::string & _topic,
void(T::*)(const std::string &) _fp, T * _obj, bool _latching = false ) [inline]
```

Subscribe to a topic using a class method as the callback.

Parameters

in	<code>_topic</code>	The topic to subscribe to
in	<code>_fp</code>	Class method to be called on receipt of new message
in	<code>_obj</code>	Class instance to be used on receipt of new message
in	<code>_latching</code>	If true, latch latest incoming message; otherwise don't latch

Returns

Pointer to new **Subscriber** (p. 1086) object

References gazebo::transport::SubscribeOptions::Init(), and SingletonT< T >::Instance().

```
10.146.3.21 SubscriberPtr gazebo::transport::Node::Subscribe ( const std::string & _topic, void(*)(const std::string &) _fp, bool
_latching = false ) [inline]
```

Subscribe to a topic using a bare function as the callback.

Parameters

in	<code>_topic</code>	The topic to subscribe to
in	<code>_fp</code>	Function to be called on receipt of new message
in	<code>_latching</code>	If true, latch latest incoming message; otherwise don't latch

Returns

Pointer to new **Subscriber** (p. 1086) object

References gazebo::transport::SubscribeOptions::Init(), and SingletonT< T >::Instance().

The documentation for this class was generated from the following file:

- **Node.hh**

10.147 gazebo::common::NodeAnimation Class Reference

Node animation.

```
#include <common/common.hh>
```

Public Member Functions

- **NodeAnimation** (const std::string &_name)
constructor
- **~NodeAnimation** ()
Destructor. It empties the key frames list.
- void **AddKeyFrame** (const double _time, const **math::Matrix4** &_trans)
Adds a key frame at a specific time.
- void **AddKeyFrame** (const double _time, const **math::Pose** &_pose)
Adds a key fram at a specific time.
- **math::Matrix4** **GetFrameAt** (double _time, bool _loop=true) const
Returns a frame transformation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.
- unsigned int **GetFrameCount** () const
Returns the number of key frames.
- void **GetKeyFrame** (const unsigned int _i, double &_time, **math::Matrix4** &_trans) const
Finds a key frame using the index.
- std::pair< double, **math::Matrix4** > **GetKeyFrame** (const unsigned int _i) const
Returns a key frame using the index.
- double **GetLength** () const
Returns the duration of the animations.
- std::string **GetName** () const
Returns the name.
- double **GetTimeAtX** (const double _x) const
Returns the time where a transformation's translational value along the X axis is equal to _x.
- void **Scale** (const double _scale)
Scales each transformation in the key frames.
- void **SetName** (const std::string &_name)
Changes the name of the animation.

Protected Attributes

- std::map< double, **math::Matrix4** > **keyFrames**
the dictionary of key frames, indexed by time
- double **length**
the duration of the animations (time of last key frame)
- std::string **name**
the name of the animation

10.147.1 Detailed Description

Node animation.

10.147.2 Constructor & Destructor Documentation

10.147.2.1 gazebo::common::NodeAnimation::NodeAnimation (const std::string & *_name*)

constructor

Parameters

in	<i>_name</i>	the name of the node
----	--------------	----------------------

10.147.2.2 gazebo::common::NodeAnimation::~~NodeAnimation ()

Destructor. It empties the key frames list.

10.147.3 Member Function Documentation

10.147.3.1 void gazebo::common::NodeAnimation::AddKeyFrame (const double *_time*, const math::Matrix4 & *_trans*)

Adds a key frame at a specific time.

Parameters

in	<i>_time</i>	the time of the key frame
in	<i>_trans</i>	the transformation

10.147.3.2 void gazebo::common::NodeAnimation::AddKeyFrame (const double *_time*, const math::Pose & *_pose*)

Adds a key fram at a specific time.

Parameters

in	<i>_time</i>	the tiem of the key frame
in	<i>_pose</i>	the pose

10.147.3.3 math::Matrix4 gazebo::common::NodeAnimation::GetFrameAt (double *_time*, bool *_loop* = true) const

Returns a frame transformation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.

Parameters

in	<i>_time</i>	the time
in	<i>_loop</i>	when true, the time is divided by the duration (see GetLength)

10.147.3.4 unsigned int gazebo::common::NodeAnimation::GetFrameCount () const

Returns the number of key frames.

Returns

the count

10.147.3.5 `void gazebo::common::NodeAnimation::GetKeyFrame (const unsigned int _i, double & _time, math::Matrix4 & _trans) const`

Finds a key frame using the index.

Note the index of a key frame can change as frames are added.

Parameters

in	<i>_i</i>	the index
out	<i>_time</i>	the time of the frame, or -1 if the index id is out of bounds
out	<i>_trans</i>	the transformation for this key frame

10.147.3.6 `std::pair<double, math::Matrix4> gazebo::common::NodeAnimation::GetKeyFrame (const unsigned int _i) const`

Returns a key frame using the index.

Note the index of a key frame can change as frames are added.

Parameters

in	<i>_i</i>	the index
----	-----------	-----------

Returns

a pair that contains the time and transformation. **Time** (p. 1099) is -1 if the index is out of bounds

10.147.3.7 `double gazebo::common::NodeAnimation::GetLength () const`

Returns the duration of the animations.

Returns

the time of the last animation

10.147.3.8 `std::string gazebo::common::NodeAnimation::GetName () const`

Returns the name.

Returns

the name

10.147.3.9 `double gazebo::common::NodeAnimation::GetTimeAtX (const double _x) const`

Returns the time where a transformation's translational value along the X axis is equal to *_x*.

When no transformation is found (within a tolerance of 1e-6), the time is interpolated.

Parameters

in	_x	the value along x. You must ensure that _x is within a valid range.
----	----	---

10.147.3.10 void gazebo::common::NodeAnimation::Scale (const double *_scale*)

Scales each transformation in the key frames.

This only affects the translational values.

Parameters

in	_scale	the scaling factor
----	--------	--------------------

10.147.3.11 void gazebo::common::NodeAnimation::SetName (const std::string & *_name*)

Changes the name of the animation.

Parameters

in	<i>the</i>	new name
----	------------	----------

10.147.4 Member Data Documentation

10.147.4.1 std::map<double, math::Matrix4> gazebo::common::NodeAnimation::keyFrames [protected]

the dictionary of key frames, indexed by time

10.147.4.2 double gazebo::common::NodeAnimation::length [protected]

the duration of the animations (time of last key frame)

10.147.4.3 std::string gazebo::common::NodeAnimation::name [protected]

the name of the animation

The documentation for this class was generated from the following file:

- **SkeletonAnimation.hh**

10.148 gazebo::common::NodeAssignment Class Reference

Vertex to node weighted assignment for skeleton animation visualization.

```
#include <Mesh.hh>
```

Public Member Functions

- **NodeAssignment ()**

Constructor.

Public Attributes

- unsigned int **nodeIndex**
node (or bone) index
- unsigned int **vertexIndex**
index of the vertex
- float **weight**
the weight (between 0 and 1)

10.148.1 Detailed Description

Vertex to node weighted assignment for skeleton animation visualization.

10.148.2 Constructor & Destructor Documentation

10.148.2.1 gazebo::common::NodeAssignment::NodeAssignment ()

Constructor.

10.148.3 Member Data Documentation

10.148.3.1 unsigned int gazebo::common::NodeAssignment::nodeIndex

node (or bone) index

10.148.3.2 unsigned int gazebo::common::NodeAssignment::vertexIndex

index of the vertex

10.148.3.3 float gazebo::common::NodeAssignment::weight

the weight (between 0 and 1)

The documentation for this class was generated from the following file:

- **Mesh.hh**

10.149 gazebo::common::NodeTransform Class Reference

NodeTransform (p. 742) **Skeleton.hh** (p. 1465) common/common.hh

```
#include <Skeleton.hh>
```

Public Types

- enum **TransformType** { **TRANSLATE**, **ROTATE**, **SCALE**, **MATRIX** }
Enumeration of the transform types.

Public Member Functions

- **NodeTransform** (**TransformType** _type=**MATRIX**)
Constructor.
- **NodeTransform** (**math::Matrix4** _mat, std::string _sid="_default_", **TransformType** _type=**MATRIX**)
Constructor.
- **~NodeTransform** ()
Destructor. It does nothing.
- **math::Matrix4** **Get** ()
Returns the transformation matrix.
- std::string **GetSID** ()
Returns thr SID.
- **TransformType** **GetType** ()
Returns the transformation type.
- **math::Matrix4** **operator**() ()
Matrix cast operator.
- **math::Matrix4** **operator*** (**NodeTransform** _t)
Node transform multiplication operator.
- **math::Matrix4** **operator*** (**math::Matrix4** _m)
Matrix multiplication operator.
- void **PrintSource** ()
Prints the transform matrix to std::err stream.
- void **RecalculateMatrix** ()
Sets the transform matrix from the source according to the type.
- void **Set** (**math::Matrix4** _mat)
Assign a transformation.
- void **SetComponent** (unsigned int _idx, double _value)
Set a transformation matrix component value.
- void **SetSID** (std::string _sid)
Set the SID.
- void **SetSourceValues** (**math::Matrix4** _mat)
Set source data values _param[in] _mat the values.
- void **SetSourceValues** (**math::Vector3** _vec)
Set source data values.
- void **SetSourceValues** (**math::Vector3** _axis, double _angle)
Sets source matrix values from roation.
- void **SetType** (**TransformType** _type)
Set transform type.

Protected Attributes

- `std::string` **sid**
the sid
- `std::vector< double >` **source**
source data values (can be a matrix, a position or rotation)
- `math::Matrix4` **transform**
transform
- **TransformType** `type`
transform type

10.149.1 Detailed Description

NodeTransform (p. 742) **Skeleton.hh** (p. 1465) `common/common.hh`

A transformation node

10.149.2 Member Enumeration Documentation

10.149.2.1 `enum gazebo::common::NodeTransform::TransformType`

Enumeration of the transform types.

Enumerator:

TRANSLATE
ROTATE
SCALE
MATRIX

10.149.3 Constructor & Destructor Documentation

10.149.3.1 `gazebo::common::NodeTransform::NodeTransform (TransformType _type = MATRIX)`

Constructor.

Parameters

<code>in</code>	<code>_type</code>	the type of transform
-----------------	--------------------	-----------------------

10.149.3.2 `gazebo::common::NodeTransform::NodeTransform (math::Matrix4 _mat, std::string _sid = "_default_", TransformType _type = MATRIX)`

Constructor.

Parameters

<code>in</code>	<code>_mat</code>	the matrix
<code>in</code>	<code>_sid</code>	identifier
<code>in</code>	<code>_type</code>	the type of transform

10.149.3.3 gazebo::common::NodeTransform::~~NodeTransform ()

Destructor. It does nothing.

10.149.4 Member Function Documentation

10.149.4.1 math::Matrix4 gazebo::common::NodeTransform::Get ()

Returns the transformation matrix.

Returns

the matrix

10.149.4.2 std::string gazebo::common::NodeTransform::GetSID ()

Returns the SID.

Returns

the SID

10.149.4.3 TransformType gazebo::common::NodeTransform::GetType ()

Returns the transformation type.

Returns

the type

10.149.4.4 math::Matrix4 gazebo::common::NodeTransform::operator()()

Matrix cast operator.

Returns

the transform

10.149.4.5 math::Matrix4 gazebo::common::NodeTransform::operator*(NodeTransform _t)

Node transform multiplication operator.

Parameters

in	<code>_t</code>	a transform
----	-----------------	-------------

Returns

transform matrix multiplied by `_t`'s transform

10.149.4.6 `math::Matrix4 gazebo::common::NodeTransform::operator*(math::Matrix4 _m)`

Matrix multiplication operator.

Parameters

<code>in</code>	<code>_m</code>	a matrix
-----------------	-----------------	----------

Returns

transform matrix multiplied by `_m`

10.149.4.7 `void gazebo::common::NodeTransform::PrintSource ()`

Prints the transform matrix to `std::err` stream.

10.149.4.8 `void gazebo::common::NodeTransform::RecalculateMatrix ()`

Sets the transform matrix from the source according to the type.

10.149.4.9 `void gazebo::common::NodeTransform::Set (math::Matrix4 _mat)`

Assign a transformation.

Parameters

<code>in</code>	<code>_mat</code>	the transform
-----------------	-------------------	---------------

10.149.4.10 `void gazebo::common::NodeTransform::SetComponent (unsigned int _idx, double _value)`

Set a transformation matrix component value.

Parameters

<code>in</code>	<code>_idx</code>	the component index
<code>in</code>	<code>_value</code>	the value

10.149.4.11 `void gazebo::common::NodeTransform::SetSID (std::string _sid)`

Set the SID.

Parameters

<code>in</code>	<code>_sid</code>	the sid
-----------------	-------------------	---------

10.149.4.12 void gazebo::common::NodeTransform::SetSourceValues (math::Matrix4 *_mat*)

Set source data values *_param[in]* *_mat* the values.

10.149.4.13 void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 *_vec*)

Set source data values.

10.149.4.14 void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 *_axis*, double *_angle*)

Sets source matrix values from rotation.

Parameters

<i>in</i>	<i>_axis</i>	of rotation
<i>in</i>	<i>_angle</i>	of rotation

10.149.4.15 void gazebo::common::NodeTransform::SetType (TransformType *_type*)

Set transform type.

Parameters

<i>in</i>	<i>_type</i>	the type
-----------	--------------	----------

10.149.5 Member Data Documentation

10.149.5.1 std::string gazebo::common::NodeTransform::sid [protected]

the sid

10.149.5.2 std::vector<double> gazebo::common::NodeTransform::source [protected]

source data values (can be a matrix, a position or rotation)

10.149.5.3 math::Matrix4 gazebo::common::NodeTransform::transform [protected]

transform

10.149.5.4 TransformType gazebo::common::NodeTransform::type [protected]

transform type

The documentation for this class was generated from the following file:

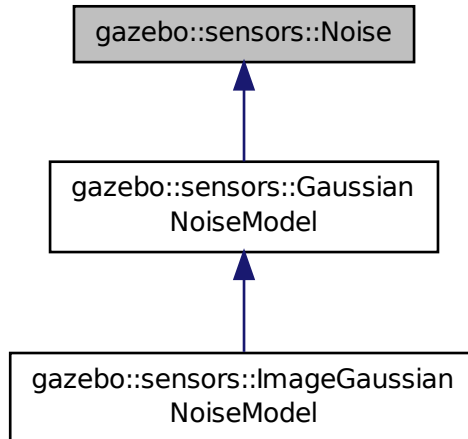
- **Skeleton.hh**

10.150 gazebo::sensors::Noise Class Reference

Noise (p. 748) models for sensor output signals.

```
#include <Noise.hh>
```

Inheritance diagram for gazebo::sensors::Noise:



Public Types

- enum **NoiseType** { **NONE**, **CUSTOM**, **GAUSSIAN** }

Which noise types we support.

Public Member Functions

- **Noise** (**NoiseType** _type)
Constructor.
- virtual **~Noise** ()
Destructor.
- double **Apply** (double _in)
Apply noise to input data value.
- virtual double **ApplyImpl** (double _in)
Apply noise to input data value.
- virtual void **Fini** ()
Finalize the noise model.
- **NoiseType GetNoiseType** () const
Accessor for NoiseType.
- virtual void **Load** (sdf::ElementPtr _sdf)

Load noise parameters from sdf.

- virtual void **SetCamera** (**rendering::CameraPtr** _camera)
Set camera needed to create image noise.
- virtual void **SetCustomNoiseCallback** (boost::function< double(double)> _cb)
Register a custom noise callback.

10.150.1 Detailed Description

Noise (p. 748) models for sensor output signals.

10.150.2 Member Enumeration Documentation

10.150.2.1 enum gazebo::sensors::Noise::NoiseType

Which noise types we support.

Enumerator:

NONE
CUSTOM
GAUSSIAN

10.150.3 Constructor & Destructor Documentation

10.150.3.1 gazebo::sensors::Noise::Noise (NoiseType _type) [explicit]

Constructor.

This should not be called directly unless creating an empty noise model. Use **NoiseFactory::NewNoiseModel** (p. 751) to instantiate a new noise model.

Parameters

in	_type	Type of noise model.
----	-------	----------------------

See Also

NoiseFactory::NewNoiseModel (p. 751)

10.150.3.2 virtual gazebo::sensors::Noise::~~Noise () [virtual]

Destructor.

10.150.4 Member Function Documentation

10.150.4.1 double gazebo::sensors::Noise::Apply (double _in)

Apply noise to input data value.

Parameters

<code>in</code>	<code>_in</code>	Input data value.
-----------------	------------------	-------------------

Returns

Data with noise applied.

10.150.4.2 `virtual double gazebo::sensors::Noise::ApplyImpl (double _in) [virtual]`

Apply noise to input data value.

This gets overridden by derived classes, and called by `Apply`.

Parameters

<code>in</code>	<code>_in</code>	Input data value.
-----------------	------------------	-------------------

Returns

Data with noise applied.

Reimplemented in `gazebo::sensors::GaussianNoiseModel` (p. 459).

10.150.4.3 `virtual void gazebo::sensors::Noise::Fini () [virtual]`

Finalize the noise model.

Reimplemented in `gazebo::sensors::ImageGaussianNoiseModel` (p. 522), and `gazebo::sensors::GaussianNoiseModel` (p. 459).

10.150.4.4 `NoiseType gazebo::sensors::Noise::GetNoiseType () const`

Accessor for `NoiseType`.

Returns

Type of noise currently in use.

10.150.4.5 `virtual void gazebo::sensors::Noise::Load (sdf::ElementPtr _sdf) [virtual]`

Load noise parameters from `sdf`.

Parameters

<code>in</code>	<code>_sdf</code>	SDF parameters.
<code>in</code>	<code>_sensor</code>	Type of sensor.

Reimplemented in `gazebo::sensors::ImageGaussianNoiseModel` (p. 522), and `gazebo::sensors::GaussianNoiseModel` (p. 460).

10.150.4.6 virtual void gazebo::sensors::Noise::SetCamera (rendering::CameraPtr _camera) [virtual]

Set camera needed to create image noise.

This is only needed for image sensors, i.e. camera/multicamera/depth sensors, which use shaders for more efficient noise generation.

Parameters

in	_camera	Camera associated to an image sensor
----	---------	--------------------------------------

Reimplemented in [gazebo::sensors::ImageGaussianNoiseModel](#) (p. 522).

10.150.4.7 virtual void gazebo::sensors::Noise::SetCustomNoiseCallback (boost::function< double(double)> _cb) [virtual]

Register a custom noise callback.

Parameters

in	_cb	Callback function for applying a custom noise model. This is useful if users want to use their own noise model from a sensor plugin.
----	-----	--

The documentation for this class was generated from the following file:

- [Noise.hh](#)

10.151 gazebo::sensors::NoiseFactory Class Reference

Use this noise manager for creating and loading noise models.

```
#include <sensors/sensors.hh>
```

Static Public Member Functions

- static **NoisePtr NewNoiseModel** (sdf::ElementPtr _sdf, const std::string &_sensorType="")
Load a noise model based on the input sdf parameters and sensor type.

10.151.1 Detailed Description

Use this noise manager for creating and loading noise models.

10.151.2 Member Function Documentation

10.151.2.1 static **NoisePtr gazebo::sensors::NoiseFactory::NewNoiseModel** (sdf::ElementPtr _sdf, const std::string &_sensorType = "") [static]

Load a noise model based on the input sdf parameters and sensor type.

Parameters

in	<code>_sdf</code>	Noise (p. 748) sdf parameters.
in	<code>_sensorType</code>	Type of sensor. This is currently used to distinguish between image and non image sensors in order to create the appropriate noise model.

Returns

Pointer to the noise model created.

The documentation for this class was generated from the following file:

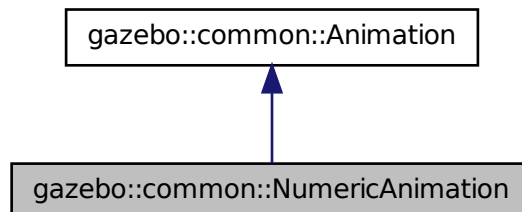
- **Noise.hh**

10.152 gazebo::common::NumericAnimation Class Reference

A numeric animation.

```
#include <Animation.hh>
```

Inheritance diagram for gazebo::common::NumericAnimation:



Public Member Functions

- **NumericAnimation** (const std::string &_name, double _length, bool _loop)
Constructor.
- virtual **~NumericAnimation** ()
Destructor.
- **NumericKeyFrame * CreateKeyFrame** (double _time)
Create a numeric keyframe at the given time.
- void **GetInterpolatedKeyFrame** (**NumericKeyFrame** &_kf) const
Get a keyframe using the animation's current time.

Additional Inherited Members

10.152.1 Detailed Description

A numeric animation.

10.152.2 Constructor & Destructor Documentation

10.152.2.1 `gazebo::common::NumericAnimation::NumericAnimation (const std::string & _name, double _length, bool _loop)`

Constructor.

Parameters

in	<i>_name</i>	String name of the animation. This should be unique.
in	<i>_length</i>	Length of the animation in seconds
in	<i>_loop</i>	True == loop the animation

10.152.2.2 `virtual gazebo::common::NumericAnimation::~~NumericAnimation () [virtual]`

Destructor.

10.152.3 Member Function Documentation

10.152.3.1 `NumericKeyFrame* gazebo::common::NumericAnimation::CreateKeyFrame (double _time)`

Create a numeric keyframe at the given time.

Parameters

in	<i>_time</i>	Time (p. 1099) at which to create the keyframe
----	--------------	---

Returns

Pointer to the new keyframe

10.152.3.2 `void gazebo::common::NumericAnimation::GetInterpolatedKeyFrame (NumericKeyFrame & _kf) const`

Get a keyframe using the animation's current time.

Parameters

out	<i>_kf</i>	NumericKeyFrame (p. 754) reference to hold the interpolated result
-----	------------	---

The documentation for this class was generated from the following file:

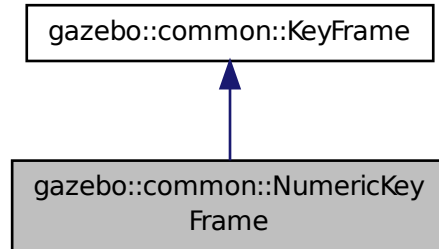
- **Animation.hh**

10.153 gazebo::common::NumericKeyFrame Class Reference

A keyframe for a **NumericAnimation** (p. 752).

```
#include <KeyFrame.hh>
```

Inheritance diagram for gazebo::common::NumericKeyFrame:



Public Member Functions

- **NumericKeyFrame** (double _time)
Constructor.
- virtual **~NumericKeyFrame** ()
Destructor.
- const double & **GetValue** () const
Get the value of the keyframe.
- void **SetValue** (const double &_value)
Set the value of the keyframe.

Protected Attributes

- double **value**
numeric value

10.153.1 Detailed Description

A keyframe for a **NumericAnimation** (p. 752).

10.153.2 Constructor & Destructor Documentation

10.153.2.1 gazebo::common::NumericKeyFrame::NumericKeyFrame (double _time)

Constructor.

Parameters

in	_time	Time (p. 1099) of the keyframe
----	-------	--------------------------------

10.153.2.2 virtual gazebo::common::NumericKeyFrame::~~NumericKeyFrame () [virtual]

Destructor.

10.153.3 Member Function Documentation

10.153.3.1 const double& gazebo::common::NumericKeyFrame::GetValue () const

Get the value of the keyframe.

Returns

the value of the keyframe

10.153.3.2 void gazebo::common::NumericKeyFrame::SetValue (const double & _value)

Set the value of the keyframe.

Parameters

in	_value	The new value
----	--------	---------------

10.153.4 Member Data Documentation

10.153.4.1 double gazebo::common::NumericKeyFrame::value [protected]

numeric value

The documentation for this class was generated from the following file:

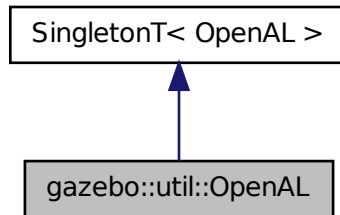
- **KeyFrame.hh**

10.154 gazebo::util::OpenAL Class Reference

3D audio setup and playback.

```
#include <util/util.hh>
```

Inheritance diagram for gazebo::util::OpenAL:



Public Member Functions

- **OpenALSinkPtr CreateSink** (sdf::ElementPtr _sdf)
Create an audio listener.
- **OpenALSourcePtr CreateSource** (sdf::ElementPtr _sdf)
*Create an **OpenALSource** (p. 758) object.*
- void **Fini** ()
Finalize.
- bool **Load** (sdf::ElementPtr _sdf=sdf::ElementPtr())
*Load the **OpenAL** (p. 755) server.*

Additional Inherited Members

10.154.1 Detailed Description

3D audio setup and playback.

10.154.2 Member Function Documentation

10.154.2.1 OpenALSinkPtr gazebo::util::OpenAL::CreateSink (sdf::ElementPtr _sdf)

Create an audio listener.

Currently, only one listener may be created.

Parameters

in	_sdf	SDF element parameters for an audio_source.
----	------	---

Returns

A pointer to an **OpenALSink** (p. 757) object.

10.154.2.2 OpenALSourcePtr gazebo::util::OpenAL::CreateSource (sdf::ElementPtr _sdf)

Create an **OpenALSource** (p. 758) object.

Parameters

in	_sdf	SDF element parameters for an audio_source.
----	------	---

Returns

A pointer to an **OpenALSource** (p. 758) object.

10.154.2.3 void gazebo::util::OpenAL::Fini ()

Finalize.

10.154.2.4 bool gazebo::util::OpenAL::Load (sdf::ElementPtr _sdf = sdf::ElementPtr())

Load the **OpenAL** (p. 755) server.

Returns

True on success.

The documentation for this class was generated from the following file:

- **OpenAL.hh**

10.155 gazebo::util::OpenALSink Class Reference

OpenAL (p. 755) Listener.

```
#include <OpenAL.hh>
```

Public Member Functions

- **OpenALSink** ()
Constructor.
- virtual **~OpenALSink** ()
Destructor.
- bool **SetPose** (const **math::Pose** &_pose)
Set the position of the sink.
- bool **SetVelocity** (const **math::Vector3** &_vel)
Set the velocity of the sink.

10.155.1 Detailed Description

OpenAL (p. 755) Listener.

This can be thought of as a microphone.

10.155.2 Constructor & Destructor Documentation

10.155.2.1 gazebo::util::OpenALSink::OpenALSink ()

Constructor.

10.155.2.2 virtual gazebo::util::OpenALSink::~~OpenALSink () [virtual]

Destructor.

10.155.3 Member Function Documentation

10.155.3.1 bool gazebo::util::OpenALSink::SetPose (const math::Pose & *_pose*)

Set the position of the sink.

Parameters

in	<i>_pose</i>	New pose of the sink.
----	--------------	-----------------------

Returns

True on success.

10.155.3.2 bool gazebo::util::OpenALSink::SetVelocity (const math::Vector3 & *_vel*)

Set the velocity of the sink.

Parameters

in	<i>_vel</i>	Velocity of the sink.
----	-------------	-----------------------

Returns

True on success.

The documentation for this class was generated from the following file:

- **OpenAL.hh**

10.156 gazebo::util::OpenALSource Class Reference

OpenAL (p. 755) Source.

```
#include <OpenAL.hh>
```

Public Member Functions

- **OpenALSource** ()

Constructor.

- virtual **~OpenALSource** ()

Destructor.

- void **FillBufferFromFile** (const std::string &_audioFile)

*Fill the **OpenAL** (p. 755) audio buffer with data from a sound file.*

- bool **FillBufferFromPCM** (uint8_t *_pcmData, unsigned int _dataCount, int _sampleRate)

*Fill the **OpenAL** (p. 755) audio buffer from PCM data.*

- std::vector< std::string > **GetCollisionNames** () const

Get a vector of all the collision names.

- bool **GetOnContact** () const

Return true if the audio source is played on contact with another object.

- bool **HasCollisionName** (const std::string &_name) const

Get whether the source has a collision name set.

- bool **IsPlaying** ()

Is the audio playing.

- bool **Load** (sdf::ElementPtr _sdf)

Load the source from sdf.

- void **Pause** ()

Pause a sound.

- void **Play** ()

Play a sound.

- void **Rewind** ()

Rewind the sound to the beginning.

- bool **SetGain** (float _g)

Set the pitch of the source.

- bool **SetLoop** (bool _state)

Set whether the source loops the audio.

- bool **SetPitch** (float _p)

Set the pitch of the source.

- bool **SetPose** (const **math::Pose** &_pose)

Set the position of the source.

- bool **SetVelocity** (const **math::Vector3** &_vel)

Set the velocity of the source.

- void **Stop** ()

Stop a sound.

10.156.1 Detailed Description

OpenAL (p. 755) Source.

This can be thought of as a speaker.

10.156.2 Constructor & Destructor Documentation

10.156.2.1 gazebo::util::OpenALSource::OpenALSource ()

Constructor.

10.156.2.2 `virtual gazebo::util::OpenALSource::~~OpenALSource () [virtual]`

Destructor.

10.156.3 Member Function Documentation

10.156.3.1 `void gazebo::util::OpenALSource::FillBufferFromFile (const std::string & _audioFile)`

Fill the **OpenAL** (p. 755) audio buffer with data from a sound file.

Parameters

<code>in</code>	<code>_audioFile</code>	Name and an audio file.
-----------------	-------------------------	-------------------------

10.156.3.2 `bool gazebo::util::OpenALSource::FillBufferFromPCM (uint8_t * _pcmData, unsigned int _dataCount, int _sampleRate)`

Fill the **OpenAL** (p. 755) audio buffer from PCM data.

Parameters

<code>in</code>	<code>_pcmData</code>	Pointer to the PCM audio data.
<code>in</code>	<code>_dataCount</code>	Size of the PCM data.
<code>in</code>	<code>_sampleRate</code>	Sample rate for the PCM data.

Returns

True on success.

10.156.3.3 `std::vector<std::string> gazebo::util::OpenALSource::GetCollisionNames () const`

Get a vector of all the collision names.

Returns

All the collision names used to trigger audio playback on contact.

10.156.3.4 `bool gazebo::util::OpenALSource::GetOnContact () const`

Return true if the audio source is played on contact with another object.

Contact is determine based on a set of collision objects.

Returns

True if audio is played on contact.

See Also

AddCollision()

10.156.3.5 `bool gazebo::util::OpenALSource::HasCollisionName (const std::string & _name) const`

Get whether the source has a collision name set.

Parameters

<code><i>in</i></code>	<code><i>_name</i></code>	Name of a collision to check for.
------------------------	---------------------------	-----------------------------------

Returns

True if the collision name was found.

10.156.3.6 `bool gazebo::util::OpenALSource::IsPlaying ()`

Is the audio playing.

10.156.3.7 `bool gazebo::util::OpenALSource::Load (sdf::ElementPtr _sdf)`

Load the source from sdf.

Parameters

<code><i>in</i></code>	<code><i>_sdf</i></code>	SDF element parameters for an audio_source.
------------------------	--------------------------	---

Returns

True on success.

10.156.3.8 `void gazebo::util::OpenALSource::Pause ()`

Pause a sound.

10.156.3.9 `void gazebo::util::OpenALSource::Play ()`

Play a sound.

10.156.3.10 `void gazebo::util::OpenALSource::Rewind ()`

Rewind the sound to the beginning.

10.156.3.11 `bool gazebo::util::OpenALSource::SetGain (float _g)`

Set the pitch of the source.

Parameters

<code><i>in</i></code>	<code><i>_g</i></code>	Gain value.
------------------------	------------------------	-------------

Returns

True on success.

10.156.3.12 `bool gazebo::util::OpenALSource::SetLoop (bool _state)`

Set whether the source loops the audio.

Parameters

<code>in</code>	<code><i>_state</i></code>	True to cause playback to loop.
-----------------	----------------------------	---------------------------------

Returns

True on success.

10.156.3.13 `bool gazebo::util::OpenALSource::SetPitch (float _p)`

Set the pitch of the source.

Parameters

<code>in</code>	<code><i>_p</i></code>	Pitch value.
-----------------	------------------------	--------------

Returns

True on success.

10.156.3.14 `bool gazebo::util::OpenALSource::SetPose (const math::Pose & _pose)`

Set the position of the source.

Parameters

<code>in</code>	<code><i>_pose</i></code>	New pose of the source.
-----------------	---------------------------	-------------------------

Returns

True on success.

10.156.3.15 `bool gazebo::util::OpenALSource::SetVelocity (const math::Vector3 & _vel)`

Set the velocity of the source.

Parameters

<code>in</code>	<code><i>_vel</i></code>	New velocity of the source.
-----------------	--------------------------	-----------------------------

Returns

True on success.

10.156.3.16 void gazebo::util::OpenALSource::Stop ()

Stop a sound.

The documentation for this class was generated from the following file:

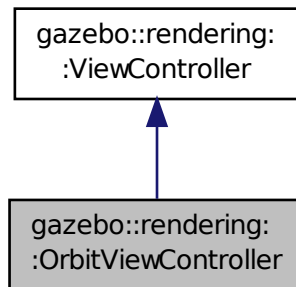
- **OpenAL.hh**

10.157 gazebo::rendering::OrbitViewController Class Reference

Orbit view controller.

```
#include <OrbitViewController.hh>
```

Inheritance diagram for gazebo::rendering::OrbitViewController:

**Public Member Functions**

- **OrbitViewController** (**UserCameraPtr** _camera)
Constructor.
- virtual **~OrbitViewController** ()
Destructor.
- **math::Vector3 GetFocalPoint** () const
Get the focal point.
- virtual void **HandleKeyPressEvent** (const std::string &_key)
Handle a key press event.
- void **HandleKeyReleaseEvent** (const std::string &_key)
Handle a key release event.
- virtual void **HandleMouseEvent** (const **common::MouseEvent** &_event)

- Handle a mouse event.*
- virtual void **Init** ()
Initialize the controller.
- virtual void **Init** (const **math::Vector3** &_focalPoint)
Initialize the controller with a focal point.
- void **SetDistance** (float _d)
Set the distance to the focal point.
- void **SetFocalPoint** (const **math::Vector3** &_fp)
Set the focal point.
- virtual void **Update** ()
Update.

Static Public Member Functions

- static std::string **GetTypeString** ()
Get the type name of this view controller.

Additional Inherited Members

10.157.1 Detailed Description

Orbit view controller.

10.157.2 Constructor & Destructor Documentation

10.157.2.1 gazebo::rendering::OrbitViewController::OrbitViewController (**UserCameraPtr** _camera)

Constructor.

Parameters

in	_camera	Pointer to the camera to control.
----	---------	-----------------------------------

10.157.2.2 virtual gazebo::rendering::OrbitViewController::~~OrbitViewController () [virtual]

Destructor.

10.157.3 Member Function Documentation

10.157.3.1 **math::Vector3** gazebo::rendering::OrbitViewController::GetFocalPoint () const

Get the focal point.

Returns

The focal point

10.157.3.2 `static std::string gazebo::rendering::OrbitViewController::GetTypeString () [static]`

Get the type name of this view controller.

Returns

The view controller name: "orbit".

10.157.3.3 `virtual void gazebo::rendering::OrbitViewController::HandleKeyPressEvent (const std::string & _key) [virtual]`

Handle a key press event.

Parameters

in	<code>_key</code>	The key that was pressed.
----	-------------------	---------------------------

Implements `gazebo::rendering::ViewController` (p. 1194).

10.157.3.4 `void gazebo::rendering::OrbitViewController::HandleKeyReleaseEvent (const std::string & _key) [virtual]`

Handle a key release event.

Parameters

in	<code>_key</code>	The key that was released.
----	-------------------	----------------------------

Implements `gazebo::rendering::ViewController` (p. 1194).

10.157.3.5 `virtual void gazebo::rendering::OrbitViewController::HandleMouseEvent (const common::MouseEvent & _event) [virtual]`

Handle a mouse event.

Parameters

in	<code>_event</code>	The mouse event.
----	---------------------	------------------

Implements `gazebo::rendering::ViewController` (p. 1195).

10.157.3.6 `virtual void gazebo::rendering::OrbitViewController::Init () [virtual]`

Initialize the controller.

Implements `gazebo::rendering::ViewController` (p. 1195).

10.157.3.7 `virtual void gazebo::rendering::OrbitViewController::Init (const math::Vector3 & _focalPoint) [virtual]`

Initialize the controller with a focal point.

Parameters

in	<code>_focalPoint</code>	Point to look at.
----	--------------------------	-------------------

Reimplemented from `gazebo::rendering::ViewController` (p. 1195).

10.157.3.8 void gazebo::rendering::OrbitViewController::SetDistance (float *d*)

Set the distance to the focal point.

Parameters

in	<code>_d</code>	The distance from the focal point.
----	-----------------	------------------------------------

10.157.3.9 void gazebo::rendering::OrbitViewController::SetFocalPoint (const math::Vector3 & *fp*)

Set the focal point.

Parameters

in	<code>_fp</code>	The focal point
----	------------------	-----------------

10.157.3.10 virtual void gazebo::rendering::OrbitViewController::Update () [virtual]

Update.

Implements `gazebo::rendering::ViewController` (p. 1195).

The documentation for this class was generated from the following file:

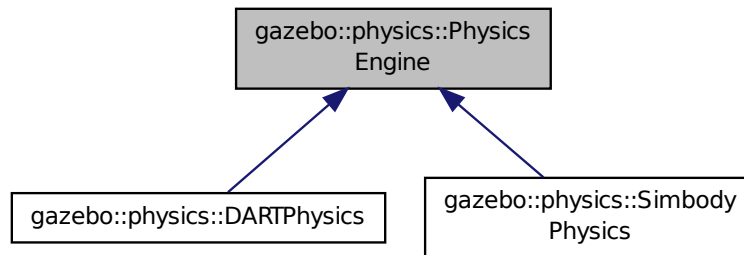
- `OrbitViewController.hh`

10.158 gazebo::physics::PhysicsEngine Class Reference

Base (p. 168) class for a physics engine.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::PhysicsEngine:



Public Member Functions

- **PhysicsEngine** (**WorldPtr** _world)
 - Default constructor.*
- virtual **~PhysicsEngine** ()
 - Destructor.*
- virtual **CollisionPtr CreateCollision** (const std::string &_shapeType, **LinkPtr** _link)=0
 - Create a collision.*
- **CollisionPtr CreateCollision** (const std::string &_shapeType, const std::string &_linkName)
 - Create a collision.*
- virtual **JointPtr CreateJoint** (const std::string &_type, **ModelPtr** _parent=**ModelPtr**())=0
 - Create a new joint.*
- virtual **LinkPtr CreateLink** (**ModelPtr** _parent)=0
 - Create a new body.*
- virtual **ModelPtr CreateModel** (**BasePtr** _base)
 - Create a new model.*
- virtual **ShapePtr CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)=0
 - Create a **physics::Shape** (p. 932) object.*
- virtual void **DebugPrint** () const =0
 - Debug print out of the physic engine state.*
- virtual void **Fini** ()
 - Finilize the physics engine.*
- virtual bool **GetAutoDisableFlag** ()
 - : Remove this function, and replace it with a more generic property map*
- **ContactManager * GetContactManager** () const
 - Get a pointer to the contact manger.*
- virtual double **GetContactMaxCorrectingVel** ()
 - : Remove this function, and replace it with a more generic property map.*
- virtual double **GetContactSurfaceLayer** ()
 - : Remove this function, and replace it with a more generic property map.*
- virtual **math::Vector3 GetGravity** () const

- Return the gavity vector.*

 - virtual unsigned int **GetMaxContacts** ()
 - : Remove this function, and replace it with a more generic property map.
 - double **GetMaxStepSize** () const
 - Get max step size.*
 - virtual boost::any **GetParam** (const std::string &_key) const
 - Get an parameter of the physics engine.*
 - boost::recursive_mutex * **GetPhysicsUpdateMutex** () const
 - returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 780).*
 - double **GetRealTimeUpdateRate** () const
 - Get real time update rate.*
 - virtual int **GetSORPGSIters** () **GAZEBO_DEPRECATED**(3.0)
 - : Remove this function, and replace it with a more generic property map
 - virtual int **GetSORPGSPreconIters** () **GAZEBO_DEPRECATED**(3.0)
 - : Remove this function, and replace it with a more generic property map
 - virtual double **GetSORPGSW** () **GAZEBO_DEPRECATED**(3.0)
 - : Remove this function, and replace it with a more generic property map.
 - double **GetTargetRealTimeFactor** () const
 - Get target real time factor.*
 - virtual std::string **GetType** () const =0
 - Return the physics engine type (ode|bullet|dart|simbody).*
 - double **GetUpdatePeriod** ()
 - Get the simulation update period.*
 - virtual double **GetWorldCFM** ()
 - : Remove this function, and replace it with a more generic property map
 - virtual double **GetWorldERP** ()
 - : Remove this function, and replace it with a more generic property map
 - virtual void **Init** ()=0
 - Initialize the physics engine.*
 - virtual void **InitForThread** ()=0
 - Init the engine for threads.*
 - virtual void **Load** (sdf::ElementPtr _sdf)
 - Load the physics engine.*
 - virtual void **Reset** ()
 - Rest the physics engine.*
 - virtual void **SetAutoDisableFlag** (bool _autoDisable)
 - : Remove this function, and replace it with a more generic property map
 - virtual void **SetContactMaxCorrectingVel** (double _vel)
 - : Remove this function, and replace it with a more generic property map
 - virtual void **SetContactSurfaceLayer** (double _layerDepth)
 - : Remove this function, and replace it with a more generic property map
 - virtual void **SetGravity** (const gazebo::math::Vector3 &_gravity)=0
 - Set the gavity vector.*
 - virtual void **SetMaxContacts** (unsigned int _maxContacts)
 - : Remove this function, and replace it with a more generic property map
 - void **SetMaxStepSize** (double _stepSize)
 - Set max step size.*

- virtual bool **SetParam** (const std::string &_key, const boost::any &_value)
Set a parameter of the physics engine.
- void **SetRealTimeUpdateRate** (double _rate)
Set real time update rate.
- virtual void **SetSeed** (uint32_t _seed)=0
Set the random number seed for the physics engine.
- virtual void **SetSORPGSIters** (unsigned int _iters) **GAZEBO_DEPRECATED(3.0)**
: Remove this function, and replace it with a more generic property map
- virtual void **SetSORPGSPreconIters** (unsigned int _iters) **GAZEBO_DEPRECATED(3.0)**
: Remove this function, and replace it with a more generic property map
- virtual void **SetSORPGSW** (double _w) **GAZEBO_DEPRECATED(3.0)**
: Remove this function, and replace it with a more generic property map
- void **SetTargetRealTimeFactor** (double _factor)
Set target real time factor.
- virtual void **SetWorldCFM** (double _cfm)
: Remove this function, and replace it with a more generic property map
- virtual void **SetWorldERP** (double _erp)
: Remove this function, and replace it with a more generic property map
- virtual void **UpdateCollision** ()=0
Update the physics engine collision.
- virtual void **UpdatePhysics** ()
Update the physics engine.

Protected Member Functions

- virtual void **OnPhysicsMsg** (ConstPhysicsPtr &_msg)
virtual callback for gztopic "~/physics".
- virtual void **OnRequest** (ConstRequestPtr &_msg)
virtual callback for gztopic "~/request".

Protected Attributes

- **ContactManager * contactManager**
Class that handles all contacts generated by the physics engine.
- double **maxStepSize**
Real time update rate.
- **transport::NodePtr node**
Node for communication.
- **transport::SubscriberPtr physicsSub**
Subscribe to the physics topic.
- boost::recursive_mutex * **physicsUpdateMutex**
Mutex to protect the update cycle.
- double **realTimeUpdateRate**
Real time update rate.
- **transport::SubscriberPtr requestSub**
Subscribe to the request topic.

- **transport::PublisherPtr responsePub**

Response publisher.

- sdf::ElementPtr **sdf**

Our SDF values.

- double **targetRealTimeFactor**

Target real time factor.

- **WorldPtr world**

Pointer to the world.

10.158.1 Detailed Description

Base (p. 168) class for a physics engine.

10.158.2 Constructor & Destructor Documentation

10.158.2.1 gazebo::physics::PhysicsEngine::PhysicsEngine (WorldPtr *_world*) [explicit]

Default constructor.

Parameters

in	<i>_world</i>	Pointer to the world.
----	---------------	-----------------------

10.158.2.2 virtual gazebo::physics::PhysicsEngine::~PhysicsEngine () [virtual]

Destructor.

10.158.3 Member Function Documentation

10.158.3.1 virtual CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision (const std::string & *_shapeType*, LinkPtr *_link*) [pure virtual]

Create a collision.

Parameters

in	<i>_shapeType</i>	Type of collision to create.
in	<i>_link</i>	Parent link.

Implemented in **gazebo::physics::DARTPhysics** (p. 356), and **gazebo::physics::SimbodyPhysics** (p. 989).

10.158.3.2 CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision (const std::string & *_shapeType*, const std::string & *_linkName*)

Create a collision.

Parameters

in	<code>_shapeType</code>	Type of collision to create.
in	<code>_linkName</code>	Name of the parent link.

10.158.3.3 `virtual JointPtr gazebo::physics::PhysicsEngine::CreateJoint (const std::string & _type, ModelPtr _parent = ModelPtr ()) [pure virtual]`

Create a new joint.

Parameters

in	<code>_type</code>	Type of joint to create.
in	<code>_parent</code>	Model (p. 678) parent.

Implemented in **gazebo::physics::DARTPhysics** (p. 356), and **gazebo::physics::SimbodyPhysics** (p. 989).

10.158.3.4 `virtual LinkPtr gazebo::physics::PhysicsEngine::CreateLink (ModelPtr _parent) [pure virtual]`

Create a new body.

Parameters

in	<code>_parent</code>	Parent model for the link.
----	----------------------	----------------------------

Implemented in **gazebo::physics::DARTPhysics** (p. 356), and **gazebo::physics::SimbodyPhysics** (p. 990).

10.158.3.5 `virtual ModelPtr gazebo::physics::PhysicsEngine::CreateModel (BasePtr _base) [virtual]`

Create a new model.

Parameters

in	<code>_base</code>	Boost shared pointer to a new model.
----	--------------------	--------------------------------------

Reimplemented in **gazebo::physics::DARTPhysics** (p. 357), and **gazebo::physics::SimbodyPhysics** (p. 990).

10.158.3.6 `virtual ShapePtr gazebo::physics::PhysicsEngine::CreateShape (const std::string & _shapeType, CollisionPtr _collision) [pure virtual]`

Create a **physics::Shape** (p. 932) object.

Parameters

in	<code>_shapeType</code>	Type of shape to create.
in	<code>_collision</code>	Collision (p. 235) parent.

Implemented in **gazebo::physics::DARTPhysics** (p. 357), and **gazebo::physics::SimbodyPhysics** (p. 990).

10.158.3.7 `virtual void gazebo::physics::PhysicsEngine::DebugPrint () const [pure virtual]`

Debug print out of the physic engine state.

Implemented in `gazebo::physics::DARTPhysics` (p. 357), and `gazebo::physics::SimbodyPhysics` (p. 990).

10.158.3.8 `virtual void gazebo::physics::PhysicsEngine::Fini () [virtual]`

Finilize the physics engine.

Reimplemented in `gazebo::physics::DARTPhysics` (p. 357), and `gazebo::physics::SimbodyPhysics` (p. 990).

10.158.3.9 `virtual bool gazebo::physics::PhysicsEngine::GetAutoDisableFlag () [inline],[virtual]`

: Remove this function, and replace it with a more generic property map access functions to set ODE parameters..

Returns

Auto disable flag.

10.158.3.10 `ContactManager* gazebo::physics::PhysicsEngine::GetContactManager () const`

Get a pointer to the contact manger.

Returns

Pointer to the contact manager.

10.158.3.11 `virtual double gazebo::physics::PhysicsEngine::GetContactMaxCorrectingVel () [inline],[virtual]`

: Remove this function, and replace it with a more generic property map access functions to set ODE parameters.

Returns

Max correcting velocity.

10.158.3.12 `virtual double gazebo::physics::PhysicsEngine::GetContactSurfaceLayer () [inline],[virtual]`

: Remove this function, and replace it with a more generic property map access functions to set ODE parameters.

Returns

Contact (p. 279) suerface layer depth.

10.158.3.13 `virtual math::Vector3 gazebo::physics::PhysicsEngine::GetGravity () const [virtual]`

Return the gavity vector.

Returns

The gavity vector.

10.158.3.14 `virtual unsigned int gazebo::physics::PhysicsEngine::GetMaxContacts () [inline], [virtual]`

: Remove this function, and replace it with a more generic property map.

access functions to set ODE parameters.

Returns

Maximum number of allows contacts.

10.158.3.15 `double gazebo::physics::PhysicsEngine::GetMaxStepSize () const`

Get max step size.

Returns

Max step size.

10.158.3.16 `virtual boost::any gazebo::physics::PhysicsEngine::GetParam (const std::string & _key) const [virtual]`

Get an parameter of the physics engine.

Parameters

in	_attr	String key
----	-------	------------

See Also

SetParam (p. 777)

Returns

The value of the parameter

Reimplemented in **gazebo::physics::SimbodyPhysics** (p. 991), and **gazebo::physics::DARTPhysics** (p. 357).

10.158.3.17 `boost::recursive_mutex* gazebo::physics::PhysicsEngine::GetPhysicsUpdateMutex () const [inline]`

returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 780).

Returns

Pointer to the physics mutex.

10.158.3.18 `double gazebo::physics::PhysicsEngine::GetRealTimeUpdateRate () const`

Get real time update rate.

Returns

Update rate

10.158.3.19 `virtual int gazebo::physics::PhysicsEngine::GetSORPGSIters () [inline],[virtual]`

: Remove this function, and replace it with a more generic property map access functions to set ODE parameters.

Returns

SORPGS iterations.

10.158.3.20 `virtual int gazebo::physics::PhysicsEngine::GetSORPGSPreconIters () [inline],[virtual]`

: Remove this function, and replace it with a more generic property map access functions to set ODE parameters.

Returns

SORPGS precondition iterations.

10.158.3.21 `virtual double gazebo::physics::PhysicsEngine::GetSORPGSW () [inline],[virtual]`

: Remove this function, and replace it with a more generic property map access functions to set ODE parameters

Returns

SORPGSW value.

10.158.3.22 `double gazebo::physics::PhysicsEngine::GetTargetRealTimeFactor () const`

Get target real time factor.

Returns

Target real time factor

10.158.3.23 `virtual std::string gazebo::physics::PhysicsEngine::GetType () const [pure virtual]`

Return the physics engine type (ode|bullet|dart|simbody).

Returns

Type of the physics engine.

Implemented in `gazebo::physics::DARTPhysics` (p. 358), and `gazebo::physics::SimbodyPhysics` (p. 991).

10.158.3.24 `double gazebo::physics::PhysicsEngine::GetUpdatePeriod ()`

Get the simulation update period.

Returns

Simulation update period.

10.158.3.25 `virtual double gazebo::physics::PhysicsEngine::GetWorldCFM ()` `[inline],[virtual]`

: Remove this function, and replace it with a more generic property map

Get **World** (p. 1239) CFM.

Returns

World (p. 1239) CFM.

10.158.3.26 `virtual double gazebo::physics::PhysicsEngine::GetWorldERP ()` `[inline],[virtual]`

: Remove this function, and replace it with a more generic property map

Get **World** (p. 1239) ERP.

Returns

World (p. 1239) ERP.

10.158.3.27 `virtual void gazebo::physics::PhysicsEngine::Init ()` `[pure virtual]`

Initialize the physics engine.

Implemented in **gazebo::physics::DARTPhysics** (p. 358), and **gazebo::physics::SimbodyPhysics** (p. 992).

10.158.3.28 `virtual void gazebo::physics::PhysicsEngine::InitForThread ()` `[pure virtual]`

Init the engine for threads.

Implemented in **gazebo::physics::DARTPhysics** (p. 358), and **gazebo::physics::SimbodyPhysics** (p. 992).

10.158.3.29 `virtual void gazebo::physics::PhysicsEngine::Load (sdf::ElementPtr _sdf)` `[virtual]`

Load the physics engine.

Parameters

<code>in</code>	<code>_sdf</code>	Pointer to the SDF parameters.
-----------------	-------------------	--------------------------------

Reimplemented in **gazebo::physics::DARTPhysics** (p. 359), and **gazebo::physics::SimbodyPhysics** (p. 992).

10.158.3.30 `virtual void gazebo::physics::PhysicsEngine::OnPhysicsMsg (ConstPhysicsPtr & _msg) [protected], [virtual]`

virtual callback for gztopic "~/physics".

Parameters

in	<code>_msg</code>	Physics message.
----	-------------------	------------------

Reimplemented in `gazebo::physics::SimbodyPhysics` (p. 992), and `gazebo::physics::DARTPhysics` (p. 359).

10.158.3.31 `virtual void gazebo::physics::PhysicsEngine::OnRequest (ConstRequestPtr & _msg) [protected], [virtual]`

virtual callback for gztopic "~/request".

Parameters

in	<code>_msg</code>	Request message.
----	-------------------	------------------

Reimplemented in `gazebo::physics::SimbodyPhysics` (p. 993), and `gazebo::physics::DARTPhysics` (p. 359).

10.158.3.32 `virtual void gazebo::physics::PhysicsEngine::Reset () [inline], [virtual]`

Rest the physics engine.

Reimplemented in `gazebo::physics::DARTPhysics` (p. 359), and `gazebo::physics::SimbodyPhysics` (p. 994).

10.158.3.33 `virtual void gazebo::physics::PhysicsEngine::SetAutoDisableFlag (bool _autoDisable) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	<code>_autoDisable</code>	True to enable auto disabling of bodies.
----	---------------------------	--

10.158.3.34 `virtual void gazebo::physics::PhysicsEngine::SetContactMaxCorrectingVel (double _vel) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	<code>_vel</code>	Max correcting velocity.
----	-------------------	--------------------------

10.158.3.35 `virtual void gazebo::physics::PhysicsEngine::SetContactSurfaceLayer (double _layerDepth) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	<i>_layerDepth</i>	Surface layer depth
----	--------------------	---------------------

10.158.3.36 `virtual void gazebo::physics::PhysicsEngine::SetGravity (const gazebo::math::Vector3 & _gravity) [pure virtual]`

Set the gavity vector.

Parameters

in	<i>_gravity</i>	New gravity vector.
----	-----------------	---------------------

Implemented in **`gazebo::physics::DARTPhysics`** (p. 359), and **`gazebo::physics::SimbodyPhysics`** (p. 994).

10.158.3.37 `virtual void gazebo::physics::PhysicsEngine::SetMaxContacts (unsigned int _maxContacts) [virtual]`

: Remove this function, and replace it with a more generic property map

access functions to set ODE parameters

Parameters

in	<i>_maxContacts</i>	Maximum number of contacts.
----	---------------------	-----------------------------

10.158.3.38 `void gazebo::physics::PhysicsEngine::SetMaxStepSize (double _stepSize)`

Set max step size.

Parameters

in	<i>_stepSize</i>	Max step size.
----	------------------	----------------

10.158.3.39 `virtual bool gazebo::physics::PhysicsEngine::SetParam (const std::string & _key, const boost::any & _value) [virtual]`

Set a parameter of the physics engine.

See `SetParam` documentation for descriptions of duplicate parameters.

Parameters

in	_key	String key Below is a list of _key parameter definitions: <ol style="list-style-type: none"> 1. "solver_type" (string) - returns solver used by engine, e.g. "sequential_impulse" for Bullet, "quick" for ODE "Featherstone and Lemkes" for DART and "Spatial Algebra and Elastic Foundation" for Simbody. -# "type" (string) - deprecated, use keyword "solver_type". -# "cfm" (double) - global CFM -# "erp" (double) - global ERP -# "precon_iters" (bool) - precondition iterations (experimental). 2. "iters" (int) - number of LCP PGS iterations. If sor_lcp_tolerance is negative, full iteration count is executed. Otherwise, PGS may stop iteration early if sor_lcp_tolerance is satisfied by the total RMS residual. 3. "sor" (double) - relaxation parameter for Projected Gauss-Seidel (PGS) updates. 4. "contact_max_correcting_vel" (double) - truncates correction impulses from ERP by this value. 5. "contact_surface_layer" (double) - ERP is 0 for interpenetration depths below this value. 6. "max_contacts" (int) - max number of contact constraints between any pair of collision bodies. 7. "min_step_size" (double) - minimum internal step size. (defined but not used in ode). 8. "max_step_size" (double) - maximum physics step size when physics update step must return.
in	_value	The value to set to

Returns

true if SetParam is successful, false if operation fails.

Reimplemented in **gazebo::physics::SimbodyPhysics** (p. 994), and **gazebo::physics::DARTPhysics** (p. 360).

10.158.3.40 void gazebo::physics::PhysicsEngine::SetRealTimeUpdateRate (double _rate)

Set real time update rate.

Parameters

in	_rate	Update rate
----	-------	-------------

10.158.3.41 virtual void gazebo::physics::PhysicsEngine::SetSeed (uint32_t _seed) [pure virtual]

Set the random number seed for the physics engine.

Parameters

in	_seed	The random number seed.
----	-------	-------------------------

Implemented in **gazebo::physics::DARTPhysics** (p. 360), and **gazebo::physics::SimbodyPhysics** (p. 995).

10.158.3.42 `virtual void gazebo::physics::PhysicsEngine::SetSORPGSIter (unsigned int _iters) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

<code>in</code>	<code><i>_iter</i></code>	Number of iterations.
-----------------	---------------------------	-----------------------

10.158.3.43 `virtual void gazebo::physics::PhysicsEngine::SetSORPGSPreconlters (unsigned int _iters) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

<code>in</code>	<code><i>_iter</i></code>	Number of iterations.
-----------------	---------------------------	-----------------------

10.158.3.44 `virtual void gazebo::physics::PhysicsEngine::SetSORPGSW (double _w) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

<code>in</code>	<code><i>_w</i></code>	SORPGSW value.
-----------------	------------------------	----------------

10.158.3.45 `void gazebo::physics::PhysicsEngine::SetTargetRealTimeFactor (double _factor)`

Set target real time factor.

Parameters

<code>in</code>	<code><i>_factor</i></code>	Target real time factor
-----------------	-----------------------------	-------------------------

10.158.3.46 `virtual void gazebo::physics::PhysicsEngine::SetWorldCFM (double _cfm) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

<code>in</code>	<code><i>_cfm</i></code>	Constraint force mixing.
-----------------	--------------------------	--------------------------

10.158.3.47 `virtual void gazebo::physics::PhysicsEngine::SetWorldERP (double _erp) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	_erp	Error reduction parameter.
----	------	----------------------------

10.158.3.48 `virtual void gazebo::physics::PhysicsEngine::UpdateCollision () [pure virtual]`

Update the physics engine collision.

Implemented in `gazebo::physics::DARTPhysics` (p. 361), and `gazebo::physics::SimbodyPhysics` (p. 996).

10.158.3.49 `virtual void gazebo::physics::PhysicsEngine::UpdatePhysics () [inline],[virtual]`

Update the physics engine.

Reimplemented in `gazebo::physics::DARTPhysics` (p. 361), and `gazebo::physics::SimbodyPhysics` (p. 996).

10.158.4 Member Data Documentation

10.158.4.1 `ContactManager* gazebo::physics::PhysicsEngine::contactManager [protected]`

Class that handles all contacts generated by the physics engine.

10.158.4.2 `double gazebo::physics::PhysicsEngine::maxStepSize [protected]`

Real time update rate.

10.158.4.3 `transport::NodePtr gazebo::physics::PhysicsEngine::node [protected]`

Node for communication.

10.158.4.4 `transport::SubscriberPtr gazebo::physics::PhysicsEngine::physicsSub [protected]`

Subscribe to the physics topic.

10.158.4.5 `boost::recursive_mutex* gazebo::physics::PhysicsEngine::physicsUpdateMutex [protected]`

Mutex to protect the update cycle.

10.158.4.6 `double gazebo::physics::PhysicsEngine::realTimeUpdateRate [protected]`

Real time update rate.

10.158.4.7 `transport::SubscriberPtr gazebo::physics::PhysicsEngine::requestSub` [protected]

Subscribe to the request topic.

10.158.4.8 `transport::PublisherPtr gazebo::physics::PhysicsEngine::responsePub` [protected]

Response publisher.

10.158.4.9 `sdf::ElementPtr gazebo::physics::PhysicsEngine::sdf` [protected]

Our SDF values.

10.158.4.10 `double gazebo::physics::PhysicsEngine::targetRealTimeFactor` [protected]

Target real time factor.

10.158.4.11 `WorldPtr gazebo::physics::PhysicsEngine::world` [protected]

Pointer to the world.

The documentation for this class was generated from the following file:

- **PhysicsEngine.hh**

10.159 gazebo::physics::PhysicsFactory Class Reference

The physics factory instantiates different physics engines.

```
#include <physics/physics.hh>
```

Static Public Member Functions

- static bool **IsRegistered** (const std::string &_name)
Check if a physics engine is registered.
- static **PhysicsEnginePtr NewPhysicsEngine** (const std::string &_className, **WorldPtr** _world)
Create a new instance of a physics engine.
- static void **RegisterAll** ()
Register everything.
- static void **RegisterPhysicsEngine** (std::string _className, **PhysicsFactoryFn** _factoryfn)
Register a physics class.

10.159.1 Detailed Description

The physics factory instantiates different physics engines.

10.159.2 Member Function Documentation

10.159.2.1 `static bool gazebo::physics::PhysicsFactory::IsRegistered (const std::string & _name) [static]`

Check if a physics engine is registered.

Parameters

in	<code><i>_name</i></code>	Name of the physics engine.
----	---------------------------	-----------------------------

Returns

True if physics engine is registered, false otherwise.

10.159.2.2 `static PhysicsEnginePtr gazebo::physics::PhysicsFactory::NewPhysicsEngine (const std::string & _className, WorldPtr _world) [static]`

Create a new instance of a physics engine.

Parameters

in	<code><i>_className</i></code>	Name of the physics class.
in	<code><i>_world</i></code>	World (p. 1239) to pass to the created physics engine.

10.159.2.3 `static void gazebo::physics::PhysicsFactory::RegisterAll () [static]`

Register everything.

10.159.2.4 `static void gazebo::physics::PhysicsFactory::RegisterPhysicsEngine (std::string _className, PhysicsFactoryFn _factoryfn) [static]`

Register a physics class.

Parameters

in	<code><i>_className</i></code>	Name of the physics class.
in	<code><i>_factoryfn</i></code>	Function pointer used to create a physics engine.

The documentation for this class was generated from the following file:

- **PhysicsFactory.hh**

10.160 gazebo::common::PID Class Reference

Generic **PID** (p. 782) controller class.

```
#include <common/common.hh>
```

Public Member Functions

- **PID** (double _p=0.0, double _i=0.0, double _d=0.0, double _imax=0.0, double _imin=0.0, double _cmdMax=0.0, double _cmdMin=0.0)
Constructor, zeros out Pid values when created and initialize Pid-gains and integral term limits:[iMax:iMin]-[I1:I2].
- virtual \sim **PID** ()
Destructor.
- double **GetCmd** ()
*Return current command for this **PID** (p. 782) controller.*
- double **GetCmdMax** () const
Get the maximum value for the command.
- double **GetCmdMin** () const
Get the maximum value for the command.
- double **GetDGain** () const
Get the derivative Gain.
- void **GetErrors** (double &_pe, double &_ie, double &_de)
*Return **PID** (p. 782) error terms for the controller.*
- double **GetIGain** () const
Get the integral Gain.
- double **GetIMax** () const
Get the integral upper limit.
- double **GetIMin** () const
Get the integral lower limit.
- double **GetPGain** () const
Get the proportional Gain.
- void **Init** (double _p=0.0, double _i=0.0, double _d=0.0, double _imax=0.0, double _imin=0.0, double _cmdMax=0.0, double _cmdMin=0.0)
Initialize PID-gains and integral term limits:[iMax:iMin]-[I1:I2].
- **PID & operator=** (const **PID** &_p)
Assignment operator.
- void **Reset** ()
Reset the errors and command.
- void **SetCmd** (double _cmd)
*Set current target command for this **PID** (p. 782) controller.*
- void **SetCmdMax** (double _c)
Set the maximum value for the command.
- void **SetCmdMin** (double _c)
Set the maximum value for the command.
- void **SetDGain** (double _d)
Set the derivitive Gain.
- void **SetIGain** (double _i)
Set the integral Gain.
- void **SetIMax** (double _i)
Set the integral upper limit.
- void **SetIMin** (double _i)
Set the integral lower limit.
- void **SetPGain** (double _p)
Set the proportional Gain.
- double **Update** (double _error, **common::Time** _dt)
Update the Pid loop with nonuniform time step size.

10.160.1 Detailed Description

Generic **PID** (p. 782) controller class.

Generic proportional-integral-derivative controller class that keeps track of PID-error states and control inputs given the state of a system and a user specified target state.

10.160.2 Constructor & Destructor Documentation

10.160.2.1 `gazebo::common::PID::PID (double _p = 0.0, double _i = 0.0, double _d = 0.0, double _imax = 0.0, double _imin = 0.0, double _cmdMax = 0.0, double _cmdMin = 0.0)`

Constructor, zeros out Pid values when created and initialize Pid-gains and integral term limits:[iMax:iMin]-[I1:I2].

Parameters

in	<code>_p</code>	The proportional gain.
in	<code>_i</code>	The integral gain.
in	<code>_d</code>	The derivative gain.
in	<code>_imax</code>	The integral upper limit.
in	<code>_imin</code>	The integral lower limit.
in	<code>_cmdMax</code>	Output max value.
in	<code>_cmdMin</code>	Output min value.

10.160.2.2 `virtual gazebo::common::PID::~~PID () [virtual]`

Destructor.

10.160.3 Member Function Documentation

10.160.3.1 `double gazebo::common::PID::GetCmd ()`

Return current command for this **PID** (p. 782) controller.

Returns

the command value

10.160.3.2 `double gazebo::common::PID::GetCmdMax () const`

Get the maximum value for the command.

Returns

The maximum value

10.160.3.3 `double gazebo::common::PID::GetCmdMin () const`

Get the maximum value for the command.

Returns

The maximum value

10.160.3.4 `double gazebo::common::PID::GetDGain () const`

Get the derivative Gain.

Returns

The derivative gain value

10.160.3.5 `void gazebo::common::PID::GetErrors (double & _pe, double & _ie, double & _de)`

Return **PID** (p. 782) error terms for the controller.

Parameters

<code>in</code>	<code>_pe</code>	The proportional error.
<code>in</code>	<code>_ie</code>	The integral error.
<code>in</code>	<code>_de</code>	The derivative error.

10.160.3.6 `double gazebo::common::PID::GetIGain () const`

Get the integral Gain.

Returns

The integral gain value

10.160.3.7 `double gazebo::common::PID::GetIMax () const`

Get the integral upper limit.

Returns

The integral upper limit value

10.160.3.8 `double gazebo::common::PID::GetIMin () const`

Get the integral lower limit.

Returns

The integral lower limit value

10.160.3.9 `double gazebo::common::PID::GetPGain () const`

Get the proportional Gain.

Returns

The proportional gain value

10.160.3.10 `void gazebo::common::PID::Init (double _p = 0.0, double _i = 0.0, double _d = 0.0, double _imax = 0.0, double _imin = 0.0, double _cmdMax = 0.0, double _cmdMin = 0.0)`

Initialize PID-gains and integral term limits:[iMax:iMin]-[1:12].

Parameters

in	<code>_p</code>	The proportional gain.
in	<code>_i</code>	The integral gain.
in	<code>_d</code>	The derivative gain.
in	<code>_imax</code>	The integral upper limit.
in	<code>_imin</code>	The integral lower limit.
in	<code>_cmdMax</code>	Output max value.
in	<code>_cmdMin</code>	Output min value.

10.160.3.11 `PID& gazebo::common::PID::operator= (const PID & _p) [inline]`

Assignment operator.

Parameters

in	<code>_p</code>	a reference to a PID (p. 782) to assign values from
----	-----------------	--

Returns

reference to this instance

10.160.3.12 `void gazebo::common::PID::Reset ()`

Reset the errors and command.

10.160.3.13 `void gazebo::common::PID::SetCmd (double _cmd)`

Set current target command for this **PID** (p. 782) controller.

Parameters

in	<code>_cmd</code>	New command
----	-------------------	-------------

10.160.3.14 void gazebo::common::PID::SetCmdMax (double *_c*)

Set the maximum value for the command.

Parameters

in	<i>_c</i>	The maximum value
----	-----------	-------------------

10.160.3.15 void gazebo::common::PID::SetCmdMin (double *_c*)

Set the maximum value for the command.

Parameters

in	<i>_c</i>	The maximum value
----	-----------	-------------------

10.160.3.16 void gazebo::common::PID::SetDGain (double *_d*)

Set the derivative Gain.

Parameters

in	<i>_p</i>	derivative gain value
----	-----------	-----------------------

10.160.3.17 void gazebo::common::PID::SetIGain (double *_i*)

Set the integral Gain.

Parameters

in	<i>_p</i>	integral gain value
----	-----------	---------------------

10.160.3.18 void gazebo::common::PID::SetIMax (double *_i*)

Set the integral upper limit.

Parameters

in	<i>_p</i>	integral upper limit value
----	-----------	----------------------------

10.160.3.19 void gazebo::common::PID::SetIMin (double *_i*)

Set the integral lower limit.

Parameters

in	<i>_p</i>	integral lower limit value
----	-----------	----------------------------

10.160.3.20 void gazebo::common::PID::SetPGain (double *_p*)

Set the proportional Gain.

Parameters

<i>in</i>	<i>_p</i>	proportional gain value
-----------	-----------	-------------------------

10.160.3.21 double gazebo::common::PID::Update (double *_error*, common::Time *_dt*)

Update the Pid loop with nonuniform time step size.

Parameters

<i>_in]</i>	<i>_error</i>	Error since last call (<i>p_state</i> - <i>p_target</i>).
<i>_in]</i>	<i>_dt</i>	Change in time since last update call. Normally, this is called at every time step, The return value is an updated command to be passed to the object being controlled.

Returns

the command value

The documentation for this class was generated from the following file:

- **PID.hh**

10.161 gazebo::math::Plane Class Reference

A plane and related functions.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Plane** ()
Constructor.
- **Plane** (const **Vector3** &*_normal*, double *_offset=0.0*)
Constructor from a normal and a distanec.
- **Plane** (const **Vector3** &*_normal*, const **Vector2d** &*_size*, double *_offset*)
Constructor.
- virtual ~**Plane** ()
Destructor.
- double **Distance** (const **Vector3** &*_origin*, const **Vector3** &*_dir*) const
Get distance to the plane give an origin and direction.
- **Plane** & **operator=** (const **Plane** &*_p*)
Equal operator.
- void **Set** (const **Vector3** &*_normal*, const **Vector2d** &*_size*, double *offset*)
Set the plane.

Public Attributes

- double **d**
Plane (p. 788) *offset*.
- **Vector3 normal**
Plane (p. 788) *normal*.
- **Vector2d size**
Plane (p. 788) *size*.

10.161.1 Detailed Description

A plane and related functions.

10.161.2 Constructor & Destructor Documentation

10.161.2.1 gazebo::math::Plane::Plane ()

Constructor.

10.161.2.2 gazebo::math::Plane::Plane (const Vector3 & *_normal*, double *_offset* = 0.0)

Constructor from a normal and a distanec.

Parameters

in	<i>_normal</i>	The plane normal
in	<i>_offset</i>	Offset along the normal

10.161.2.3 gazebo::math::Plane::Plane (const Vector3 & *_normal*, const Vector2d & *_size*, double *_offset*)

Constructor.

Parameters

in	<i>_normal</i>	The plane normal
in	<i>_size</i>	Size of the plane
in	<i>_offset</i>	Offset along the normal

10.161.2.4 virtual gazebo::math::Plane::~~Plane () [virtual]

Destructor.

10.161.3 Member Function Documentation

10.161.3.1 double gazebo::math::Plane::Distance (const Vector3 & *_origin*, const Vector3 & *_dir*) const

Get distance to the plane give an origin and direction.

Parameters

in	<i>_origin</i>	the origin
in	<i>_dir</i>	a direction

Returns

the shortest distance

10.161.3.2 `Plane& gazebo::math::Plane::operator=(const Plane & _p)`

Equal operator.

Parameters

	<i>_p</i>	another plane
--	-----------	---------------

Returns

itself

10.161.3.3 `void gazebo::math::Plane::Set (const Vector3 & _normal, const Vector2d & _size, double offset)`

Set the plane.

Parameters

in	<i>_normal</i>	The plane normal
in	<i>_size</i>	Size of the plane
in	<i>_offset</i>	Offset along the normal

10.161.4 Member Data Documentation

10.161.4.1 `double gazebo::math::Plane::d`

Plane (p. 788) offset.

10.161.4.2 `Vector3 gazebo::math::Plane::normal`

Plane (p. 788) normal.

10.161.4.3 `Vector2d gazebo::math::Plane::size`

Plane (p. 788) size.

The documentation for this class was generated from the following file:

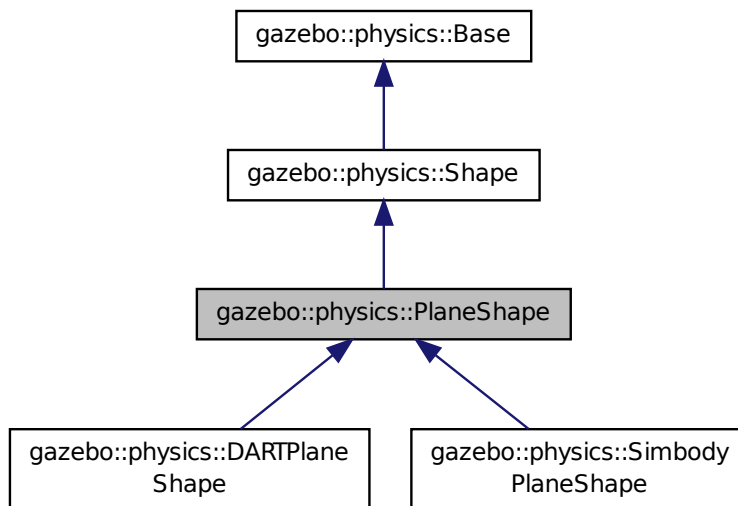
- **Plane.hh**

10.162 gazebo::physics::PlaneShape Class Reference

Collision (p. 235) for an infinite plane.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::PlaneShape:



Public Member Functions

- **PlaneShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~PlaneShape** ()
Destructor.
- virtual void **CreatePlane** ()
Create the plane.
- void **FillMsg** (msgs::Geometry &_msg)
Fill a geometry message with data from this object.
- **math::Vector3 GetNormal** () const
Get the plane normal.
- **math::Vector2d GetSize** () const
Get the size.
- virtual void **Init** ()
Initialize the plane.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
Process a geometry message and use the data to update this object.
- virtual void **SetAltitude** (const **math::Vector3** &_pos)
Set the altitude of the plane.

- void **SetNormal** (const **math::Vector3** &_norm)
Set the normal.
- virtual void **SetScale** (const **math::Vector3** &_scale)
Set the scale of the plane.
- void **SetSize** (const **math::Vector2d** &_size)
Set the size.

Additional Inherited Members

10.162.1 Detailed Description

Collision (p. 235) for an infinite plane.

This collision is used primarily for ground planes. Note that while the plane is infinite, only the part near the camera is drawn.

10.162.2 Constructor & Destructor Documentation

10.162.2.1 `gazebo::physics::PlaneShape::PlaneShape (CollisionPtr _parent) [explicit]`

Constructor.

Parameters

<code>in</code>	<code>_parent</code>	Link (p. 595) to which we are attached.
-----------------	----------------------	--

10.162.2.2 `virtual gazebo::physics::PlaneShape::~~PlaneShape () [virtual]`

Destructor.

10.162.3 Member Function Documentation

10.162.3.1 `virtual void gazebo::physics::PlaneShape::CreatePlane () [virtual]`

Create the plane.

Reimplemented in **gazebo::physics::SimbodyPlaneShape** (p. 999), and **gazebo::physics::DARTPlaneShape** (p. 362).

Referenced by `gazebo::physics::DARTPlaneShape::CreatePlane()`.

10.162.3.2 `void gazebo::physics::PlaneShape::FillMsg (msgs::Geometry & _msg) [virtual]`

Fill a geometry message with data from this object.

Parameters

<code>out</code>	<code>_msg</code>	Message to fill.
------------------	-------------------	------------------

Implements **gazebo::physics::Shape** (p. 934).

10.162.3.3 `math::Vector3 gazebo::physics::PlaneShape::GetNormal () const`

Get the plane normal.

Returns

The plane normal.

10.162.3.4 `math::Vector2d gazebo::physics::PlaneShape::GetSize () const`

Get the size.

Returns

Size of the plane.

10.162.3.5 `virtual void gazebo::physics::PlaneShape::Init () [virtual]`

Initialize the plane.

Implements **gazebo::physics::Shape** (p. 935).

10.162.3.6 `virtual void gazebo::physics::PlaneShape::ProcessMsg (const msgs::Geometry & .msg) [virtual]`

Process a geometry message and use the data to update this object.

Parameters

in	_msg	Message to update from.
----	------	-------------------------

Implements **gazebo::physics::Shape** (p. 935).

10.162.3.7 `virtual void gazebo::physics::PlaneShape::SetAltitude (const math::Vector3 & .pos) [virtual]`

Set the altitude of the plane.

Parameters

in	_pos	Position of the plane.
----	------	------------------------

Reimplemented in **gazebo::physics::DARTPlaneShape** (p. 362), and **gazebo::physics::SimbodyPlaneShape** (p. 999).

Referenced by `gazebo::physics::DARTPlaneShape::SetAltitude()`.

10.162.3.8 `void gazebo::physics::PlaneShape::SetNormal (const math::Vector3 & .norm)`

Set the normal.

Parameters

in	_norm	Plane normal.
----	-------	---------------

10.162.3.9 virtual void gazebo::physics::PlaneShape::SetScale (const math::Vector3 & _scale) [virtual]

Set the scale of the plane.

Returns

_scale Scale to set the plane to.

Implements **gazebo::physics::Shape** (p. 935).

10.162.3.10 void gazebo::physics::PlaneShape::SetSize (const math::Vector2d & _size)

Set the size.

Parameters

in	_size	2D size of the plane.
----	-------	-----------------------

The documentation for this class was generated from the following file:

- **PlaneShape.hh**

10.163 gazebo::PluginT< T > Class Template Reference

A class which all plugins must inherit from.

```
#include <common/common.hh>
```

Classes

- union **fptr_union_t**
Pointer to shared library registration function definition.

Public Types

- typedef boost::shared_ptr< T > **TPtr**
plugin pointer type definition

Public Member Functions

- **PluginT** ()
Constructor.
- virtual **~PluginT** ()
Destructor.

- `std::string GetFilename ()` const
Get the name of the handler.
- `std::string GetHandle ()` const
Get the short name of the handler.
- `PluginType GetType ()` const
Returns the type of the plugin.

Static Public Member Functions

- static `TPtr Create` (const `std::string &_filename`, const `std::string &_handle`)
a class method that creates a plugin from a file name.

Protected Attributes

- `std::string filename`
Path to the shared library file.
- `std::string handle`
Short name.
- `PluginType type`
Type of plugin.

10.163.1 Detailed Description

`template<class T>class gazebo::PluginT< T >`

A class which all plugins must inherit from.

10.163.2 Member Typedef Documentation

10.163.2.1 `template<class T> typedef boost::shared_ptr<T> gazebo::PluginT< T >::TPtr`

plugin pointer type definition

10.163.3 Constructor & Destructor Documentation

10.163.3.1 `template<class T> gazebo::PluginT< T >::PluginT ()` [`inline`]

Constructor.

10.163.3.2 `template<class T> virtual gazebo::PluginT< T >::~~PluginT ()` [`inline`],[`virtual`]

Destructor.

10.163.4 Member Function Documentation

10.163.4.1 `template<class T> static TPtr gazebo::PluginT< T >::Create (const std::string & _filename, const std::string & _handle) [inline],[static]`

a class method that creates a plugin from a file name.

It locates the shared library and loads it dynamically.

Parameters

in	<code>_filename</code>	the path to the shared library.
in	<code>_handle</code>	short name of the handler

Returns

Shared Pointer to this class type

10.163.4.2 `template<class T> std::string gazebo::PluginT< T >::GetFilename () const [inline]`

Get the name of the handler.

10.163.4.3 `template<class T> std::string gazebo::PluginT< T >::GetHandle () const [inline]`

Get the short name of the handler.

10.163.4.4 `template<class T> PluginType gazebo::PluginT< T >::GetType () const [inline]`

Returns the type of the plugin.

Returns

type of the plugin

10.163.5 Member Data Documentation

10.163.5.1 `template<class T> std::string gazebo::PluginT< T >::filename [protected]`

Path to the shared library file.

10.163.5.2 `template<class T> std::string gazebo::PluginT< T >::handle [protected]`

Short name.

10.163.5.3 `template<class T> PluginType gazebo::PluginT< T >::type [protected]`

Type of plugin.

The documentation for this class was generated from the following file:

- **Plugin.hh**

10.164 gazebo::math::Pose Class Reference

Encapsulates a position and rotation in three space.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Pose** ()
Default constructors.
- **Pose** (const **Vector3** &_pos, const **Quaternion** &_rot)
Constructor.
- **Pose** (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)
Constructor.
- **Pose** (const **Pose** &_pose)
Copy constructor.
- virtual ~**Pose** ()
Destructor.
- **Pose CoordPoseSolve** (const **Pose** &_b) const
Find the inverse of a pose; i.e., if $b = this + a$, given b and $this$, find a .
- **Vector3 CoordPositionAdd** (const **Vector3** &_pos) const
Add one point to a vector: result = this + pos.
- **Vector3 CoordPositionAdd** (const **Pose** &_pose) const
Add one point to another: result = this + pose.
- **Vector3 CoordPositionSub** (const **Pose** &_pose) const
Subtract one position from another: result = this - pose.
- **Quaternion CoordRotationAdd** (const **Quaternion** &_rot) const
Add one rotation to another: result = this->rot + rot.
- **Quaternion CoordRotationSub** (const **Quaternion** &_rot) const
Subtract one rotation from another: result = this->rot - rot.
- void **Correct** ()
Fix any nan values.
- **Pose GetInverse** () const
Get the inverse of this pose.
- bool **IsFinite** () const
See if a pose is finite (e.g., not nan)
- bool **operator!=** (const **Pose** &_pose) const
Inequality operator.
- **Pose operator*** (const **Pose** &_pose)
Multiplication operator.
- **Pose operator+** (const **Pose** &_pose) const
Addition operator A is the transform from O to P specified in frame O B is the transform from P to Q specified in frame P then, $B + A$ is the transform from O to Q specified in frame O .
- const **Pose** & **operator+=** (const **Pose** &_pose)
Add-Equals operator.
- **Pose operator-** () const
Negation operator A is the transform from O to P in frame O then $-A$ is transform from P to O specified in frame P .
- **Pose operator-** (const **Pose** &_pose) const

Subtraction operator A is the transform from O to P in frame O B is the transform from O to Q in frame O B - A is the transform from P to Q in frame P.

- **const Pose & operator-=** (const **Pose** &_pose)

Subtraction operator.

- **Pose & operator=** (const **Pose** &_pose)

Equal operator.

- **bool operator==** (const **Pose** &_pose) const

Equality operator.

- **void Reset** ()

Reset the pose.

- **Pose RotatePositionAboutOrigin** (const **Quaternion** &_rot) const

Rotate vector part of a pose about the origin.

- **void Round** (int _precision)

Round all values to _precision decimal places.

- **void Set** (const **Vector3** &_pos, const **Quaternion** &_rot)

*Set the pose from a **Vector3** (p. 1165) and a **Quaternion** (p. 824).*

- **void Set** (const **Vector3** &_pos, const **Vector3** &_rpy)

Set the pose from pos and rpy vectors.

- **void Set** (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)

Set the pose from a six tuple.

Public Attributes

- **Vector3 pos**

The position.

- **Quaternion rot**

The rotation.

Static Public Attributes

- static const **Pose Zero**

math::Pose(0, 0, 0, 0, 0, 0)

Friends

- **std::ostream & operator<<** (std::ostream &_out, const **gazebo::math::Pose** &_pose)

Stream insertion operator.

- **std::istream & operator>>** (std::istream &_in, **gazebo::math::Pose** &_pose)

Stream extraction operator.

10.164.1 Detailed Description

Encapsulates a position and rotation in three space.

10.164.2 Constructor & Destructor Documentation

10.164.2.1 gazebo::math::Pose::Pose ()

Default constructors.

10.164.2.2 gazebo::math::Pose::Pose (const Vector3 & *_pos*, const Quaternion & *_rot*)

Constructor.

Parameters

in	<i>_pos</i>	A position
in	<i>_rot</i>	A rotation

10.164.2.3 gazebo::math::Pose::Pose (double *_x*, double *_y*, double *_z*, double *_roll*, double *_pitch*, double *_yaw*)

Constructor.

Parameters

in	<i>_x</i>	x position in meters.
in	<i>_y</i>	y position in meters.
in	<i>_z</i>	z position in meters.
in	<i>_roll</i>	Roll (rotation about X-axis) in radians.
in	<i>_pitch</i>	Pitch (rotation about y-axis) in radians.
in	<i>_yaw</i>	Yaw (rotation about z-axis) in radians.

10.164.2.4 gazebo::math::Pose::Pose (const Pose & *_pose*)

Copy constructor.

Parameters

in	<i>_pose</i>	Pose (p. 797) to copy
----	--------------	------------------------------

10.164.2.5 virtual gazebo::math::Pose::~~Pose () [virtual]

Destructor.

10.164.3 Member Function Documentation

10.164.3.1 Pose gazebo::math::Pose::CoordPoseSolve (const Pose & *_b*) const

Find the inverse of a pose; i.e., if $b = \text{this} + a$, given b and this , find a .

Parameters

in	<i>_b</i>	the other pose
----	-----------	----------------

10.164.3.2 Vector3 gazebo::math::Pose::CoordPositionAdd (const Vector3 & *_pos*) const

Add one point to a vector: result = this + pos.

Parameters

<i>in</i>	<i>_pos</i>	Position to add to this pose
-----------	-------------	------------------------------

Returns

the resulting position

10.164.3.3 Vector3 gazebo::math::Pose::CoordPositionAdd (const Pose & *_pose*) const

Add one point to another: result = this + pose.

Parameters

<i>in</i>	<i>_pose</i>	The Pose (p. 797) to add
-----------	--------------	---------------------------------

Returns

The resulting position

10.164.3.4 Vector3 gazebo::math::Pose::CoordPositionSub (const Pose & *_pose*) const [inline]

Subtract one position from another: result = this - pose.

Parameters

<i>in</i>	<i>_pose</i>	Pose (p. 797) to subtract
-----------	--------------	----------------------------------

Returns

The resulting position

References gazebo::math::Quaternion::GetInverse(), pos, rot, gazebo::math::Vector3::x, gazebo::math::Quaternion::x, gazebo::math::Vector3::y, gazebo::math::Quaternion::y, gazebo::math::Vector3::z, and gazebo::math::Quaternion::z.

10.164.3.5 Quaternion gazebo::math::Pose::CoordRotationAdd (const Quaternion & *_rot*) const

Add one rotation to another: result = this->rot + rot.

Parameters

<i>in</i>	<i>_rot</i>	Rotation to add
-----------	-------------	-----------------

Returns

The resulting rotation

10.164.3.6 **Quaternion** gazebo::math::Pose::CoordRotationSub (const Quaternion & *_rot*) const `[inline]`

Subtract one rotation from another: result = this->rot - rot.

Parameters

<i>in</i>	<i>_rot</i>	The rotation to subtract
-----------	-------------	--------------------------

Returns

The resulting rotation

References gazebo::math::Quaternion::GetInverse(), and gazebo::math::Quaternion::Normalize().

10.164.3.7 **void** gazebo::math::Pose::Correct () `[inline]`

Fix any nan values.

10.164.3.8 **Pose** gazebo::math::Pose::GetInverse () const

Get the inverse of this pose.

Returns

the inverse pose

10.164.3.9 **bool** gazebo::math::Pose::IsFinite () const

See if a pose is finite (e.g., not nan)

10.164.3.10 **bool** gazebo::math::Pose::operator!= (const Pose & *_pose*) const

Inequality operator.

Parameters

<i>in</i>	<i>_pose</i>	Pose (p. 797) for comparison
-----------	--------------	-------------------------------------

Returns

True if not equal

10.164.3.11 **Pose** gazebo::math::Pose::operator* (const Pose & *_pose*)

Multiplication operator.

Parameters

<i>in</i>	<i>_pose</i>	the other pose
-----------	--------------	----------------

Returns

itself

10.164.3.12 Pose gazebo::math::Pose::operator+ (const Pose & *_pose*) const

Addition operator A is the transform from O to P specified in frame O B is the transform from P to Q specified in frame P then, B + A is the transform from O to Q specified in frame O.

Parameters

in	<i>_pose</i>	Pose (p. 797) to add to this pose
----	--------------	--

Returns

The resulting pose

10.164.3.13 const Pose& gazebo::math::Pose::operator+=(const Pose & *_pose*)

Add-Equals operator.

Parameters

in	<i>_pose</i>	Pose (p. 797) to add to this pose
----	--------------	--

Returns

The resulting pose

10.164.3.14 Pose gazebo::math::Pose::operator- () const [inline]

Negation operator A is the transform from O to P in frame O then -A is transform from P to O specified in frame P.

Returns

The resulting pose

10.164.3.15 Pose gazebo::math::Pose::operator- (const Pose & *_pose*) const [inline]

Subtraction operator A is the transform from O to P in frame O B is the transform from O to Q in frame O B - A is the transform from P to Q in frame P.

Parameters

in	<i>_pose</i>	Pose (p. 797) to subtract from this one
----	--------------	--

Returns

The resulting pose

References rot.

10.164.3.16 `const Pose& gazebo::math::Pose::operator-= (const Pose & _pose)`

Subtraction operator.

Parameters

<code>in</code>	<code>_pose</code>	Pose (p. 797) to subtract from this one
-----------------	--------------------	--

Returns

The resulting pose

10.164.3.17 `Pose& gazebo::math::Pose::operator= (const Pose & _pose)`

Equal operator.

Parameters

<code>in</code>	<code>_pose</code>	Pose (p. 797) to copy
-----------------	--------------------	------------------------------

10.164.3.18 `bool gazebo::math::Pose::operator== (const Pose & _pose) const`

Equality operator.

Parameters

<code>in</code>	<code>_pose</code>	Pose (p. 797) for comparison
-----------------	--------------------	-------------------------------------

Returns

True if equal

10.164.3.19 `void gazebo::math::Pose::Reset ()`

Reset the pose.

10.164.3.20 `Pose gazebo::math::Pose::RotatePositionAboutOrigin (const Quaternion & _rot) const`

Rotate vector part of a pose about the origin.

Parameters

<code>in</code>	<code>_rot</code>	rotation
-----------------	-------------------	----------

Returns

the rotated pose

10.164.3.21 `void gazebo::math::Pose::Round (int _precision)`

Round all values to `_precision` decimal places.

Parameters

<code>in</code>	<code><i>_precision</i></code>	
-----------------	--------------------------------	--

10.164.3.22 `void gazebo::math::Pose::Set (const Vector3 & _pos, const Quaternion & _rot)`

Set the pose from a **Vector3** (p. 1165) and a **Quaternion** (p. 824).

Parameters

<code>in</code>	<code><i>_pos</i></code>	The position.
<code>in</code>	<code><i>_rot</i></code>	The rotation.

10.164.3.23 `void gazebo::math::Pose::Set (const Vector3 & _pos, const Vector3 & _rpy)`

Set the pose from `pos` and `rpy` vectors.

Parameters

<code>in</code>	<code><i>_pos</i></code>	The position.
<code>in</code>	<code><i>_rpy</i></code>	The rotation expressed as Euler angles.

10.164.3.24 `void gazebo::math::Pose::Set (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)`

Set the pose from a six tuple.

Parameters

<code>in</code>	<code><i>_x</i></code>	x position in meters.
<code>in</code>	<code><i>_y</i></code>	y position in meters.
<code>in</code>	<code><i>_z</i></code>	z position in meters.
<code>in</code>	<code><i>_roll</i></code>	Roll (rotation about X-axis) in radians.
<code>in</code>	<code><i>_pitch</i></code>	Pitch (rotation about y-axis) in radians.
<code>in</code>	<code><i>_yaw</i></code>	Pitch (rotation about z-axis) in radians.

10.164.4 Friends And Related Function Documentation

10.164.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Pose & _pose)` [`friend`]

Stream insertion operator.

Parameters

<code>in</code>	<code>_out</code>	output stream
<code>in</code>	<code>_pose</code>	pose to output

Returns

the stream

10.164.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Pose & _pose) [friend]`

Stream extraction operator.

Parameters

<code>in</code>	<code>_in</code>	the input stream
<code>in</code>	<code>_pose</code>	the pose

Returns

the stream

10.164.5 Member Data Documentation

10.164.5.1 Vector3 gazebo::math::Pose::pos

The position.

Referenced by gazebo::physics::DARTTypes::ConvPose(), and CoordPositionSub().

10.164.5.2 Quaternion gazebo::math::Pose::rot

The rotation.

Referenced by gazebo::physics::DARTTypes::ConvPose(), CoordPositionSub(), and operator-().

10.164.5.3 `const Pose gazebo::math::Pose::Zero [static]`

`math::Pose(0, 0, 0, 0, 0, 0)`

The documentation for this class was generated from the following file:

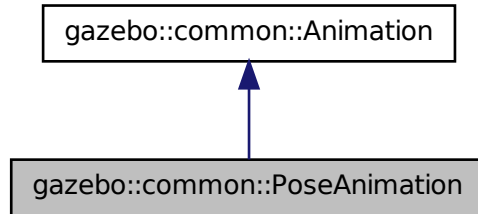
- **Pose.hh**

10.165 gazebo::common::PoseAnimation Class Reference

A pose animation.

```
#include <Animation.hh>
```

Inheritance diagram for gazebo::common::PoseAnimation:



Public Member Functions

- **PoseAnimation** (const std::string &_name, double _length, bool _loop)
Constructor.
- virtual **~PoseAnimation** ()
Destructor.
- **PoseKeyFrame * CreateKeyFrame** (double _time)
Create a pose keyframe at the given time.
- void **GetInterpolatedKeyFrame** (**PoseKeyFrame** &_kf) const
Get a keyframe using the animation's current time.

Protected Member Functions

- void **BuildInterpolationSplines** () const
Update the pose splines.
- void **GetInterpolatedKeyFrame** (double _time, **PoseKeyFrame** &_kf) const
Get a keyframe using a passed in time.

Additional Inherited Members

10.165.1 Detailed Description

A pose animation.

10.165.2 Constructor & Destructor Documentation

10.165.2.1 gazebo::common::PoseAnimation::PoseAnimation (const std::string & _name, double _length, bool _loop)

Constructor.

Parameters

in	<i>_name</i>	String name of the animation. This should be unique.
in	<i>_length</i>	Length of the animation in seconds
in	<i>_loop</i>	True == loop the animation

10.165.2.2 virtual gazebo::common::PoseAnimation::~~PoseAnimation () [virtual]

Destructor.

10.165.3 Member Function Documentation

10.165.3.1 void gazebo::common::PoseAnimation::BuildInterpolationSplines () const [protected]

Update the pose splines.

10.165.3.2 PoseKeyFrame* gazebo::common::PoseAnimation::CreateKeyFrame (double *_time*)

Create a pose keyframe at the given time.

Parameters

in	<i>_time</i>	Time (p. 1099) at which to create the keyframe
----	--------------	---

Returns

Pointer to the new keyframe

10.165.3.3 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (PoseKeyFrame & *_kf*) const

Get a keyframe using the animation's current time.

Parameters

out	<i>_kf</i>	PoseKeyFrame (p. 808) reference to hold the interpolated result
-----	------------	--

10.165.3.4 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (double *_time*, PoseKeyFrame & *_kf*) const [protected]

Get a keyframe using a passed in time.

Parameters

in	<i>_time</i>	Time (p. 1099) in seconds
out	<i>_kf</i>	PoseKeyFrame (p. 808) reference to hold the interpolated result

The documentation for this class was generated from the following file:

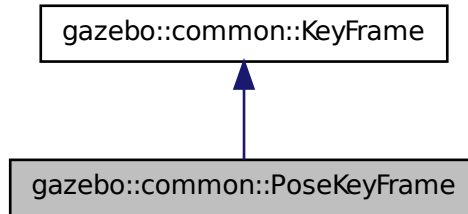
- **Animation.hh**

10.166 gazebo::common::PoseKeyFrame Class Reference

A keyframe for a **PoseAnimation** (p. 805).

```
#include <KeyFrame.hh>
```

Inheritance diagram for gazebo::common::PoseKeyFrame:



Public Member Functions

- **PoseKeyFrame** (double _time)
Constructor.
- virtual **~PoseKeyFrame** ()
Destructor.
- const **math::Quaternion** & **GetRotation** () const
Get the rotation of the keyframe.
- const **math::Vector3** & **GetTranslation** () const
Get the translation of the keyframe.
- void **SetRotation** (const **math::Quaternion** &_rot)
Set the rotation for the keyframe.
- void **SetTranslation** (const **math::Vector3** &_trans)
Set the translation for the keyframe.

Protected Attributes

- **math::Quaternion** rotate
the rotation quaternion
- **math::Vector3** translate
the translation vector

10.166.1 Detailed Description

A keyframe for a **PoseAnimation** (p. 805).

10.166.2 Constructor & Destructor Documentation

10.166.2.1 gazebo::common::PoseKeyFrame::PoseKeyFrame (double *_time*)

Constructor.

Parameters

in	<i>_time</i>	of the keyframe
----	--------------	-----------------

10.166.2.2 virtual gazebo::common::PoseKeyFrame::~~PoseKeyFrame () [virtual]

Destructor.

10.166.3 Member Function Documentation

10.166.3.1 const math::Quaternion& gazebo::common::PoseKeyFrame::GetRotation () const

Get the rotation of the keyframe.

Returns

The rotation amount

10.166.3.2 const math::Vector3& gazebo::common::PoseKeyFrame::GetTranslation () const

Get the translation of the keyframe.

Returns

The translation amount

10.166.3.3 void gazebo::common::PoseKeyFrame::SetRotation (const math::Quaternion & *_rot*)

Set the rotation for the keyframe.

Parameters

in	<i>_rot</i>	Rotation amount
----	-------------	-----------------

10.166.3.4 void gazebo::common::PoseKeyFrame::SetTranslation (const math::Vector3 & *_trans*)

Set the translation for the keyframe.

Parameters

in	<i>_trans</i>	Translation amount
----	---------------	--------------------

10.166.4 Member Data Documentation

10.166.4.1 `math::Quaternion gazebo::common::PoseKeyFrame::rotate` [protected]

the rotation quaternion

10.166.4.2 `math::Vector3 gazebo::common::PoseKeyFrame::translate` [protected]

the translation vector

The documentation for this class was generated from the following file:

- `KeyFrame.hh`

10.167 gazebo::rendering::Projector Class Reference

Projects a material onto surface, light a light projector.

```
#include <rendering/rendering.hh>
```

Classes

- class `ProjectorFrameListener`

Public Member Functions

- **Projector** (`VisualPtr _parent`)
Constructor.
- virtual `~Projector` ()
Destructor.
- **VisualPtr GetParent** ()
Get the parent visual.
- void **Load** (`sdf::ElementPtr _sdf`)
Load from an sdf pointer.
- void **Load** (`const msgs::Projector &_msg`)
Load from a message.
- void **Load** (`const std::string &_name, const math::Pose &_pose=math::Pose(0, 0, 0, 0, 0, 0), const std::string &_textureName="", double _nearClip=0.25, double _farClip=15.0, double _fov=M_PI *0.25`)
Load the projector.
- void **SetEnabled** (`bool _enabled`)
Set whether the projector is enabled or disabled.
- void **SetTexture** (`const std::string &_textureName`)
Load a texture into the projector.
- void **Toggle** ()
Toggle the activation of the projector.

10.167.1 Detailed Description

Projects a material onto surface, light a light projector.

10.167.2 Constructor & Destructor Documentation

10.167.2.1 gazebo::rendering::Projector::Projector (VisualPtr *_parent*)

Constructor.

Parameters

<i>in</i>	<i>_parent</i>	Name of the parent visual.
-----------	----------------	----------------------------

10.167.2.2 virtual gazebo::rendering::Projector::~~Projector () [virtual]

Destructor.

10.167.3 Member Function Documentation

10.167.3.1 VisualPtr gazebo::rendering::Projector::GetParent ()

Get the parent visual.

Returns

Pointer to the parent visual.

10.167.3.2 void gazebo::rendering::Projector::Load (sdf::ElementPtr *_sdf*)

Load from an sdf pointer.

Parameters

<i>in</i>	<i>_sdf</i>	Pointer to the SDF element.
-----------	-------------	-----------------------------

10.167.3.3 void gazebo::rendering::Projector::Load (const msgs::Projector & *_msg*)

Load from a message.

Parameters

<i>in</i>	<i>_msg</i>	Load from a message.
-----------	-------------	----------------------

```
10.167.3.4 void gazebo::rendering::Projector::Load ( const std::string & _name, const math::Pose & _pose =
            math::Pose (0, 0, 0, 0, 0, 0), const std::string & _textureName = "", double _nearClip = 0.25, double
            _farClip = 15.0, double _fov = M_PI *0.25 )
```

Load the projector.

Parameters

in	<code>_name</code>	Name of the projector.
in	<code>_pos</code>	Pose of the projector.
in	<code>_textureName</code>	Name of the texture to project.
in	<code>_nearClip</code>	Near clip distance.
in	<code>_farClip</code>	Far clip distance.
in	<code>_fov</code>	Field of view.

```
10.167.3.5 void gazebo::rendering::Projector::SetEnabled ( bool _enabled )
```

Set whether the projector is enabled or disabled.

Parameters

in	<code>_enabled</code>	True to enable the projector.
----	-----------------------	-------------------------------

```
10.167.3.6 void gazebo::rendering::Projector::SetTexture ( const std::string & _textureName )
```

Load a texture into the projector.

Parameters

in	<code>_textureName</code>	Name of the texture to project.
----	---------------------------	---------------------------------

```
10.167.3.7 void gazebo::rendering::Projector::Toggle ( )
```

Toggle the activation of the projector.

The documentation for this class was generated from the following file:

- **Projector.hh**

10.168 gazebo::transport::Publication Class Reference

A publication for a topic.

```
#include <transport/transport.hh>
```

Public Member Functions

- **Publication** (const std::string &_topic, const std::string &_msgType)

Constructor.

- virtual `~Publication ()`
Destructor.
- void **AddPublisher** (`PublisherPtr _pub`)
Add a publisher.
- void **AddSubscription** (`const CallbackHelperPtr _callback`)
Subscribe a callback to our topic.
- void **AddSubscription** (`const NodePtr &_node`)
Subscribe a node to our topic.
- void **AddTransport** (`const PublicationTransportPtr &_publink`)
Add a transport.
- unsigned int **GetCallbackCount** () const
Get the number of callbacks.
- bool **GetLocallyAdvertised** () const
Was the topic has been advertised from this process?
- std::string **GetMsgType** () const
Get the type of message.
- unsigned int **GetNodeCount** () const
Get the number of nodes.
- **MessagePtr GetPrevMsg** (`uint32_t _publd`)
Get a previous message for a publisher.
- unsigned int **GetRemoteSubscriptionCount** ()
Get the number of remote subscriptions.
- unsigned int **GetTransportCount** () const
Get the number of transports.
- bool **HasTransport** (`const std::string &_host, unsigned int _port`)
Does a given transport exist?
- void **LocalPublish** (`const std::string &_data`)
Publish data to local subscribers (skip serialization)
- int **Publish** (`MessagePtr _msg, boost::function< void(uint32_t)> _cb, uint32_t _id`)
Publish data to remote subscribers.
- void **RemovePublisher** (`PublisherPtr _pub`)
Remove a publisher.
- void **RemoveSubscription** (`const NodePtr &_node`)
Unsubscribe a node from our topic.
- void **RemoveSubscription** (`const std::string &_host, unsigned int _port`)
Unsubscribe a a node by host/port from our topic.
- void **RemoveTransport** (`const std::string &_host, unsigned int _port`)
Remove a transport.
- void **SetLocallyAdvertised** (`bool _value`)
Set whether this topic has been advertised from this process.
- void **SetPrevMsg** (`uint32_t _publd, MessagePtr _msg`)
Set the previous message for a publisher.

10.168.1 Detailed Description

A publication for a topic.

This facilitates transport of messages

10.168.2 Constructor & Destructor Documentation

10.168.2.1 gazebo::transport::Publication::Publication (const std::string & *_topic*, const std::string & *_msgType*)

Constructor.

Parameters

in	<i>_topic</i>	The topic we're publishing
in	<i>_msgType</i>	The type of the topic we're publishing

10.168.2.2 virtual gazebo::transport::Publication::~~Publication () [virtual]

Destructor.

10.168.3 Member Function Documentation

10.168.3.1 void gazebo::transport::Publication::AddPublisher (PublisherPtr *_pub*)

Add a publisher.

Parameters

in, out	<i>_pub</i>	Pointer to publisher object to be added
---------	-------------	---

10.168.3.2 void gazebo::transport::Publication::AddSubscription (const CallbackHelperPtr *_callback*)

Subscribe a callback to our topic.

Parameters

in	<i>_callback</i>	The callback
----	------------------	--------------

10.168.3.3 void gazebo::transport::Publication::AddSubscription (const NodePtr & *_node*)

Subscribe a node to our topic.

Parameters

in	<i>_node</i>	The node
----	--------------	----------

10.168.3.4 void gazebo::transport::Publication::AddTransport (const PublicationTransportPtr & *_publink*)

Add a transport.

Parameters

in	<i>_publink</i>	Pointer to publication transport object to be added
----	-----------------	---

10.168.3.5 `unsigned int gazebo::transport::Publication::GetCallbackCount () const`

Get the number of callbacks.

Returns

The number of callbacks

10.168.3.6 `bool gazebo::transport::Publication::GetLocallyAdvertised () const`

Was the topic has been advertised from this process?

Returns

true if the topic has been advertised from this process, false otherwise

10.168.3.7 `std::string gazebo::transport::Publication::GetMsgType () const`

Get the type of message.

Returns

The type of message

10.168.3.8 `unsigned int gazebo::transport::Publication::GetNodeCount () const`

Get the number of nodes.

Returns

The number of nodes

10.168.3.9 `MessagePtr gazebo::transport::Publication::GetPrevMsg (uint32_t _pubId)`

Get a previous message for a publisher.

Parameters

<code>in</code>	<code>_pubId</code>	ID of the publisher.
-----------------	---------------------	----------------------

Returns

Pointer to the previous message. NULL if there is no previous message.

10.168.3.10 `unsigned int gazebo::transport::Publication::GetRemoteSubscriptionCount ()`

Get the number of remote subscriptions.

Returns

The number of remote subscriptions

10.168.3.11 `unsigned int gazebo::transport::Publication::GetTransportCount () const`

Get the number of transports.

Returns

The number of transports

10.168.3.12 `bool gazebo::transport::Publication::HasTransport (const std::string & _host, unsigned int _port)`

Does a given transport exist?

Parameters

<code>in</code>	<code>_host</code>	Hostname of the transport
<code>in</code>	<code>_port</code>	Port of the transport

Returns

true if the transport exists, false otherwise

10.168.3.13 `void gazebo::transport::Publication::LocalPublish (const std::string & _data)`

Publish data to local subscribers (skip serialization)

Parameters

<code>in</code>	<code>_data</code>	The data to be published
-----------------	--------------------	--------------------------

10.168.3.14 `int gazebo::transport::Publication::Publish (MessagePtr _msg, boost::function< void(uint32_t)> _cb, uint32_t _id)`

Publish data to remote subscribers.

Parameters

<code>in</code>	<code>_msg</code>	Message to be published
<code>in</code>	<code>_cb</code>	Callback to be invoked after publishing is completed

Returns

Number of remote subscribers that will receive the message.

10.168.3.15 `void gazebo::transport::Publication::RemovePublisher (PublisherPtr _pub)`

Remove a publisher.

Parameters

in	<i>_pub</i>	Pointer to publisher object to remove.
----	-------------	--

10.168.3.16 void gazebo::transport::Publication::RemoveSubscription (const NodePtr & *_node*)

Unsubscribe a node from our topic.

Parameters

in	<i>_node</i>	The node
----	--------------	----------

10.168.3.17 void gazebo::transport::Publication::RemoveSubscription (const std::string & *_host*, unsigned int *_port*)

Unsubscribe a a node by host/port from our topic.

Parameters

in	<i>_host</i>	The node's hostname
in	<i>_port</i>	The node's port

10.168.3.18 void gazebo::transport::Publication::RemoveTransport (const std::string & *_host*, unsigned int *_port*)

Remove a transport.

Parameters

in	<i>_host</i>	The transport's hostname
in	<i>_port</i>	The transport's port

10.168.3.19 void gazebo::transport::Publication::SetLocallyAdvertised (bool *_value*)

Set whether this topic has been advertised from this process.

Parameters

in	<i>_value</i>	If true, the topic was locally advertise, otherwise it was not
----	---------------	--

10.168.3.20 void gazebo::transport::Publication::SetPrevMsg (uint32_t *_pubId*, MessagePtr *_msg*)

Set the previous message for a publisher.

Parameters

in	<i>_pubId</i>	ID of the publisher.
in	<i>_msg</i>	The previous message.

The documentation for this class was generated from the following file:

- **Publication.hh**

10.169 gazebo::transport::PublicationTransport Class Reference

transport/transport.hh

```
#include <PublicationTransport.hh>
```

Public Member Functions

- **PublicationTransport** (const std::string &_topic, const std::string &_msgType)
Constructor.
- virtual ~**PublicationTransport** ()
Destructor.
- void **AddCallback** (const boost::function< void(const std::string &)> &_cb)
Add a callback to the transport.
- void **Fini** ()
Finalize the transport.
- const **ConnectionPtr** **GetConnection** () const
Get the underlying connection.
- std::string **GetMsgType** () const
Get the topic type.
- std::string **GetTopic** () const
Get the topic name.
- void **Init** (const **ConnectionPtr** &_conn, bool _latched)
Initialize the transport.

10.169.1 Detailed Description

transport/transport.hh

Reads data from a remote advertiser, and passes the data along to local subscribers

10.169.2 Constructor & Destructor Documentation

10.169.2.1 gazebo::transport::PublicationTransport::PublicationTransport (const std::string & _topic, const std::string & _msgType)

Constructor.

Parameters

in	<i>_topic</i>	Topic that we're publishing
in	<i>_topic</i>	Type of the topic that we're publishing

10.169.2.2 virtual gazebo::transport::PublicationTransport::~~PublicationTransport () [virtual]

Destructor.

10.169.3 Member Function Documentation

10.169.3.1 void gazebo::transport::PublicationTransport::AddCallback (const boost::function< void(const std::string &)> & _cb)

Add a callback to the transport.

Parameters

in	<i>_cb</i>	The callback to be added
----	------------	--------------------------

10.169.3.2 void gazebo::transport::PublicationTransport::Fini ()

Finalize the transport.

10.169.3.3 const ConnectionPtr gazebo::transport::PublicationTransport::GetConnection () const

Get the underlying connection.

Returns

Pointer to the underlying connection

10.169.3.4 std::string gazebo::transport::PublicationTransport::GetMsgType () const

Get the topic type.

Returns

The topic type

10.169.3.5 std::string gazebo::transport::PublicationTransport::GetTopic () const

Get the topic name.

Returns

The topic name

10.169.3.6 void gazebo::transport::PublicationTransport::Init (const ConnectionPtr & _conn, bool _latched)

Initialize the transport.

Parameters

in	<i>_conn</i>	The underlying connection.
in	<i>_latched</i>	True to grab the last message sent on the topic.

The documentation for this class was generated from the following file:

- **PublicationTransport.hh**

10.170 gazebo::transport::Publisher Class Reference

A publisher of messages on a topic.

```
#include <transport/transport.hh>
```

Public Member Functions

- **Publisher** (const std::string &_topic, const std::string &_msgType, unsigned int _limit, double _hzRate)
Constructor.
- virtual ~**Publisher** ()
Destructor.
- void **Fini** ()
Finalize the publisher.
- std::string **GetMsgType** () const
Get the message type.
- unsigned int **GetOutgoingCount** () const
Get the number of outgoing messages.
- std::string **GetPrevMsg** () const
Get the previously published message.
- **MessagePtr GetPrevMsgPtr** () const
Get the previously published message.
- std::string **GetTopic** () const
Get the topic name.
- bool **HasConnections** () const
Are there any connections?
- void **Publish** (const google::protobuf::Message &_message, bool _block=false)
Publish a protobuf message on the topic.
- template<typename M >
void **Publish** (M _message, bool _block=false)
Publish an arbitrary message on the topic.
- void **SendMessage** ()
Send latest message over the wire. For internal use only.
- void **SetNode** (**NodePtr** _node)
Set our containing node.
- void **SetPublication** (**PublicationPtr** _publication)
Set the publication object for a particular publication.
- void **WaitForConnection** () const
Block until a connection has been established with this publisher.
- bool **WaitForConnection** (const **common::Time** &_timeout) const
Block until a connection has been established with this publisher.

10.170.1 Detailed Description

A publisher of messages on a topic.

10.170.2 Constructor & Destructor Documentation

10.170.2.1 `gazebo::transport::Publisher::Publisher (const std::string & _topic, const std::string & _msgType, unsigned int _limit, double _hzRate)`

Constructor.

Parameters

<code>in</code>	<code><i>_topic</i></code>	Name of topic to be published
<code>in</code>	<code><i>_msgType</i></code>	Type of the message to be published
<code>in</code>	<code><i>_limit</i></code>	Maximum number of outgoing messages to queue
<code>in</code>	<code><i>_hz</i></code>	Update rate for the publisher. Units are 1.0/seconds.

10.170.2.2 `virtual gazebo::transport::Publisher::~~Publisher () [virtual]`

Destructor.

10.170.3 Member Function Documentation

10.170.3.1 `void gazebo::transport::Publisher::Fini ()`

Finalize the publisher.

10.170.3.2 `std::string gazebo::transport::Publisher::GetMsgType () const`

Get the message type.

Returns

The message type

10.170.3.3 `unsigned int gazebo::transport::Publisher::GetOutgoingCount () const`

Get the number of outgoing messages.

Returns

The number of outgoing messages

10.170.3.4 `std::string gazebo::transport::Publisher::GetPrevMsg () const`

Get the previously published message.

Returns

The previously published message, if any

10.170.3.5 MessagePtr gazebo::transport::Publisher::GetPrevMsgPtr () const

Get the previously published message.

Returns

The previously published message, if any

10.170.3.6 std::string gazebo::transport::Publisher::GetTopic () const

Get the topic name.

Returns

The topic name

10.170.3.7 bool gazebo::transport::Publisher::HasConnections () const

Are there any connections?

Returns

true if there are any connections, false otherwise

**10.170.3.8 void gazebo::transport::Publisher::Publish (const google::protobuf::Message & _message, bool _block = false)
[inline]**

Publish a protobuf message on the topic.

Parameters

in	<i>_message</i>	Message to be published
in	<i>_block</i>	Whether to block until the message is actually written out

**10.170.3.9 template<typename M > void gazebo::transport::Publisher::Publish (M _message, bool _block = false)
[inline]**

Publish an arbitrary message on the topic.

Parameters

in	<i>_message</i>	Message to be published
in	<i>_block</i>	Whether to block until the message is actually written out

10.170.3.10 void gazebo::transport::Publisher::SendMessage ()

Send latest message over the wire. For internal use only.

10.170.3.11 void gazebo::transport::Publisher::SetNode (NodePtr *_node*)

Set our containing node.

Parameters

in	<i>_node</i>	Pointer to a node. Should be the node that create this publisher.
----	--------------	---

10.170.3.12 void gazebo::transport::Publisher::SetPublication (PublicationPtr *_publication*)

Set the publication object for a particular publication.

Parameters

in	<i>_publication</i>	Pointer to the publication object to be set
----	---------------------	---

10.170.3.13 void gazebo::transport::Publisher::WaitForConnection () const

Block until a connection has been established with this publisher.

10.170.3.14 bool gazebo::transport::Publisher::WaitForConnection (const common::Time & *_timeout*) const

Block until a connection has been established with this publisher.

Parameters

in	<i>_timeout</i>	Maxiumum time to wait. Use a negative time value to wait forever.
----	-----------------	---

Returns

True if a connection was established.

The documentation for this class was generated from the following file:

- **Publisher.hh**

10.171 gazebo::transport::PublishTask Class Reference

```
#include <Node.hh>
```

Public Member Functions

- **PublishTask** (transport::PublisherPtr *_pub*, const google::protobuf::Message & *_message*)

Constructor.

- `tbb::task * execute ()`

Overridden function from `tbb::task` that executes the publish task.

10.171.1 Detailed Description

Task used by `Node::Publish` (p. 736) to publish on a one-time publisher

10.171.2 Constructor & Destructor Documentation

10.171.2.1 `gazebo::transport::PublishTask::PublishTask (transport::PublisherPtr _pub, const google::protobuf::Message & _message) [inline]`

Constructor.

Parameters

in	<code>_pub</code>	Publisher (p. 820) to publish the message on.
in	<code>_message</code>	Message to publish

10.171.3 Member Function Documentation

10.171.3.1 `tbb::task* gazebo::transport::PublishTask::execute () [inline]`

Overridden function from `tbb::task` that executes the publish task.

References NULL.

The documentation for this class was generated from the following file:

- **Node.hh**

10.172 gazebo::math::Quaternion Class Reference

A quaternion class.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Quaternion** ()

Default Constructor.

- **Quaternion** (const double &_w, const double &_x, const double &_y, const double &_z)

Constructor.

- **Quaternion** (const double &_roll, const double &_pitch, const double &_yaw)

Constructor from Euler angles in radians.

- **Quaternion** (const **Vector3** &_axis, const double &_angle)

Constructor from axis angle.

- **Quaternion** (const **Vector3** &_rpy)

Constructor.

- **Quaternion** (const **Quaternion** &_qt)

Copy constructor.

- **~Quaternion** ()

Destructor.

- void **Correct** ()

Correct any nan.

- double **Dot** (const **Quaternion** &_q) const

Dot product.

- void **GetAsAxis** (**Vector3** &_axis, double &_angle) const

Return rotation as axis and angle.

- **Vector3 GetAsEuler** () const

Return the rotation in Euler angles.

- **Matrix3 GetAsMatrix3** () const

Get the quaternion as a 3x3 matrix.

- **Matrix4 GetAsMatrix4** () const

Get the quaternion as a 4x4 matrix.

- **Quaternion GetExp** () const

Return the exponent.

- **Quaternion GetInverse** () const

Get the inverse of this quaternion.

- **Quaternion GetLog** () const

Return the logarithm.

- double **GetPitch** ()

Get the Euler pitch angle in radians.

- double **GetRoll** ()

Get the Euler roll angle in radians.

- **Vector3 GetXAxis** () const

Return the X axis.

- double **GetYaw** ()

Get the Euler yaw angle in radians.

- **Vector3 GetYAxis** () const

Return the Y axis.

- **Vector3 GetZAxis** () const

Return the Z axis.

- void **Invert** ()

Invert the quaternion.

- bool **IsFinite** () const

See if a quatern is finite (e.g., not nan)

- void **Normalize** ()

Normalize the quaternion.

- bool **operator!=** (const **Quaternion** &_qt) const

Not equal to operator.

- **Quaternion operator*** (const **Quaternion** &_q) const

Multiplication operator.

- **Quaternion operator*** (const double &_f) const

Multiplication operator.

- **Vector3 operator*** (const **Vector3** &_v) const
Vector3 (p. 1165) multiplication operator.
- **Quaternion operator*=** (const **Quaternion** &qt)
Multiplication operator.
- **Quaternion operator+** (const **Quaternion** &_qt) const
Addition operator.
- **Quaternion operator+=** (const **Quaternion** &_qt)
Addition operator.
- **Quaternion operator-** (const **Quaternion** &_qt) const
Substraction operator.
- **Quaternion operator-** () const
Unary minus operator.
- **Quaternion operator-=** (const **Quaternion** &_qt)
Substraction operator.
- **Quaternion & operator=** (const **Quaternion** &_qt)
Equal operator.
- bool **operator==** (const **Quaternion** &_qt) const
Equal to operator.
- **Vector3 RotateVector** (const **Vector3** &_vec) const
Rotate a vector using the quaternion.
- **Vector3 RotateVectorReverse** (**Vector3** _vec) const
Do the reverse rotation of a vector by this quaternion.
- void **Round** (int _precision)
Round all values to _precision decimal places.
- void **Scale** (double _scale)
Scale a Quaternionion.
- void **Set** (double _u, double _x, double _y, double _z)
Set this quaternion from 4 floating numbers.
- void **SetFromAxis** (double _x, double _y, double _z, double _a)
Set the quaternion from an axis and angle.
- void **SetFromAxis** (const **Vector3** &_axis, double _a)
Set the quaternion from an axis and angle.
- void **SetFromEuler** (const **Vector3** &_vec)
Set the quaternion from Euler angles.
- void **SetFromEuler** (double _roll, double _pitch, double _yaw)
Set the quaternion from Euler angles.
- void **SetTolIdentity** ()
Set the quatern to the identity.

Static Public Member Functions

- static **Quaternion EulerToQuaternion** (const **Vector3** &_vec)
Convert euler angles to quatern.
- static **Quaternion EulerToQuaternion** (double _x, double _y, double _z)
Convert euler angles to quatern.
- static **Quaternion Slerp** (double _fT, const **Quaternion** &_rkP, const **Quaternion** &_rkQ, bool _shortest-Path=false)

Spherical linear interpolation between 2 quaternions, given the ends and an interpolation parameter between 0 and 1.

- static **Quaternion Squad** (double *_t*, const **Quaternion** &*_rkP*, const **Quaternion** &*_rkA*, const **Quaternion** &*_rkB*, const **Quaternion** &*_rkQ*, bool *_shortestPath*=false)

Spherical quadratic interpolation given the ends and an interpolation parameter between 0 and 1.

Public Attributes

- double **w**
Attributes of the quaternion.
- double **x**
Attributes of the quaternion.
- double **y**
Attributes of the quaternion.
- double **z**
Attributes of the quaternion.

Friends

- std::ostream & **operator**<< (std::ostream &*_out*, const gazebo::math::Quaternion &*_q*)
Stream insertion operator.
- std::istream & **operator**>> (std::istream &*_in*, gazebo::math::Quaternion &*_q*)
Stream extraction operator.

10.172.1 Detailed Description

A quaternion class.

10.172.2 Constructor & Destructor Documentation

10.172.2.1 gazebo::math::Quaternion::Quaternion ()

Default Constructor.

10.172.2.2 gazebo::math::Quaternion::Quaternion (const double & *_w*, const double & *_x*, const double & *_y*, const double & *_z*)

Constructor.

Parameters

<i>in</i>	<i>_w</i>	W param
<i>in</i>	<i>_x</i>	X param
<i>in</i>	<i>_y</i>	Y param
<i>in</i>	<i>_z</i>	Z param

10.172.2.3 gazebo::math::Quaternion::Quaternion (const double & *_roll*, const double & *_pitch*, const double & *_yaw*)

Constructor from Euler angles in radians.

Parameters

in	<code>_roll</code>	roll
in	<code>_pitch</code>	pitch
in	<code>_yaw</code>	yaw

10.172.2.4 `gazebo::math::Quaternion::Quaternion (const Vector3 & .axis, const double & .angle)`

Constructor from axis angle.

Parameters

in	<code>_axis</code>	the rotation axis
in	<code>_angle</code>	the rotation angle in radians

10.172.2.5 `gazebo::math::Quaternion::Quaternion (const Vector3 & .rpy)`

Constructor.

Parameters

in	<code>_rpy</code>	euler angles
----	-------------------	--------------

10.172.2.6 `gazebo::math::Quaternion::Quaternion (const Quaternion & .qt)`

Copy constructor.

Parameters

<code>qt</code>	Quaternion (p. 824) to copy
-----------------	------------------------------------

10.172.2.7 `gazebo::math::Quaternion::~~Quaternion ()`

Destructor.

10.172.3 Member Function Documentation

10.172.3.1 `void gazebo::math::Quaternion::Correct () [inline]`

Correct any nan.

References `gazebo::math::equal()`.

10.172.3.2 `double gazebo::math::Quaternion::Dot (const Quaternion & .q) const`

Dot product.

Parameters

<code>in</code>	<code>_q</code>	the other quaternion
-----------------	-----------------	----------------------

Returns

the product

10.172.3.3 `static Quaternion gazebo::math::Quaternion::EulerToQuaternion (const Vector3 & _vec) [static]`

Convert euler angles to quaternion.

Parameters

<code>in</code>		
-----------------	--	--

10.172.3.4 `static Quaternion gazebo::math::Quaternion::EulerToQuaternion (double _x, double _y, double _z) [static]`

Convert euler angles to quaternion.

Parameters

<code>in</code>	<code>_x</code>	rotation along x
<code>in</code>	<code>_y</code>	rotation along y
<code>in</code>	<code>_z</code>	rotation along z

10.172.3.5 `void gazebo::math::Quaternion::GetAsAxis (Vector3 & _axis, double & _angle) const`

Return rotation as axis and angle.

Parameters

<code>in</code>	<code>_axis</code>	rotation axis
<code>in</code>	<code>_angle</code>	ccw angle in radians

10.172.3.6 `Vector3 gazebo::math::Quaternion::GetAsEuler () const`

Return the rotation in Euler angles.

Returns

This quaternion as an Euler vector

10.172.3.7 `Matrix3 gazebo::math::Quaternion::GetAsMatrix3 () const`

Get the quaternion as a 3x3 matrix.

10.172.3.8 Matrix4 gazebo::math::Quaternion::GetAsMatrix4 () const

Get the quaternion as a 4x4 matrix.

Returns

a 4x4 matrix

10.172.3.9 Quaternion gazebo::math::Quaternion::GetExp () const

Return the exponent.

Returns

the exp

10.172.3.10 Quaternion gazebo::math::Quaternion::GetInverse () const [inline]

Get the inverse of this quaternion.

Returns

Inverse quarenion

References gazebo::math::equal(), w, x, y, and z.

Referenced by gazebo::math::Pose::CoordPositionSub(), and gazebo::math::Pose::CoordRotationSub().

10.172.3.11 Quaternion gazebo::math::Quaternion::GetLog () const

Return the logarithm.

Returns

the log

10.172.3.12 double gazebo::math::Quaternion::GetPitch ()

Get the Euler pitch angle in radians.

Returns

the pitch

10.172.3.13 double gazebo::math::Quaternion::GetRoll ()

Get the Euler roll angle in radians.

Returns

the roll

10.172.3.14 **Vector3** gazebo::math::Quaternion::GetXAxis () const

Return the X axis.

Returns

the vector

10.172.3.15 **double** gazebo::math::Quaternion::GetYaw ()

Get the Euler yaw angle in radians.

Returns

the yaw

10.172.3.16 **Vector3** gazebo::math::Quaternion::GetYAxis () const

Return the Y axis.

Returns

the vector

10.172.3.17 **Vector3** gazebo::math::Quaternion::GetZAxis () const

Return the Z axis.

Returns

the vector

10.172.3.18 **void** gazebo::math::Quaternion::Invert ()

Invert the quaternion.

10.172.3.19 **bool** gazebo::math::Quaternion::IsFinite () const

See if a quatern is finite (e.g., not nan)

Returns

True if quatern is finite

10.172.3.20 **void** gazebo::math::Quaternion::Normalize ()

Normalize the quaternion.

Referenced by gazebo::math::Pose::CoordRotationSub().

10.172.3.21 `bool gazebo::math::Quaternion::operator!=(const Quaternion & _qt) const`

Not equal to operator.

Parameters

<code>in</code>	<code>_qt</code>	Quaternion (p. 824) for comparison
-----------------	------------------	---

Returns

True if not equal

10.172.3.22 `Quaternion gazebo::math::Quaternion::operator*(const Quaternion & _q) const` `[inline]`

Multiplication operator.

Parameters

<code>in</code>	<code>_qt</code>	Quaternion (p. 824) for multiplication
-----------------	------------------	---

Returns

This quaternion multiplied by the parameter

References w, x, y, and z.

10.172.3.23 `Quaternion gazebo::math::Quaternion::operator*(const double & _f) const`

Multiplication operator.

Parameters

<code>in</code>	<code>_f</code>	factor
-----------------	-----------------	--------

Returns

quaternion multiplied by `_f`

10.172.3.24 `Vector3 gazebo::math::Quaternion::operator*(const Vector3 & _v) const`

Vector3 (p. 1165) multiplication operator.

Parameters

<code>in</code>	<code>_v</code>	vector to multiply
-----------------	-----------------	--------------------

10.172.3.25 `Quaternion gazebo::math::Quaternion::operator*=(const Quaternion & qt)`

Multiplication operator.

Parameters

in	_qt	Quaternion (p. 824) for multiplication
----	-----	--

Returns

This quatern multiplied by the parameter

10.172.3.26 Quaternion gazebo::math::Quaternion::operator+ (const Quaternion & _qt) const

Addition operator.

Parameters

in	_qt	quaternion for addition
----	-----	-------------------------

Returns

this quaternion + _qt

10.172.3.27 Quaternion gazebo::math::Quaternion::operator+= (const Quaternion & _qt)

Addition operator.

Parameters

in	_qt	quaternion for addition
----	-----	-------------------------

Returns

this quaternion + qt

10.172.3.28 Quaternion gazebo::math::Quaternion::operator- (const Quaternion & _qt) const

Substraction operator.

Parameters

in	_qt	quaternion to substract
----	-----	-------------------------

Returns

this quaternion - _qt

10.172.3.29 Quaternion gazebo::math::Quaternion::operator- () const

Unary minus operator.

Returns

negates each component of the quaternion

10.172.3.30 **Quaternion** gazebo::math::Quaternion::operator-= (const Quaternion & *_qt*)

Substraction operator.

Parameters

<i>in</i>	<i>_qt</i>	Quaternion (p. 824) for subtraction
-----------	------------	--

Returns

This quatern - qt

10.172.3.31 **Quaternion&** gazebo::math::Quaternion::operator= (const Quaternion & *_qt*)

Equal operator.

Parameters

<i>in</i>	<i>_qt</i>	Quaternion (p. 824) to copy
-----------	------------	------------------------------------

10.172.3.32 **bool** gazebo::math::Quaternion::operator== (const Quaternion & *_qt*) const

Equal to operator.

Parameters

<i>in</i>	<i>_qt</i>	Quaternion (p. 824) for comparison
-----------	------------	---

Returns

True if equal

10.172.3.33 **Vector3** gazebo::math::Quaternion::RotateVector (const Vector3 & *_vec*) const `[inline]`

Rotate a vector using the quaternion.

Parameters

<i>in</i>	<i>_vec</i>	vector to rotate
-----------	-------------	------------------

Returns

the rotated vector

References gazebo::math::Vector3::x, x, gazebo::math::Vector3::y, y, gazebo::math::Vector3::z, and z.

10.172.3.34 `Vector3 gazebo::math::Quaternion::RotateVectorReverse (Vector3 _vec) const`

Do the reverse rotation of a vector by this quaternion.

Parameters

<code>in</code>	<code>_vec</code>	the vector
-----------------	-------------------	------------

Returns

the

10.172.3.35 `void gazebo::math::Quaternion::Round (int _precision)`

Round all values to `_precision` decimal places.

Parameters

<code>in</code>	<code>_precision</code>	the precision
-----------------	-------------------------	---------------

10.172.3.36 `void gazebo::math::Quaternion::Scale (double _scale)`

Scale a Quaternionion.

Parameters

<code>in</code>	<code>_scale</code>	Amount to scale this rotation
-----------------	---------------------	-------------------------------

10.172.3.37 `void gazebo::math::Quaternion::Set (double _u, double _x, double _y, double _z)`

Set this quaternion from 4 floating numbers.

Parameters

<code>in</code>	<code>_u</code>	u
<code>in</code>	<code>_x</code>	x
<code>in</code>	<code>_y</code>	y
<code>in</code>	<code>_z</code>	z

10.172.3.38 `void gazebo::math::Quaternion::SetFromAxis (double _x, double _y, double _z, double _a)`

Set the quaternion from an axis and angle.

Parameters

<code>in</code>	<code>_x</code>	X axis
<code>in</code>	<code>_y</code>	Y axis
<code>in</code>	<code>_z</code>	Z axis
<code>in</code>	<code>_a</code>	Angle (p. 145) in radians

10.172.3.39 `void gazebo::math::Quaternion::SetFromAxis (const Vector3 & _axis, double _a)`

Set the quaternion from an axis and angle.

Parameters

in	<code>_axis</code>	Axis
in	<code>_a</code>	Angle (p. 145) in radians

10.172.3.40 `void gazebo::math::Quaternion::SetFromEuler (const Vector3 & _vec)`

Set the quaternion from Euler angles.

The order of operations are roll, pitch, yaw.

Parameters

in	<code>vec</code>	Euler angle
----	------------------	-------------

10.172.3.41 `void gazebo::math::Quaternion::SetFromEuler (double _roll, double _pitch, double _yaw)`

Set the quaternion from Euler angles.

Parameters

in	<code>_roll</code>	Roll angle (radians).
in	<code>_pitch</code>	Roll angle (radians).
in	<code>_yaw</code>	Roll angle (radians).

10.172.3.42 `void gazebo::math::Quaternion::SetToIdentity ()`

Set the quatern to the identity.

10.172.3.43 `static Quaternion gazebo::math::Quaternion::Slerp (double _ft, const Quaternion & _rkP, const Quaternion & _rkQ, bool _shortestPath = false) [static]`

Spherical linear interpolation between 2 quaternions, given the ends and an interpolation parameter between 0 and 1.

Parameters

in	<code>_ft</code>	the interpolation parameter
in	<code>_rkP</code>	the beginning quaternion
in	<code>_rkQ</code>	the end quaternion
in	<code>_shortestPath</code>	when true, the rotation may be inverted to get to minimize rotation

10.172.3.44 `static Quaternion gazebo::math::Quaternion::Squad (double _ft, const Quaternion & _rkP, const Quaternion & _rkA, const Quaternion & _rkB, const Quaternion & _rkQ, bool _shortestPath = false) [static]`

Spherical quadratic interpolation given the ends and an interpolation parameter between 0 and 1.

Parameters

in	<code>_ft</code>	the interpolation parameter
in	<code>_rkP</code>	the beginning quaternion
in	<code>_rkA</code>	first intermediate quaternion
in	<code>_rkB</code>	second intermediate quaternion
in	<code>_rkQ</code>	the end quaternion
in	<code>_shortestPath</code>	when true, the rotation may be inverted to get to minimize rotation

10.172.4 Friends And Related Function Documentation

10.172.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Quaternion & _q)` [friend]

Stream insertion operator.

Parameters

in	<code>_out</code>	output stream
in	<code>_q</code>	quaternion to output

Returns

the stream

10.172.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Quaternion & _q)` [friend]

Stream extraction operator.

Parameters

in	<code>_in</code>	input stream
in	<code>_q</code>	Quaternion (p. 824) to read values into

Returns

The istream

10.172.5 Member Data Documentation

10.172.5.1 `double gazebo::math::Quaternion::w`

Attributes of the quaternion.

Referenced by `gazebo::physics::DARTTypes::ConvQuat()`, `GetInverse()`, and `operator*()`.

10.172.5.2 `double gazebo::math::Quaternion::x`

Attributes of the quaternion.

Referenced by `gazebo::physics::DARTTypes::ConvQuat()`, `gazebo::math::Pose::CoordPositionSub()`, `GetInverse()`, `operator*()`, and `RotateVector()`.

10.172.5.3 double gazebo::math::Quaternion::y

Attributes of the quaternion.

Referenced by gazebo::physics::DARTTypes::ConvQuat(), gazebo::math::Pose::CoordPositionSub(), GetInverse(), operator*(), and RotateVector().

10.172.5.4 double gazebo::math::Quaternion::z

Attributes of the quaternion.

Referenced by gazebo::physics::DARTTypes::ConvQuat(), gazebo::math::Pose::CoordPositionSub(), GetInverse(), operator*(), and RotateVector().

The documentation for this class was generated from the following file:

- **Quaternion.hh**

10.173 gazebo::math::Rand Class Reference

Random number generator class.

```
#include <gzmath/gzmath.hh>
```

Static Public Member Functions

- static double **GetDbfNormal** (double _mean=0, double _sigma=1)
Get a double from a normal distribution.
- static double **GetDbfUniform** (double _min=0, double _max=1)
Get a double from a uniform distribution.
- static int **GetIntNormal** (int _mean, int _sigma)
Get a double from a normal distribution.
- static int **GetIntUniform** (int _min, int _max)
Get a integer from a uniform distribution.
- static uint32_t **GetSeed** ()
Get the seed value.
- static void **SetSeed** (uint32_t _seed)
Set the seed value.

10.173.1 Detailed Description

Random number generator class.

10.173.2 Member Function Documentation

10.173.2.1 static double gazebo::math::Rand::GetDbfNormal (double *_mean* = 0, double *_sigma* = 1) [static]

Get a double from a normal distribution.

Parameters

in	<code>_mean</code>	Mean value for the distribution
in	<code>_sigma</code>	Sigma value for the distribution

10.173.2.2 `static double gazebo::math::Rand::GetDbUniform (double _min = 0, double _max = 1) [static]`

Get a double from a uniform distribution.

Parameters

in	<code>_min</code>	Minimum bound for the random number
in	<code>_max</code>	Maximum bound for the random number

10.173.2.3 `static int gazebo::math::Rand::GetIntNormal (int _mean, int _sigma) [static]`

Get a double from a normal distribution.

Parameters

in	<code>_mean</code>	Mean value for the distribution
in	<code>_sigma</code>	Sigma value for the distribution

10.173.2.4 `static int gazebo::math::Rand::GetIntUniform (int _min, int _max) [static]`

Get a integer from a uniform distribution.

Parameters

in	<code>_min</code>	Minimum bound for the random number
in	<code>_max</code>	Maximum bound for the random number

10.173.2.5 `static uint32_t gazebo::math::Rand::GetSeed () [static]`

Get the seed value.

Returns

The seed value used to initialize the random number generator.

10.173.2.6 `static void gazebo::math::Rand::SetSeed (uint32_t _seed) [static]`

Set the seed value.

Parameters

in	<code>_seed</code>	The seed used to initialize the random number generator.
----	--------------------	--

The documentation for this class was generated from the following file:

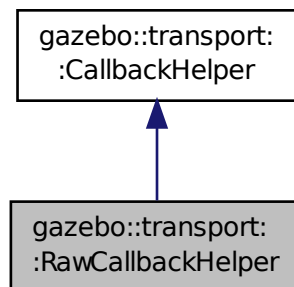
- **Rand.hh**

10.174 gazebo::transport::RawCallbackHelper Class Reference

Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.

```
#include <CallbackHelper.hh>
```

Inheritance diagram for gazebo::transport::RawCallbackHelper:



Public Member Functions

- **RawCallbackHelper** (const boost::function< void(const std::string &);> &_cb, bool _latching=false)
Constructor.
- std::string **GetMsgType** () const
Get the typename of the message that is handled.
- virtual bool **HandleData** (const std::string &_newdata, boost::function< void(uint32_t);> _cb, uint32_t _id)
Process new incoming data.
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)
Process new incoming message.
- virtual bool **IsLocal** () const
Is the callback local?

Additional Inherited Members

10.174.1 Detailed Description

Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.

Raw means that the data has not been converted into a protobuf message.

10.174.2 Constructor & Destructor Documentation

10.174.2.1 `gazebo::transport::RawCallbackHelper::RawCallbackHelper (const boost::function< void(const std::string &)> & _cb, bool _latching = false) [inline]`

Constructor.

Parameters

in	<i>_cb</i>	boost function to call on incoming messages
in	<i>_latching</i>	Set to true to make the callback helper latching.

10.174.3 Member Function Documentation

10.174.3.1 `std::string gazebo::transport::RawCallbackHelper::GetMsgType () const [inline],[virtual]`

Get the typename of the message that is handled.

Returns

String representation of the message type

Reimplemented from `gazebo::transport::CallbackHelper` (p. 193).

10.174.3.2 `virtual bool gazebo::transport::RawCallbackHelper::HandleData (const std::string & _newdata, boost::function< void(uint32_t)> _cb, uint32_t _id) [inline],[virtual]`

Process new incoming data.

Parameters

in	<i>_newdata</i>	Incoming data to be processed
----	-----------------	-------------------------------

Returns

true if successfully processed; false otherwise

Parameters

in	<i>_cb</i>	If non-null, callback to be invoked which signals that transmission is complete.
in	<i>_id</i>	ID associated with the message data.

Implements `gazebo::transport::CallbackHelper` (p. 194).

10.174.3.3 `virtual bool gazebo::transport::RawCallbackHelper::HandleMessage (MessagePtr _newMsg) [inline],[virtual]`

Process new incoming message.

Parameters

in	<code>_newMsg</code>	Incoming message to be processed
----	----------------------	----------------------------------

Returns

true if successfully processed; false otherwise

Implements `gazebo::transport::CallbackHelper` (p. 194).

10.174.3.4 `virtual bool gazebo::transport::RawCallbackHelper::IsLocal () const` `[inline],[virtual]`

Is the callback local?

Returns

true if the callback is local, false if the callback is tied to a remote connection

Implements `gazebo::transport::CallbackHelper` (p. 194).

The documentation for this class was generated from the following file:

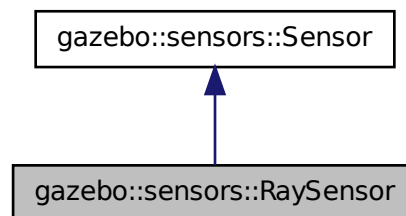
- `CallbackHelper.hh`

10.175 gazebo::sensors::RaySensor Class Reference

Sensor (p. 907) with one or more rays.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for `gazebo::sensors::RaySensor`:



Public Member Functions

- `RaySensor ()`
Constructor.
- `virtual ~RaySensor ()`

Destructor.

- **math::Angle GetAngleMax** () const
Get the maximum angle.
- **math::Angle GetAngleMin** () const
Get the minimum angle.
- double **GetAngleResolution** () const
Get the angle in radians between each range.
- int **GetFiducial** (unsigned int _index)
Get detected fiducial value for a ray.
- **physics::MultiRayShapePtr GetLaserShape** () const
*Returns a pointer to the internal **physics::MultiRayShape** (p. 723).*
- double **GetRange** (unsigned int _index)
Get detected range for a ray.
- int **GetRangeCount** () const
Get the range count.
- double **GetRangeMax** () const
Get the maximum range.
- double **GetRangeMin** () const
Get the minimum range.
- double **GetRangeResolution** () const
Get the range resolution.
- void **GetRanges** (std::vector< double > &_ranges)
Get all the ranges.
- int **GetRayCount** () const
Get the ray count.
- double **GetRetro** (unsigned int _index)
Get detected retro (intensity) value for a ray.
- virtual std::string **GetTopic** () const
Returns the topic name as set in SDF.
- **math::Angle GetVerticalAngleMax** () const
Get the vertical scan line top angle.
- **math::Angle GetVerticalAngleMin** () const
Get the vertical scan bottom angle.
- double **GetVerticalAngleResolution** () const
Get the vertical angle in radians between each range.
- int **GetVerticalRangeCount** () const
Get the vertical scan line count.
- int **GetVerticalRayCount** () const
Get the vertical scan line count.
- virtual void **Init** ()
Initialize the sensor.
- virtual bool **IsActive** ()
Returns true if sensor generation is active.
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.

Protected Member Functions

- virtual void **Fini** ()
Finalize the sensor.
- virtual bool **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.175.1 Detailed Description

Sensor (p. 907) with one or more rays.

This sensor cast rays into the world, tests for intersections, and reports the range to the nearest object. It is used by ranging sensor models (e.g., sonars and scanning laser range finders).

10.175.2 Constructor & Destructor Documentation

10.175.2.1 gazebo::sensors::RaySensor::RaySensor ()

Constructor.

10.175.2.2 virtual gazebo::sensors::RaySensor::~~RaySensor () [virtual]

Destructor.

10.175.3 Member Function Documentation

10.175.3.1 virtual void gazebo::sensors::RaySensor::Fini () [protected],[virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 912).

10.175.3.2 math::Angle gazebo::sensors::RaySensor::GetAngleMax () const

Get the maximum angle.

Returns

the maximum angle object

10.175.3.3 math::Angle gazebo::sensors::RaySensor::GetAngleMin () const

Get the minimum angle.

Returns

The minimum angle object

10.175.3.4 double gazebo::sensors::RaySensor::GetAngleResolution () const

Get the angle in radians between each range.

Returns

Resolution of the angle

10.175.3.5 int gazebo::sensors::RaySensor::GetFiducial (unsigned int *_index*)

Get detected fiducial value for a ray.

Warning: If you are accessing all the ray data in a loop it's possible that the Ray will update in the middle of your access loop. This means some data will come from one scan, and some from another scan. You can solve this problem by using `SetActive(false)` <your accessor loop> `SetActive(true)`.

Parameters

in	<i>_index</i>	Index value of specific ray
----	---------------	-----------------------------

Returns

Fiducial value

10.175.3.6 physics::MultiRayShapePtr gazebo::sensors::RaySensor::GetLaserShape () const [inline]

Returns a pointer to the internal `physics::MultiRayShape` (p. 723).

Returns

Pointer to ray shape

10.175.3.7 double gazebo::sensors::RaySensor::GetRange (unsigned int *_index*)

Get detected range for a ray.

Warning: If you are accessing all the ray data in a loop it's possible that the Ray will update in the middle of your access loop. This means some data will come from one scan, and some from another scan. You can solve this problem by using `SetActive(false)` <your accessor loop> `SetActive(true)`.

Parameters

in	<i>_index</i>	Index of specific ray
----	---------------	-----------------------

Returns

Returns DBL_MAX for no detection.

10.175.3.8 `int gazebo::sensors::RaySensor::GetRangeCount () const`

Get the range count.

Returns

The number of ranges

10.175.3.9 `double gazebo::sensors::RaySensor::GetRangeMax () const`

Get the maximum range.

Returns

The maximum range

10.175.3.10 `double gazebo::sensors::RaySensor::GetRangeMin () const`

Get the minimum range.

Returns

The minimum range

10.175.3.11 `double gazebo::sensors::RaySensor::GetRangeResolution () const`

Get the range resolution.

Returns

Resolution of the range

10.175.3.12 `void gazebo::sensors::RaySensor::GetRanges (std::vector< double > & _ranges)`

Get all the ranges.

Parameters

<code><i>_ranges</i></code>	A vector that will contain all the range data
-----------------------------	---

10.175.3.13 `int gazebo::sensors::RaySensor::GetRayCount () const`

Get the ray count.

Returns

The number of rays

10.175.3.14 `double gazebo::sensors::RaySensor::GetRetro (unsigned int _index)`

Get detected retro (intensity) value for a ray.

Warning: If you are accessing all the ray data in a loop it's possible that the Ray will update in the middle of your access loop. This means some data will come from one scan, and some from another scan. You can solve this problem by using `SetActive(false)` <your accessor loop> `SetActive(true)`.

Parameters

in	<i>_index</i>	Index of specific ray
----	---------------	-----------------------

Returns

Retro (intensity) value for ray

10.175.3.15 `virtual std::string gazebo::sensors::RaySensor::GetTopic () const` [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented from `gazebo::sensors::Sensor` (p. 914).

10.175.3.16 `math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMax () const`

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.175.3.17 `math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMin () const`

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.175.3.18 `double gazebo::sensors::RaySensor::GetVerticalAngleResolution () const`

Get the vertical angle in radians between each range.

Returns

Resolution of the angle

10.175.3.19 `int gazebo::sensors::RaySensor::GetVerticalRangeCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.175.3.20 `int gazebo::sensors::RaySensor::GetVerticalRayCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.175.3.21 `virtual void gazebo::sensors::RaySensor::Init () [virtual]`

Initialize the sensor.

Reimplemented from **`gazebo::sensors::Sensor`** (p. 915).

10.175.3.22 `virtual bool gazebo::sensors::RaySensor::IsActive () [virtual]`

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from **`gazebo::sensors::Sensor`** (p. 915).

10.175.3.23 `virtual void gazebo::sensors::RaySensor::Load (const std::string & _worldName) [virtual]`

Load the sensor with default parameters.

Parameters

<code>in</code>	<code>_worldName</code>	Name of world to load from.
-----------------	-------------------------	-----------------------------

Reimplemented from **`gazebo::sensors::Sensor`** (p. 915).

10.175.3.24 `virtual bool gazebo::sensors::RaySensor::UpdateImpl(bool)` [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	<code>_force</code>	True if update is forced, false if not
----	---------------------	--

Returns

True if the sensor was updated.

Reimplemented from `gazebo::sensors::Sensor` (p. 917).

The documentation for this class was generated from the following file:

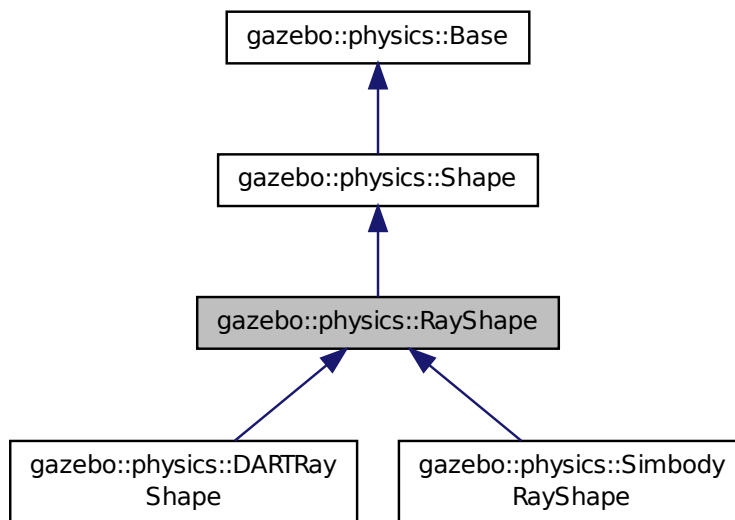
- `RaySensor.hh`

10.176 gazebo::physics::RayShape Class Reference

Base (p. 168) class for Ray collision geometry.

```
#include <physics/physics.hh>
```

Inheritance diagram for `gazebo::physics::RayShape`:



Public Member Functions

- **RayShape** (**PhysicsEnginePtr** _physicsEngine)
Constructor for a global ray.
- **RayShape** (**CollisionPtr** _parent)
Constructor.
- virtual \sim **RayShape** ()
Destructor.
- void **FillMsg** (msgs::Geometry &_msg)
Fill a message with data from this object.
- int **GetFiducial** () const
Get the fiducial id detected by this ray.
- virtual void **GetGlobalPoints** (**math::Vector3** &_posA, **math::Vector3** &_posB)
Get the global starting and ending points.
- virtual void **GetIntersection** (double &_dist, std::string &_entity)=0
Get the nearest intersection.
- double **GetLength** () const
Get the length of the ray.
- virtual void **GetRelativePoints** (**math::Vector3** &_posA, **math::Vector3** &_posB)
Get the relative starting and ending points.
- float **GetRetro** () const
Get the retro-reflectivness detected by this ray.
- virtual void **Init** ()
In the ray.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
Update this shape from a message.
- void **SetFiducial** (int _fid)
Set the fiducial id detected by this ray.
- virtual void **SetLength** (double _len)
Set the length of the ray.
- virtual void **SetPoints** (const **math::Vector3** &_posStart, const **math::Vector3** &_posEnd)
Set the ray based on starting and ending points relative to the body.
- void **SetRetro** (float _retro)
Set the retro-reflectivness detected by this ray.
- virtual void **SetScale** (const **math::Vector3** &_scale)
Set the scale of the ray.
- virtual void **Update** ()=0
Update the ray collision.

Protected Attributes

- int **contactFiducial**
Fiducial ID value.
- double **contactLen**
Length of the ray.
- double **contactRetro**
Retro reflectance value.

- **math::Vector3 globalEndPos**
End position of the ray in global cs.
- **math::Vector3 globalStartPos**
Start position of the ray in global cs.
- **math::Vector3 relativeEndPos**
End position of the ray, relative to the body.
- **math::Vector3 relativeStartPos**
Start position of the ray, relative to the body.

Additional Inherited Members

10.176.1 Detailed Description

Base (p. 168) class for Ray collision geometry.

10.176.2 Constructor & Destructor Documentation

10.176.2.1 gazebo::physics::RayShape::RayShape (PhysicsEnginePtr *_physicsEngine*) [explicit]

Constructor for a global ray.

Parameters

in	<i>_physicsEngine</i>	Pointer to the physics engine.
----	-----------------------	--------------------------------

10.176.2.2 gazebo::physics::RayShape::RayShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Collision (p. 235) parent of the shape.
----	----------------	--

10.176.2.3 virtual gazebo::physics::RayShape::~~RayShape () [virtual]

Destructor.

10.176.3 Member Function Documentation

10.176.3.1 void gazebo::physics::RayShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Fill a message with data from this object.

Parameters

out	<i>_msg</i>	Message to fill. Implement this function.
-----	-------------	---

Implements **gazebo::physics::Shape** (p. 934).

10.176.3.2 `int gazebo::physics::RayShape::GetFiducial () const`

Get the fiducial id detected by this ray.

Returns

Fiducial id detected.

10.176.3.3 `virtual void gazebo::physics::RayShape::GetGlobalPoints (math::Vector3 & _posA, math::Vector3 & _posB)`
[virtual]

Get the global starting and ending points.

Parameters

out	<code>_posA</code>	Returns the starting point.
out	<code>_posB</code>	Returns the ending point.

10.176.3.4 `virtual void gazebo::physics::RayShape::GetIntersection (double & _dist, std::string & _entity)` [pure virtual]

Get the nearest intersection.

Parameters

out	<code>_dist</code>	Distance to the intersection.
out	<code>_entity</code>	Name of the entity the ray intersected with.

Implemented in `gazebo::physics::DARTRayShape` (p. 78), and `gazebo::physics::SimbodyRayShape` (p. 1001).

10.176.3.5 `double gazebo::physics::RayShape::GetLength () const`

Get the length of the ray.

Returns

The ray length.

10.176.3.6 `virtual void gazebo::physics::RayShape::GetRelativePoints (math::Vector3 & _posA, math::Vector3 & _posB)`
[virtual]

Get the relative starting and ending points.

Parameters

in	<code>_posA</code>	Returns the starting point.
in	<code>_posB</code>	Returns the ending point.

10.176.3.7 `float gazebo::physics::RayShape::GetRetro () const`

Get the retro-reflectivness detected by this ray.

Returns

Retro reflectance value.

10.176.3.8 `virtual void gazebo::physics::RayShape::Init () [virtual]`

In the ray.

Implements **`gazebo::physics::Shape`** (p. 935).

10.176.3.9 `virtual void gazebo::physics::RayShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]`

Update this shape from a message.

Parameters

in	<code>_msg</code>	Message to update from. Implement this function.
----	-------------------	--

Implements **`gazebo::physics::Shape`** (p. 935).

10.176.3.10 `void gazebo::physics::RayShape::SetFiducial (int _fid)`

Set the fiducial id detected by this ray.

Parameters

in	<code>_fid</code>	Fiducial id detected by this ray.
----	-------------------	-----------------------------------

10.176.3.11 `virtual void gazebo::physics::RayShape::SetLength (double _len) [virtual]`

Set the length of the ray.

Parameters

in	<code>_len</code>	Length of the array.
----	-------------------	----------------------

10.176.3.12 `virtual void gazebo::physics::RayShape::SetPoints (const math::Vector3 & _posStart, const math::Vector3 & _posEnd) [virtual]`

Set the ray based on starting and ending points relative to the body.

Parameters

in	<code>_posStart</code>	Start position, relative the body.
in	<code>_posEnd</code>	End position, relative to the body.

Reimplemented in **gazebo::physics::DARTRayShape** (p. 78), and **gazebo::physics::SimbodyRayShape** (p. 1001).

10.176.3.13 `void gazebo::physics::RayShape::SetRetro (float _retro)`

Set the retro-reflectivness detected by this ray.

Parameters

in	_retro	Retro reflectance value.
----	--------	--------------------------

10.176.3.14 `virtual void gazebo::physics::RayShape::SetScale (const math::Vector3 & _scale) [virtual]`

Set the scale of the ray.

Implements **gazebo::physics::Shape** (p. 935).

10.176.3.15 `virtual void gazebo::physics::RayShape::Update () [pure virtual]`

Update the ray collision.

Reimplemented from **gazebo::physics::Base** (p. 180).

Implemented in **gazebo::physics::DARTRayShape** (p. 78), and **gazebo::physics::SimbodyRayShape** (p. 1002).

10.176.4 Member Data Documentation

10.176.4.1 `int gazebo::physics::RayShape::contactFiducial [protected]`

Fiducial ID value.

10.176.4.2 `double gazebo::physics::RayShape::contactLen [protected]`

Length of the ray.

10.176.4.3 `double gazebo::physics::RayShape::contactRetro [protected]`

Retro reflectance value.

10.176.4.4 `math::Vector3 gazebo::physics::RayShape::globalEndPos [protected]`

End position of the ray in global cs.

10.176.4.5 `math::Vector3 gazebo::physics::RayShape::globalStartPos [protected]`

Start position of the ray in global cs.

10.176.4.6 `math::Vector3 gazebo::physics::RayShape::relativeEndPos [protected]`

End position of the ray, relative to the body.

10.176.4.7 `math::Vector3 gazebo::physics::RayShape::relativeStartPos` [protected]

Start position of the ray, relative to the body.

The documentation for this class was generated from the following file:

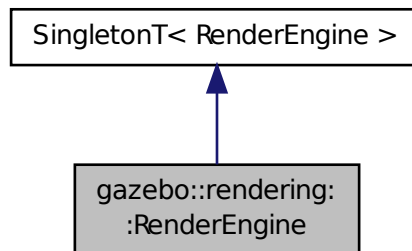
- **RayShape.hh**

10.177 gazebo::rendering::RenderEngine Class Reference

Adaptor to Ogre3d.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::RenderEngine:



Public Types

- enum **RenderPathType** {
NONE = 0, **VERTEX** = 1, **FORWARD** = 2, **DEFERRED** = 3,
RENDER_PATH_COUNT }

The type of rendering path used by the rendering engine.

Public Member Functions

- void **AddResourcePath** (const std::string &_uri)
*Add a new path for **Ogre** (p. 137) to search for resources.*
- **ScenePtr CreateScene** (const std::string &_name, bool _enableVisualizations, bool _isServer=false)
Create a scene.
- void **Fini** ()
Tears down the rendering engine.
- **RenderPathType GetRenderPathType** () const
Get the type of rendering path to use.
- **ScenePtr GetScene** (const std::string &_name="")

- Get a scene by name.*
- **ScenePtr GetScene** (unsigned int _index)
Get a scene by index.
- unsigned int **GetSceneCount** () const
Get the number of scenes.
- **WindowManagerPtr GetWindowManager** () const
Get a pointer to the window manager.
- void **Init** ()
*Initialize **Ogre** (p. 137). Load must happen before Init.*
- void **Load** ()
*Load the parameters for **Ogre** (p. 137). Load must happen before Init.*
- void **RemoveScene** (const std::string &_name)
Remove a scene.

Public Attributes

- Ogre::Root * **root**
Pointer to the root scene node.

Protected Attributes

- void * **dummyContext**
GLX context used to render the scenes. Used for gui-less operation.
- void * **dummyDisplay**
Pointer to the dummy display. Used for gui-less operation.
- uint64_t **dummyWindowId**
ID for a dummy window. Used for gui-less operation.

Additional Inherited Members

10.177.1 Detailed Description

Adaptor to Ogre3d.

Provides the interface to load, initialize the rendering engine.

10.177.2 Member Enumeration Documentation

10.177.2.1 enum gazebo::rendering::RenderEngine::RenderPathType

The type of rendering path used by the rendering engine.

Enumerator:

- NONE** No rendering is done.
- VERTEX** Most basic rendering, with least fidelity.
- FORWARD** Utilizes the RTT shader system.
- DEFERRED** Utilizes deferred rendering. Best fidelity.
- RENDER_PATH_COUNT** Count of the rendering path enums.

10.177.3 Member Function Documentation

10.177.3.1 void gazebo::rendering::RenderEngine::AddResourcePath (const std::string & _uri)

Add a new path for **Ogre** (p. 137) to search for resources.

Parameters

in	<code>_uri</code>	URI of the path. The uri should be of the form <code>file://</code> or <code>model://</code>
----	-------------------	--

10.177.3.2 ScenePtr gazebo::rendering::RenderEngine::CreateScene (const std::string & _name, bool _enableVisualizations, bool _isServer = false)

Create a scene.

Parameters

in	<code>_name</code>	The name of the scene.
in	<code>_enable-Visualizations</code>	True enables visualization elements such as laser lines.

10.177.3.3 void gazebo::rendering::RenderEngine::Fini ()

Tears down the rendering engine.

10.177.3.4 RenderPathType gazebo::rendering::RenderEngine::GetRenderPathType () const

Get the type of rendering path to use.

This is automatically determined based on the computers capabilities

Returns

The RenderPathType

10.177.3.5 ScenePtr gazebo::rendering::RenderEngine::GetScene (const std::string & _name = " ")

Get a scene by name.

Parameters

in	<code>_name</code>	Name of the scene to retrieve.
----	--------------------	--------------------------------

Returns

A pointer to the **Scene** (p. 879), or NULL if the scene doesn't exist.

10.177.3.6 ScenePtr gazebo::rendering::RenderEngine::GetScene (unsigned int _index)

Get a scene by index.

The index should be between 0 and **GetSceneCount()** (p. 858).

Parameters

in	_index	The index of the scene.
----	--------	-------------------------

Returns

A pointer to a **Scene** (p. 879), or NULL if the index was invalid.

10.177.3.7 unsigned int gazebo::rendering::RenderEngine::GetSceneCount () const

Get the number of scenes.

Returns

The number of scenes created by the **RenderEngine** (p. 855).

10.177.3.8 WindowManagerPtr gazebo::rendering::RenderEngine::GetWindowManager () const

Get a pointer to the window manager.

Returns

Pointer to the window manager.

10.177.3.9 void gazebo::rendering::RenderEngine::Init ()

Initialize **Ogre** (p. 137). Load must happen before Init.

10.177.3.10 void gazebo::rendering::RenderEngine::Load ()

Load the parameters for **Ogre** (p. 137). Load must happen before Init.

10.177.3.11 void gazebo::rendering::RenderEngine::RemoveScene (const std::string & _name)

Remove a scene.

Parameters

in	_name	The name of the scene to remove.
----	-------	----------------------------------

10.177.4 Member Data Documentation

10.177.4.1 void* gazebo::rendering::RenderEngine::dummyContext [protected]

GLX context used to render the scenes.Used for gui-less operation.

10.177.4.2 void* gazebo::rendering::RenderEngine::dummyDisplay [protected]

Pointer to the dummy display.Used for gui-less operation.

10.177.4.3 uint64_t gazebo::rendering::RenderEngine::dummyWindowId [protected]

ID for a dummy window. Used for gui-less operation.

10.177.4.4 Ogre::Root* gazebo::rendering::RenderEngine::root

Pointer to the root scene node.

The documentation for this class was generated from the following file:

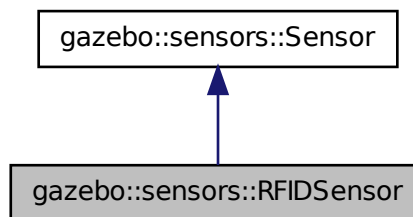
- **RenderEngine.hh**

10.178 gazebo::sensors::RFIDSensor Class Reference

Sensor (p. 907) class for RFID type of sensor.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::RFIDSensor:



Public Member Functions

- **RFIDSensor** ()
Constructor.
- virtual **~RFIDSensor** ()
Destructor.
- void **AddTag** (RFIDTag *_tag)
- virtual void **Fini** ()
Finalize the sensor.
- virtual void **Init** ()
Initialize the sensor.

- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)
Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.

Protected Member Functions

- virtual bool **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.178.1 Detailed Description

Sensor (p. 907) class for RFID type of sensor.

10.178.2 Constructor & Destructor Documentation

10.178.2.1 gazebo::sensors::RFIDSensor::RFIDSensor ()

Constructor.

10.178.2.2 virtual gazebo::sensors::RFIDSensor::~~RFIDSensor () [virtual]

Destructor.

10.178.3 Member Function Documentation

10.178.3.1 void gazebo::sensors::RFIDSensor::AddTag (RFIDTag * _tag)

10.178.3.2 virtual void gazebo::sensors::RFIDSensor::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 912).

10.178.3.3 virtual void gazebo::sensors::RFIDSensor::Init () [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.178.3.4 virtual void gazebo::sensors::RFIDSensor::Load (const std::string & _worldName, sdf::ElementPtr _sdf) [virtual]

Load the sensor with SDF parameters.

Parameters

in	<code>_sdf</code>	SDF Sensor (p. 907) parameters.
in	<code>_worldName</code>	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.178.3.5 virtual void gazebo::sensors::RFIDSensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	<code>_worldName</code>	Name of world to load from.
----	-------------------------	-----------------------------

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.178.3.6 virtual bool gazebo::sensors::RFIDSensor::UpdateImpl (bool) [protected],[virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	<code>_force</code>	True if update is forced, false if not
----	---------------------	--

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 917).

The documentation for this class was generated from the following file:

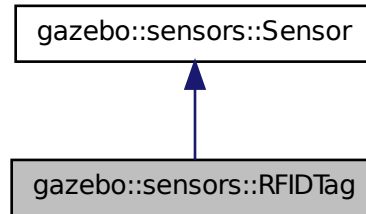
- **RFIDSensor.hh**

10.179 gazebo::sensors::RFIDTag Class Reference

RFIDTag (p. 861) to interact with RFIDTagSensors.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::RFIDTag:



Public Member Functions

- **RFIDTag** ()
Constructor.
- virtual \sim **RFIDTag** ()
Destructor.
- virtual void **Fini** ()
Finalize the sensor.
- **math::Pose GetTagPose** () const
Returns pose of tag in world coordinate.
- virtual void **Init** ()
Initialize the sensor.
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)
Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.

Protected Member Functions

- virtual bool **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.179.1 Detailed Description

RFIDTag (p. 861) to interact with RFIDTagSensors.

10.179.2 Constructor & Destructor Documentation

10.179.2.1 gazebo::sensors::RFIDTag::RFIDTag ()

Constructor.

10.179.2.2 virtual gazebo::sensors::RFIDTag::~~RFIDTag () [virtual]

Destructor.

10.179.3 Member Function Documentation

10.179.3.1 virtual void gazebo::sensors::RFIDTag::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 912).

10.179.3.2 math::Pose gazebo::sensors::RFIDTag::GetTagPose () const [inline]

Returns pose of tag in world coordinate.

Returns

Pose of object.

10.179.3.3 virtual void gazebo::sensors::RFIDTag::Init () [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.179.3.4 virtual void gazebo::sensors::RFIDTag::Load (const std::string & *_worldName*, sdf::ElementPtr *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

in	<i>_sdf</i>	SDF Sensor (p. 907) parameters.
in	<i>_worldName</i>	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

10.179.3.5 virtual void gazebo::sensors::RFIDTag::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	<i>_worldName</i>	Name of world to load from.
----	-------------------	-----------------------------

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.179.3.6 `virtual bool gazebo::sensors::RFIDTag::UpdateImpl (bool)` [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

<code>in</code>	<code>_force</code>	True if update is forced, false if not
-----------------	---------------------	--

Returns

True if the sensor was updated.

Reimplemented from `gazebo::sensors::Sensor` (p. 917).

The documentation for this class was generated from the following file:

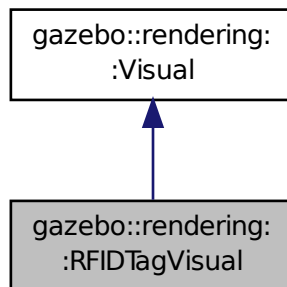
- `RFIDTag.hh`

10.180 gazebo::rendering::RFIDTagVisual Class Reference

Visualization for RFID tags sensor.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for `gazebo::rendering::RFIDTagVisual`:



Public Member Functions

- `RFIDTagVisual` (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)

Constructor.

- virtual ~**RFIDTagVisual** ()

Destructor.

Additional Inherited Members

10.180.1 Detailed Description

Visualization for RFID tags sensor.

10.180.2 Constructor & Destructor Documentation

10.180.2.1 gazebo::rendering::RFIDTagVisual::RFIDTagVisual (const std::string & *_name*, VisualPtr *_vis*, const std::string & *_topicName*)

Constructor.

Parameters

in	<i>_name</i>	Name of the visual.
in	<i>_vis</i>	Parent visual.
in	<i>_topicName</i>	Name of the topic that publishes RFID data.

See Also

- **sensors::RFIDSensor** (p. 859)

10.180.2.2 virtual gazebo::rendering::RFIDTagVisual::~~RFIDTagVisual () [virtual]

Destructor.

The documentation for this class was generated from the following file:

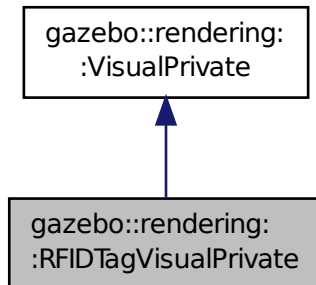
- **RFIDTagVisual.hh**

10.181 gazebo::rendering::RFIDTagVisualPrivate Class Reference

Private data for the RFID Tag **Visual** (p. 1196) class.

```
#include <RFIDTagVisualPrivate.hh>
```

Inheritance diagram for gazebo::rendering::RFIDTagVisualPrivate:



Public Attributes

- **transport::NodePtr node**
Node that handles communication.
- **transport::SubscriberPtr rfidSub**
Subscriber that receives RFID data.

Additional Inherited Members

10.181.1 Detailed Description

Private data for the RFID Tag **Visual** (p. 1196) class.

10.181.2 Member Data Documentation

10.181.2.1 transport::NodePtr gazebo::rendering::RFIDTagVisualPrivate::node

Node that handles communication.

10.181.2.2 transport::SubscriberPtr gazebo::rendering::RFIDTagVisualPrivate::rfidSub

Subscriber that receives RFID data.

The documentation for this class was generated from the following file:

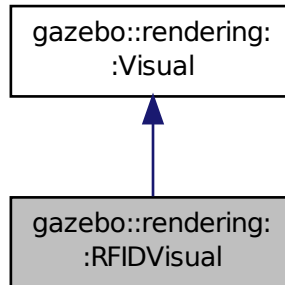
- **RFIDTagVisualPrivate.hh**

10.182 gazebo::rendering::RFIDVisual Class Reference

Visualization for RFID sensor.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::RFIDVisual:



Public Member Functions

- **RFIDVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)
Constructor.
- virtual ~**RFIDVisual** ()
Destructor.

Additional Inherited Members

10.182.1 Detailed Description

Visualization for RFID sensor.

10.182.2 Constructor & Destructor Documentation

10.182.2.1 gazebo::rendering::RFIDVisual::RFIDVisual (const std::string & *_name*, **VisualPtr** *_vis*, const std::string & *_topicName*)

Constructor.

Parameters

in	<i>_name</i>	Name of the Visual (p. 1196).
in	<i>_vis</i>	Parent Visual (p. 1196).
in	<i>_topicName</i>	Name of the topic which publishes RFID data.

10.182.2.2 virtual gazebo::rendering::RFIDVisual::~~RFIDVisual () [virtual]

Destructor.

The documentation for this class was generated from the following file:

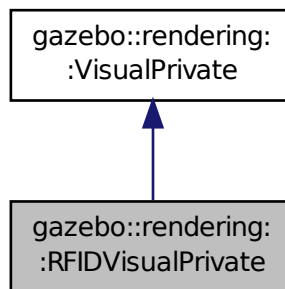
- **RFIDVisual.hh**

10.183 gazebo::rendering::RFIDVisualPrivate Class Reference

Private data for the RFID **Visual** (p. 1196) class.

```
#include <RFIDVisualPrivate.hh>
```

Inheritance diagram for gazebo::rendering::RFIDVisualPrivate:



Public Attributes

- **transport::NodePtr node**
*Pointer to the **transport::Node** (p. 731) for communication.*
- **transport::SubscriberPtr rfidSub**
*Pointer to the **transport::Subscriber** (p. 1086) for receiving data.*

Additional Inherited Members

10.183.1 Detailed Description

Private data for the RFID **Visual** (p. 1196) class.

10.183.2 Member Data Documentation

10.183.2.1 `transport::NodePtr` gazebo::rendering::RFIDVisualPrivate::node

Pointer to the `transport::Node` (p. 731) for communication.

10.183.2.2 `transport::SubscriberPtr` gazebo::rendering::RFIDVisualPrivate::rfidSub

Pointer to the `transport::Subscriber` (p. 1086) for receiving data.

The documentation for this class was generated from the following file:

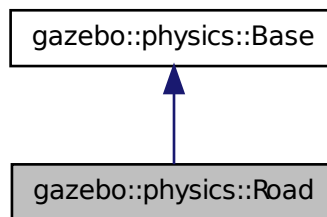
- `RFIDVisualPrivate.hh`

10.184 gazebo::physics::Road Class Reference

for building a `Road` (p. 869) from SDF

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::Road:



Public Member Functions

- `Road (BasePtr _parent)`
Constructor.
- `virtual ~Road ()`
Destructor.
- `virtual void Init ()`
Initialize the road.
- `void Load (sdf::ElementPtr _sdf)`
Load the road from SDF.

Additional Inherited Members

10.184.1 Detailed Description

for building a `Road` (p. 869) from SDF

10.184.2 Constructor & Destructor Documentation

10.184.2.1 gazebo::physics::Road::Road (BasePtr *_parent*) [explicit]

Constructor.

Parameters

<i>in</i>	<i>_parent</i>	Parent of this road object.
-----------	----------------	-----------------------------

10.184.2.2 virtual gazebo::physics::Road::~~Road () [virtual]

Destructor.

10.184.3 Member Function Documentation

10.184.3.1 virtual void gazebo::physics::Road::Init () [virtual]

Initialize the road.

Reimplemented from **gazebo::physics::Base** (p. 177).

10.184.3.2 void gazebo::physics::Road::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the road from SDF.

Parameters

<i>in</i>	<i>_sdf</i>	SDF values to load from.
-----------	-------------	--------------------------

Reimplemented from **gazebo::physics::Base** (p. 177).

The documentation for this class was generated from the following file:

- **Road.hh**

10.185 Road Class Reference

Used to render a strip of road.

```
#include <rendering/rendering.hh>
```

10.185.1 Detailed Description

Used to render a strip of road.

The documentation for this class was generated from the following file:

- **Road2d.hh**

10.186 gazebo::rendering::Road2d Class Reference

```
#include <Road2d.hh>
```

Classes

- class **Segment**
A road segment.

Public Member Functions

- **Road2d** ()
Constructor.
- virtual **~Road2d** ()
Destructor.
- void **Load** (**VisualPtr** _parent)
Load the visual using a parent visual.

10.186.1 Constructor & Destructor Documentation

10.186.1.1 gazebo::rendering::Road2d::Road2d ()

Constructor.

10.186.1.2 virtual gazebo::rendering::Road2d::~~Road2d () [virtual]

Destructor.

10.186.2 Member Function Documentation

10.186.2.1 void gazebo::rendering::Road2d::Load (**VisualPtr** _parent)

Load the visual using a parent visual.

Parameters

in	<i>_parent</i>	Pointer to the parent visual.
----	----------------	-------------------------------

The documentation for this class was generated from the following file:

- **Road2d.hh**

10.187 gazebo::math::RotationSpline Class Reference

Spline (p. 1063) for rotations.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **RotationSpline** ()
Constructor. Sets the autoCalc to true.
- **~RotationSpline** ()
Destructor. Nothing is done.
- void **AddPoint** (const **Quaternion** &_p)
Adds a control point to the end of the spline.
- void **Clear** ()
Clears all the points in the spline.
- unsigned int **GetNumPoints** () const
Gets the number of control points in the spline.
- const **Quaternion** & **GetPoint** (unsigned int _index) const
Gets the detail of one of the control points of the spline.
- **Quaternion Interpolate** (double _t, bool _useShortestPath=true)
Returns an interpolated point based on a parametric value over the whole series.
- **Quaternion Interpolate** (unsigned int _fromIndex, double _t, bool _useShortestPath=true)
Interpolates a single segment of the spline given a parametric value.
- void **RecalcTangents** ()
Recalculates the tangents associated with this spline.
- void **SetAutoCalculate** (bool _autoCalc)
Tells the spline whether it should automatically calculate tangents on demand as points are added.
- void **UpdatePoint** (unsigned int _index, const **Quaternion** &_value)
Updates a single point in the spline.

Protected Attributes

- bool **autoCalc**
Automatic recalculation of tangents when control points are updated.
- std::vector< **Quaternion** > **points**
the control points
- std::vector< **Quaternion** > **tangents**
the tangents

10.187.1 Detailed Description

Spline (p. 1063) for rotations.

10.187.2 Constructor & Destructor Documentation

10.187.2.1 gazebo::math::RotationSpline::RotationSpline ()

Constructor. Sets the autoCalc to true.

10.187.2.2 gazebo::math::RotationSpline::~~RotationSpline ()

Destructor. Nothing is done.

10.187.3 Member Function Documentation

10.187.3.1 void gazebo::math::RotationSpline::AddPoint (const Quaternion & _p)

Adds a control point to the end of the spline.

Parameters

in	_p	control point
----	----	---------------

10.187.3.2 void gazebo::math::RotationSpline::Clear ()

Clears all the points in the spline.

10.187.3.3 unsigned int gazebo::math::RotationSpline::GetNumPoints () const

Gets the number of control points in the spline.

Returns

the count

10.187.3.4 const Quaternion& gazebo::math::RotationSpline::GetPoint (unsigned int _index) const

Gets the detail of one of the control points of the spline.

Parameters

in	_index	the index of the control point.
----	--------	---------------------------------

Remarks

This point must already exist in the spline.

Returns

a quaternion (out of bound index result in assertion)

10.187.3.5 Quaternion gazebo::math::RotationSpline::Interpolate (double _t, bool _useShortestPath = true)

Returns an interpolated point based on a parametric value over the whole series.

Remarks

Given a t value between 0 and 1 representing the parametric distance along the whole length of the spline, this method returns an interpolated point.

Parameters

in	_t	Parametric value.
in	_useShortestPath	Defines if rotation should take the shortest possible path

Returns

the rotation

10.187.3.6 `Quaternion gazebo::math::RotationSpline::Interpolate (unsigned int _fromIndex, double _t, bool _useShortestPath = true)`

Interpolates a single segment of the spline given a parametric value.

Parameters

in	<i>_fromIndex</i>	The point index to treat as t = 0. <i>_fromIndex</i> + 1 is deemed to be t = 1
in	<i>_t</i>	Parametric value
in	<i>_useShortestPath</i>	Defines if rotation should take the shortest possible path

Returns

the rotation

10.187.3.7 `void gazebo::math::RotationSpline::RecalcTangents ()`

Recalculates the tangents associated with this spline.

Remarks

If you tell the spline not to update on demand by calling `setAutoCalculate(false)` then you must call this after completing your updates to the spline points.

10.187.3.8 `void gazebo::math::RotationSpline::SetAutoCalculate (bool _autoCalc)`

Tells the spline whether it should automatically calculate tangents on demand as points are added.

Remarks

The spline calculates tangents at each point automatically based on the input points. Normally it does this every time a point changes. However, if you have a lot of points to add in one go, you probably don't want to incur this overhead and would prefer to defer the calculation until you are finished setting all the points. You can do this by calling this method with a parameter of 'false'. Just remember to manually call the `recalcTangents` method when you are done.

Parameters

in	<i>_autoCalc</i>	If true, tangents are calculated for you whenever a point changes. If false, you must call <code>recalcTangents</code> to recalculate them when it best suits.
----	------------------	--

10.187.3.9 `void gazebo::math::RotationSpline::UpdatePoint (unsigned int _index, const Quaternion & _value)`

Updates a single point in the spline.

Remarks

This point must already exist in the spline.

Parameters

in	<i>_index</i>	index
in	<i>_value</i>	the new control point value

10.187.4 Member Data Documentation

10.187.4.1 `bool gazebo::math::RotationSpline::autoCalc` [protected]

Automatic recalculation of tangents when control points are updated.

10.187.4.2 `std::vector<Quaternion> gazebo::math::RotationSpline::points` [protected]

the control points

10.187.4.3 `std::vector<Quaternion> gazebo::math::RotationSpline::tangents` [protected]

the tangents

The documentation for this class was generated from the following file:

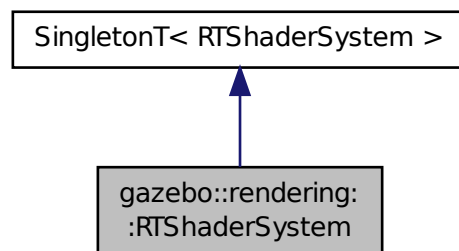
- **RotationSpline.hh**

10.188 gazebo::rendering::RTShaderSystem Class Reference

Implements **Ogre** (p. 137)'s Run-Time Shader system.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::RTShaderSystem:



Public Types

- enum **LightingModel** { **SSLM_PerVertexLighting**, **SSLM_PerPixelLighting**, **SSLM_NormalMapLighting-TangentSpace**, **SSLM_NormalMapLightingObjectSpace** }

Public Member Functions

- void **AddScene** (**ScenePtr** _scene)
Add a scene manager.
- void **ApplyShadows** (**ScenePtr** _scene)
Apply shadows to a scene.
- void **AttachEntity** (**Visual** *_vis)
Set an Ogre::Entity to use RT shaders.
- void **Clear** ()
Clear the shader system.
- void **DetachEntity** (**Visual** *_vis)
Remove and entity.
- void **Fini** ()
Finalize the shader system.
- void **GenerateShaders** (**Visual** *_vis)
Generate shaders for an entity.
- **Ogre::PSSMSHadowCameraSetup** * **GetPSSMSHadowCameraSetup** () const
*Get the **Ogre** (p. 137) PSSM Shadows camera setup.*
- void **Init** ()
Init the run time shader system.
- void **RemoveScene** (**ScenePtr** _scene)
Remove a scene.
- void **RemoveShadows** (**ScenePtr** _scene)
Remove shadows from a scene.
- void **SetPerPixelLighting** (bool _set)
Set the lighting model to per pixel or per vertex.
- void **UpdateShaders** ()
Update the shaders. This should not be called frequently.

Static Public Member Functions

- static void **AttachViewport** (**Ogre::Viewport** *_viewport, **ScenePtr** _scene)
Set a viewport to use shaders.
- static void **DetachViewport** (**Ogre::Viewport** *_viewport, **ScenePtr** _scene)
Set a viewport to not use shaders.

Additional Inherited Members

10.188.1 Detailed Description

Implements **Ogre** (p. 137)'s Run-Time Shader system.

This class allows Gazebo to generate per-pixel shaders for every material at run-time.

10.188.2 Member Enumeration Documentation

10.188.2.1 enum gazebo::rendering::RTShaderSystem::LightingModel

The type of lighting.

Enumerator:

SSLM_PerVertexLighting Per-Vertex lighting: best performance.

SSLM_PerPixelLighting Per-Pixel lighting: best look.

SSLM_NormalMapLightingTangentSpace Normal Map lighting: lighting calculations have been stored in a light map (texture) using tangent space.

SSLM_NormalMapLightingObjectSpace Normal Map lighting: lighting calculations have been stored in a light map (texture) using object space.

10.188.3 Member Function Documentation

10.188.3.1 void gazebo::rendering::RTShaderSystem::AddScene (ScenePtr _scene)

Add a scene manager.

Parameters

in	_scene	The scene to process
----	--------	----------------------

10.188.3.2 void gazebo::rendering::RTShaderSystem::ApplyShadows (ScenePtr _scene)

Apply shadows to a scene.

Parameters

in	_scene	The scene to receive shadows.
----	--------	-------------------------------

10.188.3.3 void gazebo::rendering::RTShaderSystem::AttachEntity (Visual * vis)

Set an Ogre::Entity to use RT shaders.

Parameters

in	_vis	Visual (p. 1196) that will use the RTShaderSystem (p. 875).
----	------	---

10.188.3.4 static void gazebo::rendering::RTShaderSystem::AttachViewport (Ogre::Viewport * _viewport, ScenePtr _scene) [static]

Set a viewport to use shaders.

Parameters

in	_viewport	The viewport to add.
in	_scene	The scene that the viewport uses.

10.188.3.5 void gazebo::rendering::RTShaderSystem::Clear ()

Clear the shader system.

10.188.3.6 void gazebo::rendering::RTShaderSystem::DetachEntity (Visual * _vis)

Remove and entity.

Parameters

in	_vis	Remove this visual.
----	------	---------------------

10.188.3.7 static void gazebo::rendering::RTShaderSystem::DetachViewport (Ogre::Viewport * _viewport, ScenePtr _scene)
[static]

Set a viewport to not use shaders.

Parameters

in	_viewport	The viewport to remove.
in	_scene	The scene that the viewport uses.

10.188.3.8 void gazebo::rendering::RTShaderSystem::Fini ()

Finalize the shader system.

10.188.3.9 void gazebo::rendering::RTShaderSystem::GenerateShaders (Visual * _vis)

Generate shaders for an entity.

Parameters

in	_vis	The visual to generate shaders for.
----	------	-------------------------------------

10.188.3.10 Ogre::PSSMShadowCameraSetup* gazebo::rendering::RTShaderSystem::GetPSSMShadowCameraSetup () const

Get the **Ogre** (p. 137) PSSM Shadows camera setup.

Returns

The **Ogre** (p. 137) PSSM Shadows camera setup.

10.188.3.11 void gazebo::rendering::RTShaderSystem::Init ()

Init the run time shader system.

10.188.3.12 void gazebo::rendering::RTShaderSystem::RemoveScene (ScenePtr *_scene*)

Remove a scene.

Parameters

in	<i>The</i>	scene to remove
----	------------	-----------------

10.188.3.13 void gazebo::rendering::RTShaderSystem::RemoveShadows (ScenePtr *_scene*)

Remove shadows from a scene.

Parameters

in	<i>_scene</i>	The scene to remove shadows from.
----	---------------	-----------------------------------

10.188.3.14 void gazebo::rendering::RTShaderSystem::SetPerPixelLighting (bool *_set*)

Set the lighting model to per pixel or per vertex.

Parameters

in	<i>_set</i>	True means to use per-pixel shaders.
----	-------------	--------------------------------------

10.188.3.15 void gazebo::rendering::RTShaderSystem::UpdateShaders ()

Update the shaders. This should not be called frequently.

The documentation for this class was generated from the following file:

- **RTShaderSystem.hh**

10.189 gazebo::rendering::Scene Class Reference

Representation of an entire scene graph.

```
#include <rendering/rendering.hh>
```

Public Types

- enum **SkyXMode** { **GZ_SKYX_ALL** = 0x0FFFFFFF, **GZ_SKYX_CLOUDS** = 0x0000001, **GZ_SKYX_MOON** = 0x0000002, **GZ_SKYX_NONE** = 0 }

Public Member Functions

- **Scene** (const std::string &*_name*, bool *_enableVisualizations*=false, bool *_isServer*=false)
Constructor.
- virtual ~**Scene** ()

- Destructor.*

 - void **AddVisual (VisualPtr _vis)**
Add a visual to the scene.
 - void **Clear ()**
*Clear **rendering::Scene** (p. 879).*
 - **VisualPtr CloneVisual** (const std::string &_visualName, const std::string &_newName) **GAZEBO_DEPRECATED(2.0)**
Deprecated.
 - **CameraPtr CreateCamera** (const std::string &_name, bool _autoRender=true)
Create a camera.
 - **DepthCameraPtr CreateDepthCamera** (const std::string &_name, bool _autoRender=true)
Create depth camera.
 - **GpuLaserPtr CreateGpuLaser** (const std::string &_name, bool _autoRender=true)
Create laser that generates data from rendering.
 - void **CreateGrid** (uint32_t _cellCount, float _cellLength, float _lineWidth, const **common::Color** &_color)
Create a square grid of cells.
 - **UserCameraPtr CreateUserCamera** (const std::string &_name)
Create a user camera.
 - void **DrawLine** (const **math::Vector3** &_start, const **math::Vector3** &_end, const std::string &_name)
Draw a named line.
 - **common::Color GetAmbientColor ()** const
Get the ambient color.
 - **common::Color GetBackgroundColor ()** const
Get the background color.
 - **CameraPtr GetCamera** (uint32_t _index) const
Get a camera based on an index.
 - **CameraPtr GetCamera** (const std::string &_name) const
Get a camera by name.
 - uint32_t **GetCameraCount ()** const
Get the number of cameras in this scene.
 - bool **GetFirstContact (CameraPtr _camera, const math::Vector2i &_mousePos, math::Vector3 &_position)**
Get the world pos of a the first contact at a pixel location.
 - **Grid * GetGrid** (uint32_t _index) const
Get a grid based on an index.
 - uint32_t **GetGridCount ()** const
Get the number of grids.
 - double **GetHeightBelowPoint** (const **math::Vector3** &_pt)
Get the Z-value of the first object below the given point.
 - **Heightmap * GetHeightmap ()** const
Get a pointer to the heightmap.
 - uint32_t **GetId ()** const
Get the scene ID.
 - std::string **GetIdString ()** const
Get the scene Id as a string.
 - bool **GetInitialized ()** const
*Return true if the **Scene** (p. 879) has been initialized.*
 - **LightPtr GetLight** (const std::string &_name) const

- Get a light by name.*

 - **LightPtr GetLight** (uint32_t _index) const
- Get a light based on an index.*

 - uint32_t **GetLightCount** () const
- Get the count of the lights.*

 - Ogre::SceneManager * **GetManager** () const
- Get the OGRE scene manager.*

 - **VisualPtr GetModelVisualAt** (CameraPtr _camera, const math::Vector2i & _mousePos)
- Get a model's visual at a mouse position.*

 - std::string **GetName** () const
- Get the name of the scene.*

 - **VisualPtr GetSelectedVisual** () const
- Get the currently selected visual.*

 - bool **GetShadowsEnabled** () const
- Get whether shadows are on or off.*

 - bool **GetShowClouds** () const
- Get whether or not clouds are displayed.*

 - **common::Time GetSimTime** () const
- Get the scene simulation time.*

 - **UserCameraPtr GetUserCamera** (uint32_t _index) const
- Get a user camera by index.*

 - uint32_t **GetUserCameraCount** () const
- Get the number of user cameras in this scene.*

 - **VisualPtr GetVisual** (const std::string & _name) const
- Get a visual by name.*

 - **VisualPtr GetVisual** (uint32_t _id) const
- Get a visual by id.*

 - **VisualPtr GetVisualAt** (CameraPtr _camera, const math::Vector2i & _mousePos, std::string & _mod)
- Get an entity at a pixel location using a camera.*

 - **VisualPtr GetVisualAt** (CameraPtr _camera, const math::Vector2i & _mousePos)
- Get a visual at a mouse position.*

 - **VisualPtr GetVisualBelow** (const std::string & _visualName)
- Get the closest visual below a given visual.*

 - uint32_t **GetVisualCount** () const
- Get the number of visuals.*

 - void **GetVisualsBelowPoint** (const math::Vector3 & _pt, std::vector< **VisualPtr** > & _visuals)
- Get a visual directly below a point.*

 - **VisualPtr GetWorldVisual** () const
- Get the top level world visual.*

 - void **Init** ()
- Init rendering::Scene (p. 879).*

 - void **Load** (sdf::ElementPtr _scene)
- Load the scene from a set of parameters.*

 - void **Load** ()
- Load the scene with default parameters.*

 - void **PreRender** ()
- Process all received messages.*

- void **PrintSceneGraph** ()
Print the scene graph to std_out.
- void **RemoveCamera** (const std::string &_name)
Remove a camera from the scene.
- void **RemoveProjectors** ()
Remove all projectors.
- void **RemoveVisual** (**VisualPtr** _vis)
Remove a visual from the scene.
- void **SelectVisual** (const std::string &_name, const std::string &_mode)
Select a visual by name.
- void **SetAmbientColor** (const **common::Color** &_color)
Set the ambient color.
- void **SetBackgroundColor** (const **common::Color** &_color)
Set the background color.
- void **SetFog** (const std::string &_type, const **common::Color** &_color, double _density, double _start, double _end)
Set the fog parameters.
- void **SetGrid** (bool _enabled)
Set the grid on or off.
- void **SetShadowsEnabled** (bool _value)
Set whether shadows are on or off.
- void **SetSkyXMode** (unsigned int _mode)
*Set **SkyX** (p. 137) mode to enable/disable skyx components such as clouds and moon.*
- void **SetTransparent** (bool _show)
Enable or disable transparency for all visuals.
- void **SetVisible** (const std::string &_name, bool _visible)
Hide or show a visual.
- void **SetWireframe** (bool _show)
Enable or disable wireframe for all visuals.
- void **ShowClouds** (bool _show)
Display clouds in the sky.
- void **ShowCollisions** (bool _show)
Enable or disable collision visualization.
- void **ShowCOMs** (bool _show)
Enable or disable center of mass visualization.
- void **ShowContacts** (bool _show)
Enable or disable contact visualization.
- void **ShowJoints** (bool _show)
Enable or disable joint visualization.
- void **SnapVisualToNearestBelow** (const std::string &_visualName)
Move the visual to be ontop of the nearest visual below it.
- std::string **StripSceneName** (const std::string &_name) const
Remove the name of scene from a string.

Public Attributes

- **SkyX::SkyX * skyx**
Pointer to the sky.

10.189.1 Detailed Description

Representation of an entire scene graph.

Maintains all the Visuals, Lights, and Cameras for a World.

10.189.2 Member Enumeration Documentation

10.189.2.1 enum gazebo::rendering::Scene::SkyXMode

Enumerator:

GZ_SKYX_ALL
GZ_SKYX_CLOUDS
GZ_SKYX_MOON
GZ_SKYX_NONE

10.189.3 Constructor & Destructor Documentation

10.189.3.1 gazebo::rendering::Scene::Scene (const std::string & *_name*, bool *_enableVisualizations* = false, bool *_isServer* = false)

Constructor.

Parameters

in	<i>_name</i>	Name of the scene.
in	<i>_enable-Visualizations</i>	True to enable visualizations, this should be set to true for user interfaces, and false for sensor generation.

10.189.3.2 virtual gazebo::rendering::Scene::~Scene () [virtual]

Destructor.

10.189.4 Member Function Documentation

10.189.4.1 void gazebo::rendering::Scene::AddVisual (VisualPtr *_vis*)

Add a visual to the scene.

Parameters

in	<i>_vis</i>	Visual (p. 1196) to add.
----	-------------	---------------------------------

10.189.4.2 void gazebo::rendering::Scene::Clear ()

Clear **rendering::Scene** (p. 879).

10.189.4.3 **VisualPtr** gazebo::rendering::Scene::CloneVisual (const std::string & *_visualName*, const std::string & *_newName*)

Deprecated.

10.189.4.4 **CameraPtr** gazebo::rendering::Scene::CreateCamera (const std::string & *_name*, bool *_autoRender* = true)

Create a camera.

Parameters

in	<i>_name</i>	Name of the new camera.
in	<i>_autoRender</i>	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new camera.

10.189.4.5 **DepthCameraPtr** gazebo::rendering::Scene::CreateDepthCamera (const std::string & *_name*, bool *_autoRender* = true)

Create depth camera.

Parameters

in	<i>_name</i>	Name of the new camera.
in	<i>_autoRender</i>	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new camera.

10.189.4.6 **GpuLaserPtr** gazebo::rendering::Scene::CreateGpuLaser (const std::string & *_name*, bool *_autoRender* = true)

Create laser that generates data from rendering.

Parameters

in	<i>_name</i>	Name of the new laser.
in	<i>_autoRender</i>	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new laser.

10.189.4.7 `void gazebo::rendering::Scene::CreateGrid (uint32_t _cellCount, float _cellLength, float _lineWidth, const common::Color & _color)`

Create a square grid of cells.

Parameters

in	<i>_cellCount</i>	Number of grid cells in one direction.
in	<i>_cellLength</i>	Length of one grid cell.
in	<i>_lineWidth</i>	Width of the grid lines.
in	<i>_color</i>	Color of the grid lines.

10.189.4.8 `UserCameraPtr gazebo::rendering::Scene::CreateUserCamera (const std::string & _name)`

Create a user camera.

A user camera is one design for use with a GUI.

Parameters

in	<i>_name</i>	Name of the UserCamera (p. 1137).
----	--------------	--

Returns

A pointer to the new **UserCamera** (p. 1137).

10.189.4.9 `void gazebo::rendering::Scene::DrawLine (const math::Vector3 & _start, const math::Vector3 & _end, const std::string & _name)`

Draw a named line.

Parameters

in	<i>_start</i>	Start position of the line.
in	<i>_end</i>	End position of the line.
in	<i>_name</i>	Name of the line.

10.189.4.10 `common::Color gazebo::rendering::Scene::GetAmbientColor () const`

Get the ambient color.

Returns

The scene's ambient color.

10.189.4.11 `common::Color gazebo::rendering::Scene::GetBackgroundColor () const`

Get the background color.

Returns

The background color.

10.189.4.12 `CameraPtr gazebo::rendering::Scene::GetCamera (uint32_t _index) const`

Get a camera based on an index.

Index must be between 0 and `Scene::GetCameraCount` (p. 886).

Parameters

in	<code>_index</code>	Index of the camera to get.
----	---------------------	-----------------------------

Returns

Pointer to the camera. Or NULL if the index is invalid.

10.189.4.13 `CameraPtr gazebo::rendering::Scene::GetCamera (const std::string & _name) const`

Get a camera by name.

Parameters

in	<code>_name</code>	Name of the camera.
----	--------------------	---------------------

Returns

Pointer to the camera. Or NULL if the name is invalid.

10.189.4.14 `uint32_t gazebo::rendering::Scene::GetCameraCount () const`

Get the number of cameras in this scene.

Returns

Number of lasers.

10.189.4.15 `bool gazebo::rendering::Scene::GetFirstContact (CameraPtr _camera, const math::Vector2i & _mousePos, math::Vector3 & _position)`

Get the world pos of a the first contact at a pixel location.

Parameters

in	<code>_camera</code>	Pointer to the camera.
in	<code>_mousePos</code>	2D position of the mouse in pixels.
out	<code>_position</code>	3D position of the first contact point.

Returns

True if a valid object was hit by the raycast.

10.189.4.16 Grid* gazebo::rendering::Scene::GetGrid (uint32_t *_index*) const

Get a grid based on an index.

Index must be between 0 and **Scene::GetGridCount** (p. 887).

Parameters

<i>in</i>	<i>_index</i>	Index of the grid.
-----------	---------------	--------------------

10.189.4.17 uint32_t gazebo::rendering::Scene::GetGridCount () const

Get the number of grids.

Returns

The number of grids.

10.189.4.18 double gazebo::rendering::Scene::GetHeightBelowPoint (const math::Vector3 & *_pt*)

Get the Z-value of the first object below the given point.

Parameters

<i>in</i>	<i>_pt</i>	Position to search below for a visual.
-----------	------------	--

Returns

The Z-value of the nearest visual below the point. Zero is returned if no visual is found.

10.189.4.19 Heightmap* gazebo::rendering::Scene::GetHeightmap () const

Get a pointer to the heightmap.

Returns

Pointer to the heightmap, NULL if no heightmap.

10.189.4.20 uint32_t gazebo::rendering::Scene::GetId () const

Get the scene ID.

Returns

The ID of the scene.

10.189.4.21 `std::string gazebo::rendering::Scene::GetIdString () const`

Get the scene Id as a string.

Returns

The ID as a string.

10.189.4.22 `bool gazebo::rendering::Scene::GetInitialized () const`

Return true if the **Scene** (p. 879) has been initialized.

10.189.4.23 `LightPtr gazebo::rendering::Scene::GetLight (const std::string & _name) const`

Get a light by name.

Parameters

<code>in</code>	<code>_name</code>	Name of the light to get.
-----------------	--------------------	---------------------------

Returns

Pointer to the light, or NULL if the light was not found.

10.189.4.24 `LightPtr gazebo::rendering::Scene::GetLight (uint32_t _index) const`

Get a light based on an index.

The index must be between 0 and **Scene::GetLightCount** (p. 888).

Parameters

<code>in</code>	<code>_index</code>	Index of the light.
-----------------	---------------------	---------------------

Returns

Pointer to the **Light** (p. 589) or NULL if index was invalid.

10.189.4.25 `uint32_t gazebo::rendering::Scene::GetLightCount () const`

Get the count of the lights.

Returns

The number of lights.

10.189.4.26 `Ogre::SceneManager* gazebo::rendering::Scene::GetManager () const`

Get the OGRE scene manager.

Returns

Pointer to the **Ogre** (p. 137) SceneManager.

10.189.4.27 **VisualPtr** gazebo::rendering::Scene::GetModelVisualAt (CameraPtr *_camera*, const math::Vector2i & *_mousePos*)

Get a model's visual at a mouse position.

Parameters

<i>in</i>	<i>_camera</i>	Pointer to the camera used to project the mouse position.
<i>in</i>	<i>_mousePos</i>	The 2d position of the mouse in pixels.

Returns

Pointer to the visual, NULL if none found.

10.189.4.28 **std::string** gazebo::rendering::Scene::GetName () const

Get the name of the scene.

Returns

Name of the scene.

10.189.4.29 **VisualPtr** gazebo::rendering::Scene::GetSelectedVisual () const

Get the currently selected visual.

Returns

Pointer to the currently selected visual, or NULL if nothing is selected.

10.189.4.30 **bool** gazebo::rendering::Scene::GetShadowsEnabled () const

Get whether shadows are on or off.

Returns

True if shadows are enabled.

10.189.4.31 **bool** gazebo::rendering::Scene::GetShowClouds () const

Get whether or not clouds are displayed.

Returns

True if clouds are displayed.

10.189.4.32 `common::Time gazebo::rendering::Scene::GetSimTime () const`

Get the scene simulation time.

Note this is different from `World::GetSimTime()` because there is a lag between the time new poses are sent out by `World` and when they are received and applied by the **Scene** (p. 879).

Returns

The current simulation time in **Scene** (p. 879)

10.189.4.33 `UserCameraPtr gazebo::rendering::Scene::GetUserCamera (uint32_t _index) const`

Get a user camera by index.

The index value must be between 0 and **Scene::GetUserCameraCount** (p. 890).

Parameters

<code>in</code>	<code>_index</code>	Index of the UserCamera (p. 1137) to get.
-----------------	---------------------	--

Returns

Pointer to the **UserCamera** (p. 1137), or NULL if the index was invalid.

10.189.4.34 `uint32_t gazebo::rendering::Scene::GetUserCameraCount () const`

Get the number of user cameras in this scene.

Returns

The number of user cameras.

10.189.4.35 `VisualPtr gazebo::rendering::Scene::GetVisual (const std::string & _name) const`

Get a visual by name.

Parameters

<code>in</code>	<code>_name</code>	Name of the visual to retrieve.
-----------------	--------------------	---------------------------------

Returns

Pointer to the visual, NULL if not found.

10.189.4.36 `VisualPtr gazebo::rendering::Scene::GetVisual (uint32_t _id) const`

Get a visual by id.

Parameters

in	<code>_id</code>	ID of the visual to retrieve.
----	------------------	-------------------------------

Returns

Pointer to the visual, NULL if not found.

10.189.4.37 **VisualPtr** gazebo::rendering::Scene::GetVisualAt (CameraPtr *_camera*, const math::Vector2i & *_mousePos*, std::string & *_mod*)

Get an entity at a pixel location using a camera.

Used for mouse picking.

Parameters

in	<code>_camera</code>	The ogre camera, used to do mouse picking
in	<code>_mousePos</code>	The position of the mouse in screen coordinates
out	<code>_mod</code>	Used for object manipulation

Returns

The selected entity, or NULL

10.189.4.38 **VisualPtr** gazebo::rendering::Scene::GetVisualAt (CameraPtr *_camera*, const math::Vector2i & *_mousePos*)

Get a visual at a mouse position.

Parameters

in	<code>_camera</code>	Pointer to the camera used to project the mouse position.
in	<code>_mousePos</code>	The 2d position of the mouse in pixels.

Returns

Pointer to the visual, NULL if none found.

10.189.4.39 **VisualPtr** gazebo::rendering::Scene::GetVisualBelow (const std::string & *_visualName*)

Get the closest visual below a given visual.

Parameters

in	<code>_visualName</code>	Name of the visual to search below.
----	--------------------------	-------------------------------------

Returns

Pointer to the visual below, or NULL if no visual.

10.189.4.40 `uint32_t gazebo::rendering::Scene::GetVisualCount () const`

Get the number of visuals.

Returns

The number of visuals in the **Scene** (p. 879).

10.189.4.41 `void gazebo::rendering::Scene::GetVisualsBelowPoint (const math::Vector3 & _pt, std::vector< VisualPtr > & _visuals)`

Get a visual directly below a point.

Parameters

in	<code>_pt</code>	3D point to get the visual below.
out	<code>_visuals</code>	The visuals below the point order in proximity.

10.189.4.42 `VisualPtr gazebo::rendering::Scene::GetWorldVisual () const`

Get the top level world visual.

Returns

Pointer to the world visual.

10.189.4.43 `void gazebo::rendering::Scene::Init ()`

Init **rendering::Scene** (p. 879).

10.189.4.44 `void gazebo::rendering::Scene::Load (sdf::ElementPtr _scene)`

Load the scene from a set of parameters.

Parameters

in	<code>_scene</code>	SDF scene element to load.
----	---------------------	----------------------------

10.189.4.45 `void gazebo::rendering::Scene::Load ()`

Load the scene with default parameters.

10.189.4.46 `void gazebo::rendering::Scene::PreRender ()`

Process all received messages.

10.189.4.47 void gazebo::rendering::Scene::PrintSceneGraph ()

Print the scene graph to std_out.

10.189.4.48 void gazebo::rendering::Scene::RemoveCamera (const std::string & *_name*)

Remove a camera from the scene.

Parameters

in	<i>_name</i>	Name of the camera.
----	--------------	---------------------

10.189.4.49 void gazebo::rendering::Scene::RemoveProjectors ()

Remove all projectors.

10.189.4.50 void gazebo::rendering::Scene::RemoveVisual (VisualPtr *_vis*)

Remove a visual from the scene.

Parameters

in	<i>_vis</i>	Visual (p. 1196) to remove.
----	-------------	------------------------------------

10.189.4.51 void gazebo::rendering::Scene::SelectVisual (const std::string & *_name*, const std::string & *_mode*)

Select a visual by name.

Parameters

in	<i>_name</i>	Name of the visual to select.
in	<i>_mode</i>	Selection mode (normal, or move).

10.189.4.52 void gazebo::rendering::Scene::SetAmbientColor (const common::Color & *_color*)

Set the ambient color.

Parameters

in	<i>_color</i>	The ambient color to use.
----	---------------	---------------------------

10.189.4.53 void gazebo::rendering::Scene::SetBackgroundColor (const common::Color & *_color*)

Set the background color.

Parameters

in	<i>_color</i>	The background color.
----	---------------	-----------------------

10.189.4.54 `void gazebo::rendering::Scene::SetFog (const std::string & _type, const common::Color & _color, double _density, double _start, double _end)`

Set the fog parameters.

Parameters

<code>in</code>	<code><i>_type</i></code>	Type of fog: "linear", "exp", or "exp2".
<code>in</code>	<code><i>_color</i></code>	Color of the fog.
<code>in</code>	<code><i>_density</i></code>	Fog density.
<code>in</code>	<code><i>_start</i></code>	Distance from camera to start the fog.
<code>in</code>	<code><i>_end</i></code>	Distance from camera at which the fog is at max density.

10.189.4.55 `void gazebo::rendering::Scene::SetGrid (bool _enabled)`

Set the grid on or off.

Parameters

<code>in</code>	<code><i>_enabled</i></code>	Set to true to turn on the grid
-----------------	------------------------------	---------------------------------

10.189.4.56 `void gazebo::rendering::Scene::SetShadowsEnabled (bool _value)`

Set whether shadows are on or off.

Parameters

<code>in</code>	<code><i>_value</i></code>	True to enable shadows, False to disable
-----------------	----------------------------	--

10.189.4.57 `void gazebo::rendering::Scene::SetSkyXMode (unsigned int _mode)`

Set **SkyX** (p. 137) mode to enable/disable skyx components such as clouds and moon.

Parameters

<code>in</code>	<code><i>_mode</i></code>	SkyX (p. 137) mode bitmask.
-----------------	---------------------------	------------------------------------

See Also

Scene::SkyXMode (p. 883)

10.189.4.58 `void gazebo::rendering::Scene::SetTransparent (bool _show)`

Enable or disable transparency for all visuals.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to enable transparency for all visuals.
-----------------	---------------------------	--

10.189.4.59 `void gazebo::rendering::Scene::SetVisible (const std::string & _name, bool _visible)`

Hide or show a visual.

Parameters

<code>in</code>	<code><i>_name</i></code>	Name of the visual to change.
<code>in</code>	<code><i>_visible</i></code>	True to make visual visible, False to make it invisible.

10.189.4.60 `void gazebo::rendering::Scene::SetWireframe (bool _show)`

Enable or disable wireframe for all visuals.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to enable wireframe for all visuals.
-----------------	---------------------------	---

10.189.4.61 `void gazebo::rendering::Scene::ShowClouds (bool _show)`

Display clouds in the sky.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to display clouds.
-----------------	---------------------------	-------------------------

10.189.4.62 `void gazebo::rendering::Scene::ShowCollisions (bool _show)`

Enable or disable collision visualization.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to enable collision visualization.
-----------------	---------------------------	---

10.189.4.63 `void gazebo::rendering::Scene::ShowCOMs (bool _show)`

Enable or disable center of mass visualization.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to enable center of mass visualization.
-----------------	---------------------------	--

10.189.4.64 `void gazebo::rendering::Scene::ShowContacts (bool _show)`

Enable or disable contact visualization.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to enable contact visualization.
-----------------	---------------------------	---------------------------------------

10.189.4.65 `void gazebo::rendering::Scene::ShowJoints (bool _show)`

Enable or disable joint visualization.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to enable joint visualization.
-----------------	---------------------------	-------------------------------------

10.189.4.66 `void gazebo::rendering::Scene::SnapVisualToNearestBelow (const std::string & _visualName)`

Move the visual to be ontop of the nearest visual below it.

Parameters

<code>in</code>	<code><i>_visualName</i></code>	Name of the visual to move.
-----------------	---------------------------------	-----------------------------

10.189.4.67 `std::string gazebo::rendering::Scene::StripSceneName (const std::string & _name) const`

Remove the name of scene from a string.

Parameters

<code>in</code>	<code><i>_name</i></code>	Name to string the scene name from.
-----------------	---------------------------	-------------------------------------

Returns

The stripped name.

10.189.5 Member Data Documentation

10.189.5.1 `SkyX::SkyX* gazebo::rendering::Scene::skyx`

Pointer to the sky.

The documentation for this class was generated from the following file:

- **Scene.hh**

10.190 gazebo::physics::ScrewJoint< T > Class Template Reference

A screw joint, which has both prismatic and rotational DOFs.

```
#include <physics/physics.hh>
```

Public Member Functions

- **ScrewJoint (BasePtr *_parent*)**
Constructor.
- virtual **~ScrewJoint ()**

Destructor.

- virtual unsigned int **GetAngleCount** () const
- virtual double **GetThreadPitch** (unsigned int _index) **GAZEBO_DEPRECATED(3.0)=0**
Get screw joint thread pitch.
- virtual double **GetThreadPitch** ()=0
Get screw joint thread pitch.
- virtual void **Load** (sdf::ElementPtr _sdf)
*Load a **ScrewJoint** (p. 896).*
- virtual void **SetThreadPitch** (unsigned int _index, double _threadPitch) **GAZEBO_DEPRECATED(3.0)=0**
Set screw joint thread pitch.
- virtual void **SetThreadPitch** (double _threadPitch)=0
Set screw joint thread pitch.

Protected Member Functions

- virtual void **Init** ()
Initialize joint.

Protected Attributes

- double **threadPitch**
Pitch of the thread.

10.190.1 Detailed Description

```
template<class T>class gazebo::physics::ScrewJoint< T >
```

A screw joint, which has both prismatic and rotational DOFs.

10.190.2 Constructor & Destructor Documentation

10.190.2.1 `template<class T> gazebo::physics::ScrewJoint< T >::ScrewJoint (BasePtr _parent) [inline], [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Parent of the joint.
----	----------------------	----------------------

10.190.2.2 `template<class T> virtual gazebo::physics::ScrewJoint< T >::~ScrewJoint () [inline], [virtual]`

Destructor.

10.190.3 Member Function Documentation

10.190.3.1 `template<class T> virtual unsigned int gazebo::physics::ScrewJoint< T >::GetAngleCount () const`
`[inline],[virtual]`

10.190.3.2 `template<class T> virtual double gazebo::physics::ScrewJoint< T >::GetThreadPitch (unsigned int _index)`
`[pure virtual]`

Get screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

_threadPitch Thread pitch value.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 1006), and **gazebo::physics::DARTScrewJoint** (p. 368).

10.190.3.3 `template<class T> virtual double gazebo::physics::ScrewJoint< T >::GetThreadPitch ()` `[pure virtual]`

Get screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class

Returns

_threadPitch Thread pitch value.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 1006), and **gazebo::physics::DARTScrewJoint** (p. 368).

10.190.3.4 `template<class T> virtual void gazebo::physics::ScrewJoint< T >::Init ()` `[inline],[protected],[virtual]`

Initialize joint.

Reimplemented in **gazebo::physics::DARTScrewJoint** (p. 369).

10.190.3.5 `template<class T> virtual void gazebo::physics::ScrewJoint< T >::Load (sdf::ElementPtr _sdf)` `[inline],[virtual]`

Load a **ScrewJoint** (p. 896).

Parameters

in	_sdf	SDF value to load from
----	------	------------------------

Reimplemented in **gazebo::physics::SimbodyScrewJoint** (p.1007), and **gazebo::physics::DARTScrewJoint** (p. 369).

10.190.3.6 `template<class T> virtual void gazebo::physics::ScrewJoint< T >::SetThreadPitch (unsigned int _index, double _threadPitch) [pure virtual]`

Set screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class. **Deprecated**, please use the index-less version in the future: `virtual void SetThreadPitch(double _threadPitch)` (p. 899) = 0;

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_threadPitch</i></code>	Thread pitch value.

Implemented in `gazebo::physics::SimbodyScrewJoint` (p. 1009), and `gazebo::physics::DARTScrewJoint` (p. 370).

10.190.3.7 `template<class T> virtual void gazebo::physics::ScrewJoint< T >::SetThreadPitch (double _threadPitch) [pure virtual]`

Set screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class. To clarify direction, these are modeling right handed threads with positive `thread_pitch`, i.e. the child **Link** (p. 595) is the nut (interior threads) while the parent **Link** (p. 595) is the bolt/screw (exterior threads).

Parameters

<code>in</code>	<code><i>_threadPitch</i></code>	Thread pitch value.
-----------------	----------------------------------	---------------------

Implemented in `gazebo::physics::SimbodyScrewJoint` (p. 1009), and `gazebo::physics::DARTScrewJoint` (p. 370).

10.190.4 Member Data Documentation

10.190.4.1 `template<class T> double gazebo::physics::ScrewJoint< T >::threadPitch [protected]`

Pitch of the thread.

The documentation for this class was generated from the following file:

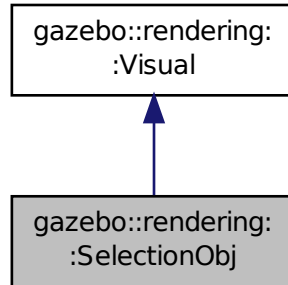
- **ScrewJoint.hh**

10.191 gazebo::rendering::SelectionObj Class Reference

Interactive selection object for models and links.

```
#include <SelectionObj.hh>
```

Inheritance diagram for gazebo::rendering::SelectionObj:



Public Types

- enum **SelectionMode** {
SELECTION_NONE = 0, **TRANS**, **ROT**, **SCALE**,
TRANS_X, **TRANS_Y**, **TRANS_Z**, **ROT_X**,
ROT_Y, **ROT_Z**, **SCALE_X**, **SCALE_Y**,
SCALE_Z }

Public Member Functions

- **SelectionObj** (const std::string &_name, **VisualPtr** _vis)
Constructor.
- virtual ~**SelectionObj** ()
Destructor.
- void **Attach** (**rendering::VisualPtr** _vis)
Attach the selection object to the given visual.
- void **Detach** ()
Detach the selection object from the current visual.
- **SelectionMode** **GetMode** ()
Get the current selection mode.
- **SelectionMode** **GetState** ()
Get the current selection state.
- void **Load** ()
Load.
- void **SetGlobal** (bool _global)
Set selection object to ignore local transforms.
- void **SetMode** (const std::string &_mode)
Set the manipulation mode.
- void **SetMode** (**SelectionMode** _mode)
Set the selection mode.

- void **SetState** (const std::string &_state)
Set state by highlighting the corresponding selection object visual.
- void **SetState** (**SelectionMode** _state)
Set state by highlighting the corresponding selection object visual.
- void **UpdateSize** ()
Update selection object size to match the parent visual.

Additional Inherited Members

10.191.1 Detailed Description

Interactive selection object for models and links.

10.191.2 Member Enumeration Documentation

10.191.2.1 enum gazebo::rendering::SelectionObj::SelectionMode

Enumerator:

SELECTION_NONE Translation in x.

TRANS Translation mode.

ROT Rotation mode.

SCALE Scale mode.

TRANS_X Translation in x.

TRANS_Y Translation in y.

TRANS_Z Translation in z.

ROT_X Rotation in x.

ROT_Y Rotation in y.

ROT_Z Rotation in z.

SCALE_X Scale in x.

SCALE_Y Scale in y.

SCALE_Z Scale in z.

10.191.3 Constructor & Destructor Documentation

10.191.3.1 gazebo::rendering::SelectionObj::SelectionObj (const std::string & _name, VisualPtr _vis)

Constructor.

Parameters

in	<code>_name</code>	Name of selection object.
in	<code>_vis</code>	Parent visual that the selection object is attached to.

10.191.3.2 `virtual gazebo::rendering::SelectionObj::~SelectionObj () [virtual]`

Destructor.

10.191.4 Member Function Documentation

10.191.4.1 `void gazebo::rendering::SelectionObj::Attach (rendering::VisualPtr _vis)`

Attach the selection object to the given visual.

Parameters

<code>in</code>	<code>_vis</code>	Pointer to visual to which the selection object will be attached.
-----------------	-------------------	---

10.191.4.2 `void gazebo::rendering::SelectionObj::Detach ()`

Detach the selection object from the current visual.

10.191.4.3 `SelectionMode gazebo::rendering::SelectionObj::GetMode ()`

Get the current selection mode.

10.191.4.4 `SelectionMode gazebo::rendering::SelectionObj::GetState ()`

Get the current selection state.

10.191.4.5 `void gazebo::rendering::SelectionObj::Load () [virtual]`

Load.

Reimplemented from `gazebo::rendering::Visual` (p. 1210).

10.191.4.6 `void gazebo::rendering::SelectionObj::SetGlobal (bool _global)`

Set selection object to ignore local transforms.

Parameters

<code>in</code>	<code>_global</code>	True to set the visuals to be in global frame.
-----------------	----------------------	--

10.191.4.7 `void gazebo::rendering::SelectionObj::SetMode (const std::string & _mode)`

Set the manipulation mode.

Parameters

<code>in</code>	<code>_mode</code>	Manipulation mode in string: translate rotate, scale.
-----------------	--------------------	---

10.191.4.8 void gazebo::rendering::SelectionObj::SetMode (SelectionMode *_mode*)

Set the selection mode.

_name Selection mode: TRANS, ROT, SCALE.

10.191.4.9 void gazebo::rendering::SelectionObj::SetState (const std::string & *_state*)

Set state by highlighting the corresponding selection object visual.

Parameters

<i>in</i>	<i>_state</i>	Selection state in string format.
-----------	---------------	-----------------------------------

10.191.4.10 void gazebo::rendering::SelectionObj::SetState (SelectionMode *_state*)

Set state by highlighting the corresponding selection object visual.

Parameters

<i>in</i>	<i>_state</i>	Selection state.
-----------	---------------	------------------

See Also

SelectionMode (p. 901)

10.191.4.11 void gazebo::rendering::SelectionObj::UpdateSize ()

Update selection object size to match the parent visual.

The documentation for this class was generated from the following file:

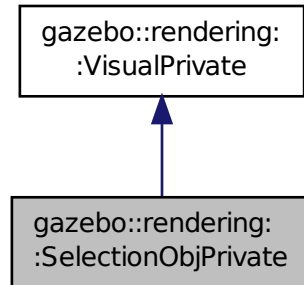
- **SelectionObj.hh**

10.192 gazebo::rendering::SelectionObjPrivate Class Reference

Private data for the Selection Obj class.

```
#include <SelectionObjPrivate.hh>
```

Inheritance diagram for gazebo::rendering::SelectionObjPrivate:



Public Attributes

- double **maxScale**
Maximum scale of the selection object visual.
- double **minScale**
Minimum scale of the selection object visual.
- **SelectionObj::SelectionMode mode**
Current manipulation mode.
- **VisualPtr rotVisual**
Rotation visual.
- **VisualPtr rotXVisual**
X rotation visual.
- **VisualPtr rotYVisual**
Y rotation visual.
- **VisualPtr rotZVisual**
Z rotation visual.
- **VisualPtr scaleVisual**
Scale visual.
- **VisualPtr scaleXVisual**
X scale visual.
- **VisualPtr scaleYVisual**
Y Scale visual.
- **VisualPtr scaleZVisual**
Z scale visual.
- **VisualPtr selectedVis**
Pointer to visual that is currently selected.
- **SelectionObj::SelectionMode state**
Current selection state.
- **VisualPtr transVisual**

Translation visual.

- **VisualPtr transXVisual**

X translation visual.

- **VisualPtr transYVisual**

Y translation visual.

- **VisualPtr transZVisual**

Z translation visual.

- **std::string xAxisMat**

Material name for the x axis.

- **std::string xAxisMatOverlay**

Overlay material name for the x axis.

- **std::string yAxisMat**

Material name for the y axis.

- **std::string yAxisMatOverlay**

Overlay material name for the y axis.

- **std::string zAxisMat**

Material name for the z axis.

- **std::string zAxisMatOverlay**

Overlay material name for the z axis.

Additional Inherited Members

10.192.1 Detailed Description

Private data for the Selection Obj class.

10.192.2 Member Data Documentation

10.192.2.1 double gazebo::rendering::SelectionObjPrivate::maxScale

Maximum scale of the selection object visual.

10.192.2.2 double gazebo::rendering::SelectionObjPrivate::minScale

Minimum scale of the selection object visual.

10.192.2.3 SelectionObj::SelectionMode gazebo::rendering::SelectionObjPrivate::mode

Current manipulation mode.

10.192.2.4 VisualPtr gazebo::rendering::SelectionObjPrivate::rotVisual

Rotation visual.

10.192.2.5 VisualPtr gazebo::rendering::SelectionObjPrivate::rotXVisual

X rotation visual.

10.192.2.6 **VisualPtr gazebo::rendering::SelectionObjPrivate::rotYVisual**

Y rotation visual.

10.192.2.7 **VisualPtr gazebo::rendering::SelectionObjPrivate::rotZVisual**

Z rotation visual.

10.192.2.8 **VisualPtr gazebo::rendering::SelectionObjPrivate::scaleVisual**

Scale visual.

10.192.2.9 **VisualPtr gazebo::rendering::SelectionObjPrivate::scaleXVisual**

X scale visual.

10.192.2.10 **VisualPtr gazebo::rendering::SelectionObjPrivate::scaleYVisual**

Y Scale visual.

10.192.2.11 **VisualPtr gazebo::rendering::SelectionObjPrivate::scaleZVisual**

Z scale visual.

10.192.2.12 **VisualPtr gazebo::rendering::SelectionObjPrivate::selectedVis**

Pointer to visual that is currently selected.

10.192.2.13 **SelectionObj::SelectionMode gazebo::rendering::SelectionObjPrivate::state**

Current selection state.

10.192.2.14 **VisualPtr gazebo::rendering::SelectionObjPrivate::transVisual**

Translation visual.

10.192.2.15 **VisualPtr gazebo::rendering::SelectionObjPrivate::transXVisual**

X translation visual.

10.192.2.16 **VisualPtr gazebo::rendering::SelectionObjPrivate::transYVisual**

Y translation visual.

10.192.2.17 `VisualPtr gazebo::rendering::SelectionObjPrivate::transZVisual`

Z translation visual.

10.192.2.18 `std::string gazebo::rendering::SelectionObjPrivate::xAxisMat`

Material name for the x axis.

10.192.2.19 `std::string gazebo::rendering::SelectionObjPrivate::xAxisMatOverlay`

Overlay material name for the x axis.

10.192.2.20 `std::string gazebo::rendering::SelectionObjPrivate::yAxisMat`

Material name for the y axis.

10.192.2.21 `std::string gazebo::rendering::SelectionObjPrivate::yAxisMatOverlay`

Overlay material name for the y axis.

10.192.2.22 `std::string gazebo::rendering::SelectionObjPrivate::zAxisMat`

Material name for the z axis.

10.192.2.23 `std::string gazebo::rendering::SelectionObjPrivate::zAxisMatOverlay`

Overlay material name for the z axis.

The documentation for this class was generated from the following file:

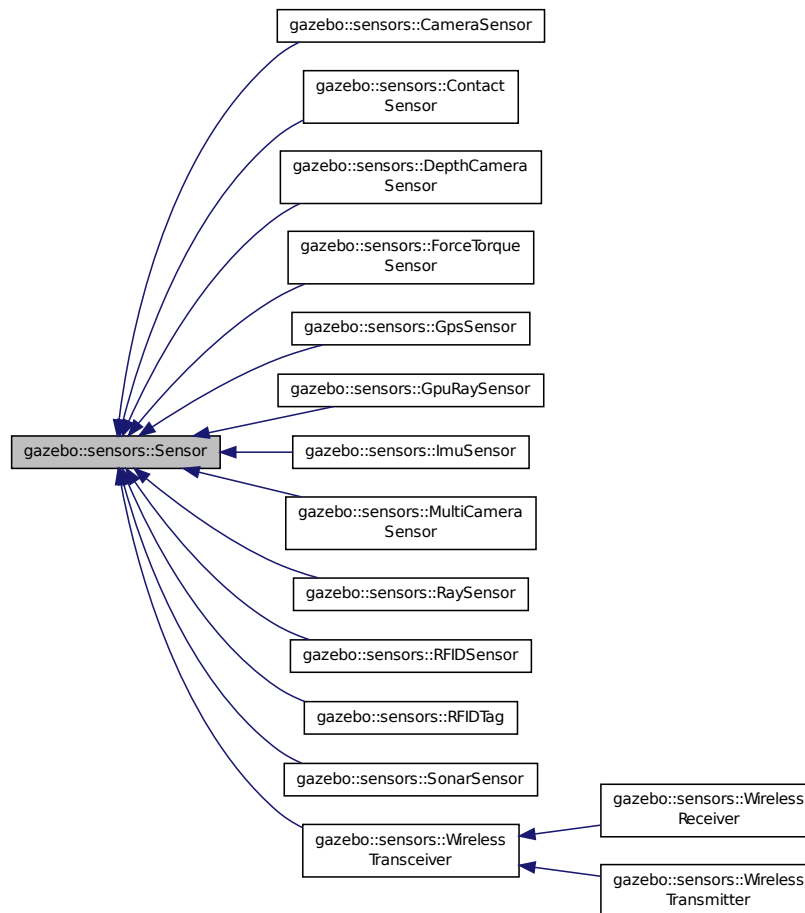
- **SelectionObjPrivate.hh**

10.193 gazebo::sensors::Sensor Class Reference

Base class for sensors.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::Sensor:



Public Member Functions

- **Sensor** (**SensorCategory** _cat)
Constructor.
- virtual **~Sensor** ()
Destructor.
- template<typename T >
event::ConnectionPtr ConnectUpdated (T _subscriber)
Connect a signal that is triggered when the sensor is updated.
- void **DisconnectUpdated** (**event::ConnectionPtr** &_c)
Disconnect from a the updated signal.
- void **FillMsg** (msgs::Sensor &_msg)
fills a msgs::Sensor message.
- virtual void **Fini** ()
Finalize the sensor.

- **SensorCategory GetCategory** () const
Get the category of the sensor.
- **uint32_t GetId** () const
Get the sensor's ID.
- **common::Time GetLastMeasurementTime** ()
Return last measurement time.
- **common::Time GetLastUpdateTime** ()
Return last update time.
- **std::string GetName** () const
Get name.
- **NoisePtr GetNoise** (unsigned int _index=0) const
Get the sensor's noise model.
- **uint32_t GetParentId** () const
Get the sensor's parent's ID.
- **std::string GetParentName** () const
Returns the name of the sensor parent.
- **virtual math::Pose GetPose** () const
Get the current pose.
- **std::string GetScopedName** () const
Get fully scoped name of the sensor.
- **virtual std::string GetTopic** () const
Returns the topic name as set in SDF.
- **std::string GetType** () const
Get sensor type.
- **double GetUpdateRate** ()
Get the update rate of the sensor.
- **bool GetVisualize** () const
Return true if user requests the sensor to be visualized via tag: <visualize>true</visualize> in SDF.
- **std::string GetWorldName** () const
Returns the name of the world the sensor is in.
- **virtual void Init** ()
Initialize the sensor.
- **virtual bool IsActive** ()
Returns true if sensor generation is active.
- **virtual void Load** (const std::string &_worldName, sdf::ElementPtr _sdf)
Load the sensor with SDF parameters.
- **virtual void Load** (const std::string &_worldName)
Load the sensor with default parameters.
- **void ResetLastUpdateTime** ()
Reset the lastUpdateTime to zero.
- **virtual void SetActive** (bool _value)
Set whether the sensor is active or not.
- **virtual void SetParent** (const std::string &_name) **GAZEBO_DEPRECATED(2.0)**
Set the parent of the sensor.
- **void SetParent** (const std::string &_name, uint32_t _id)
Set the sensor's parent.
- **void SetUpdateRate** (double _hz)
Set the update rate of the sensor.
- **void Update** (bool _force)
Update the sensor.

Protected Member Functions

- bool **NeedsUpdate** ()
Return true if the sensor needs to be updated.
- virtual bool **UpdateImpl** (bool)
This gets overwritten by derived sensor types.

Protected Attributes

- bool **active**
True if sensor generation is active.
- std::vector< **event::ConnectionPtr** > **connections**
All event connections.
- **common::Time lastMeasurementTime**
Stores last time that a sensor measurement was generated; this value must be updated within each sensor's UpdateImpl.
- **common::Time lastUpdateTime**
Time of the last update.
- **transport::NodePtr node**
Node for communication.
- std::vector< **NoisePtr** > **noises**
Noise (p. 748) added to sensor data.
- uint32_t **parentId**
The sensor's parent ID.
- std::string **parentName**
Name of the parent.
- std::vector< **SensorPluginPtr** > **plugins**
All the plugins for the sensor.
- **math::Pose pose**
Pose of the sensor.
- **transport::SubscriberPtr poseSub**
Subscribe to pose updates.
- **gazebo::rendering::ScenePtr scene**
Pointer to the Scene.
- sdf::ElementPtr **sdf**
Pointer the the SDF element for the sensor.
- **common::Time updatePeriod**
*Desired time between updates, set indirectly by **Sensor::SetUpdateRate** (p. 916).*
- **gazebo::physics::WorldPtr world**
Pointer to the world.

10.193.1 Detailed Description

Base class for sensors.

10.193.2 Constructor & Destructor Documentation

10.193.2.1 `gazebo::sensors::Sensor::Sensor (SensorCategory _cat)` `[explicit]`

Constructor.

Parameters

in	_class	
----	--------	--

10.193.2.2 `virtual gazebo::sensors::Sensor::~~Sensor ()` `[virtual]`

Destructor.

10.193.3 Member Function Documentation

10.193.3.1 `template<typename T > event::ConnectionPtr gazebo::sensors::Sensor::ConnectUpdated (T _subscriber)`
`[inline]`

Connect a signal that is triggered when the sensor is updated.

Parameters

in	_subscriber	Callback that receives the signal.
----	-------------	------------------------------------

Returns

A pointer to the connection. This must be kept in scope.

See Also

Sensor::DisconnectUpdated (p. 911)

10.193.3.2 `void gazebo::sensors::Sensor::DisconnectUpdated (event::ConnectionPtr & _c)` `[inline]`

Disconnect from a the updated signal.

Parameters

in	_c	The connection to disconnect
----	----	------------------------------

See Also

Sensor::ConnectUpdated (p. 911)

10.193.3.3 `void gazebo::sensors::Sensor::FillMsg (msgs::Sensor & _msg)`

fills a msgs::Sensor message.

Parameters

out	<code>_msg</code>	Message to fill.
-----	-------------------	------------------

10.193.3.4 virtual void gazebo::sensors::Sensor::Fini () [virtual]

Finalize the sensor.

Reimplemented in [gazebo::sensors::MultiCameraSensor](#) (p. 720), [gazebo::sensors::ForceTorqueSensor](#) (p. 451), [gazebo::sensors::GpuRaySensor](#) (p. 480), [gazebo::sensors::RFIDSensor](#) (p. 860), [gazebo::sensors::CameraSensor](#) (p. 229), [gazebo::sensors::ContactSensor](#) (p. 289), [gazebo::sensors::DepthCameraSensor](#) (p. 388), [gazebo::sensors::RaySensor](#) (p. 844), [gazebo::sensors::RFIDTag](#) (p. 863), [gazebo::sensors::GpsSensor](#) (p. 465), [gazebo::sensors::SonarSensor](#) (p. 1049), [gazebo::sensors::ImuSensor](#) (p. 527), [gazebo::sensors::WirelessTransceiver](#) (p. 1234), and [gazebo::sensors::WirelessReceiver](#) (p. 1231).

10.193.3.5 SensorCategory gazebo::sensors::Sensor::GetCategory () const

Get the category of the sensor.

Returns

The category of the sensor.

See Also

[SensorCategory](#) (p. 133)

10.193.3.6 uint32_t gazebo::sensors::Sensor::GetId () const

Get the sensor's ID.

Returns

The sensor's ID.

10.193.3.7 common::Time gazebo::sensors::Sensor::GetLastMeasurementTime ()

Return last measurement time.

Returns

Time of last measurement.

10.193.3.8 common::Time gazebo::sensors::Sensor::GetLastUpdateTime ()

Return last update time.

Returns

Time of last update.

10.193.3.9 `std::string gazebo::sensors::Sensor::GetName () const`

Get name.

Returns

Name of sensor.

10.193.3.10 `NoisePtr gazebo::sensors::Sensor::GetNoise (unsigned int _index = 0) const`

Get the sensor's noise model.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the noise model. For most sensors this will be 0. For a multi camera sensor the index can be ≥ 0 .
-----------------	----------------------------	--

Returns

The sensor's noise model.

10.193.3.11 `uint32_t gazebo::sensors::Sensor::GetParentId () const`

Get the sensor's parent's ID.

Returns

The sensor's parent's ID.

10.193.3.12 `std::string gazebo::sensors::Sensor::GetParentName () const`

Returns the name of the sensor parent.

The parent name is set by **Sensor::SetParent** (p. 916).

Returns

Name of Parent.

10.193.3.13 `virtual math::Pose gazebo::sensors::Sensor::GetPose () const` `[virtual]`

Get the current pose.

Returns

Current pose of the sensor.

10.193.3.14 `std::string gazebo::sensors::Sensor::GetScopedName () const`

Get fully scoped name of the sensor.

Returns

world_name::model_name::link_name::sensor_name.

10.193.3.15 `virtual std::string gazebo::sensors::Sensor::GetTopic () const` [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented in **`gazebo::sensors::GpuRaySensor`** (p. 484), **`gazebo::sensors::RaySensor`** (p. 847), **`gazebo::sensors::CameraSensor`** (p. 230), **`gazebo::sensors::SonarSensor`** (p. 1050), **`gazebo::sensors::MultiCameraSensor`** (p. 722), **`gazebo::sensors::ForceTorqueSensor`** (p. 451), and **`gazebo::sensors::WirelessTransceiver`** (p. 1234).

10.193.3.16 `std::string gazebo::sensors::Sensor::GetType () const`

Get sensor type.

Returns

Type of sensor.

10.193.3.17 `double gazebo::sensors::Sensor::GetUpdateRate ()`

Get the update rate of the sensor.

Returns

_hz update rate of sensor. Returns 0 if unthrottled.

10.193.3.18 `bool gazebo::sensors::Sensor::GetVisualize () const`

Return true if user requests the sensor to be visualized via tag: `<visualize>true</visualize>` in SDF.

Returns

True if visualized, false if not.

10.193.3.19 `std::string gazebo::sensors::Sensor::GetWorldName () const`

Returns the name of the world the sensor is in.

Returns

Name of the world.

10.193.3.20 `virtual void gazebo::sensors::Sensor::Init () [virtual]`

Initialize the sensor.

Reimplemented in `gazebo::sensors::GpuRaySensor` (p. 485), `gazebo::sensors::ContactSensor` (p. 291), `gazebo::sensors::RFIDSensor` (p. 860), `gazebo::sensors::CameraSensor` (p. 230), `gazebo::sensors::DepthCameraSensor` (p. 389), `gazebo::sensors::WirelessTransmitter` (p. 1238), `gazebo::sensors::RaySensor` (p. 848), `gazebo::sensors::RFIDTag` (p. 863), `gazebo::sensors::GpsSensor` (p. 466), `gazebo::sensors::SonarSensor` (p. 1050), `gazebo::sensors::MultiCameraSensor` (p. 722), `gazebo::sensors::ImuSensor` (p. 528), `gazebo::sensors::WirelessTransceiver` (p. 1234), `gazebo::sensors::ForceTorqueSensor` (p. 452), and `gazebo::sensors::WirelessReceiver` (p. 1232).

10.193.3.21 `virtual bool gazebo::sensors::Sensor::IsActive () [virtual]`

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented in `gazebo::sensors::GpuRaySensor` (p. 485), `gazebo::sensors::RaySensor` (p. 848), `gazebo::sensors::ContactSensor` (p. 291), `gazebo::sensors::CameraSensor` (p. 230), `gazebo::sensors::MultiCameraSensor` (p. 722), `gazebo::sensors::SonarSensor` (p. 1050), `gazebo::sensors::ImuSensor` (p. 528), and `gazebo::sensors::ForceTorqueSensor` (p. 452).

10.193.3.22 `virtual void gazebo::sensors::Sensor::Load (const std::string & _worldName, sdf::ElementPtr _sdf) [virtual]`

Load the sensor with SDF parameters.

Parameters

in	<code>_sdf</code>	SDF Sensor (p. 907) parameters.
in	<code>_worldName</code>	Name of world to load from.

Reimplemented in `gazebo::sensors::GpuRaySensor` (p. 486), `gazebo::sensors::ContactSensor` (p. 291), `gazebo::sensors::RFIDSensor` (p. 860), `gazebo::sensors::CameraSensor` (p. 230), `gazebo::sensors::DepthCameraSensor` (p. 389), `gazebo::sensors::RFIDTag` (p. 863), `gazebo::sensors::GpsSensor` (p. 466), and `gazebo::sensors::ImuSensor` (p. 528).

10.193.3.23 `virtual void gazebo::sensors::Sensor::Load (const std::string & _worldName) [virtual]`

Load the sensor with default parameters.

Parameters

in	<code>_worldName</code>	Name of world to load from.
----	-------------------------	-----------------------------

Reimplemented in `gazebo::sensors::GpuRaySensor` (p. 486), `gazebo::sensors::ContactSensor` (p. 291), `gazebo::sensors::RFIDSensor` (p. 861), `gazebo::sensors::CameraSensor` (p. 231), `gazebo::sensors::DepthCameraSensor` (p. 389), `gazebo::sensors::WirelessTransmitter` (p. 1238), `gazebo::sensors::RaySensor` (p. 848), `gazebo::sensors::RFIDTag` (p. 863), `gazebo::sensors::GpsSensor` (p. 466), `gazebo::sensors::SonarSensor` (p. 1050), `gazebo::sensors::MultiCameraSensor` (p. 722), `gazebo::sensors::ImuSensor` (p. 528), `gazebo::sensors::WirelessTransceiver` (p. 1234), `gazebo::sensors::ForceTorqueSensor` (p. 452), and `gazebo::sensors::Wireless-`

Receiver (p. 1232).

10.193.3.24 `bool gazebo::sensors::Sensor::NeedsUpdate ()` [protected]

Return true if the sensor needs to be updated.

Returns

True when sensor should be updated.

10.193.3.25 `void gazebo::sensors::Sensor::ResetLastUpdateTime ()`

Reset the lastUpdateTime to zero.

10.193.3.26 `virtual void gazebo::sensors::Sensor::SetActive (bool _value)` [virtual]

Set whether the sensor is active or not.

Parameters

in	<i>_value</i>	True if active, false if not.
----	---------------	-------------------------------

Reimplemented in `gazebo::sensors::DepthCameraSensor` (p. 390).

10.193.3.27 `virtual void gazebo::sensors::Sensor::SetParent (const std::string & _name)` [virtual]

Set the parent of the sensor.

Parameters

in	<i>_name</i>	Name of the parent.
----	--------------	---------------------

10.193.3.28 `void gazebo::sensors::Sensor::SetParent (const std::string & _name, uint32_t _id)`

Set the sensor's parent.

Parameters

in	<i>_name</i>	The sensor's parent's name.
in	<i>_id</i>	The sensor's parent's ID.

10.193.3.29 `void gazebo::sensors::Sensor::SetUpdateRate (double _hz)`

Set the update rate of the sensor.

Parameters

in	<i>_hz</i>	update rate of sensor.
----	------------	------------------------

10.193.3.30 `void gazebo::sensors::Sensor::Update (bool _force)`

Update the sensor.

Parameters

<code>in</code>	<code><i>_force</i></code>	True to force update, false otherwise.
-----------------	----------------------------	--

10.193.3.31 `virtual bool gazebo::sensors::Sensor::UpdateImpl (bool)` `[inline]`, `[protected]`, `[virtual]`

This gets overwritten by derived sensor types.

This function is called during `Sensor::Update`.
And in turn, `Sensor::Update` is called by
`SensorManager::Update`

Parameters

<code>in</code>	<code><i>_force</i></code>	True if update is forced, false if not
-----------------	----------------------------	--

Returns

True if the sensor was updated.

Reimplemented in [gazebo::sensors::MultiCameraSensor](#) (p. 723), [gazebo::sensors::ForceTorqueSensor](#) (p. 452), [gazebo::sensors::GpuRaySensor](#) (p. 487), [gazebo::sensors::RFIDSensor](#) (p. 861), [gazebo::sensors::CameraSensor](#) (p. 231), [gazebo::sensors::ContactSensor](#) (p. 291), [gazebo::sensors::DepthCameraSensor](#) (p. 390), [gazebo::sensors::RaySensor](#) (p. 849), [gazebo::sensors::RFIDTag](#) (p. 864), [gazebo::sensors::GpsSensor](#) (p. 466), [gazebo::sensors::SonarSensor](#) (p. 1050), [gazebo::sensors::WirelessTransmitter](#) (p. 1238), and [gazebo::sensors::ImuSensor](#) (p. 529).

10.193.4 Member Data Documentation

10.193.4.1 `bool gazebo::sensors::Sensor::active` `[protected]`

True if sensor generation is active.

10.193.4.2 `std::vector<event::ConnectionPtr> gazebo::sensors::Sensor::connections` `[protected]`

All event connections.

10.193.4.3 `common::Time gazebo::sensors::Sensor::lastMeasurementTime` `[protected]`

Stores last time that a sensor measurement was generated; this value must be updated within each sensor's `UpdateImpl`.

10.193.4.4 `common::Time gazebo::sensors::Sensor::lastUpdateTime` `[protected]`

Time of the last update.

10.193.4.5 `transport::NodePtr gazebo::sensors::Sensor::node` [protected]

Node for communication.

10.193.4.6 `std::vector<NoisePtr> gazebo::sensors::Sensor::noises` [protected]

Noise (p. 748) added to sensor data.

10.193.4.7 `uint32_t gazebo::sensors::Sensor::parentId` [protected]

The sensor's parent ID.

10.193.4.8 `std::string gazebo::sensors::Sensor::parentName` [protected]

Name of the parent.

10.193.4.9 `std::vector<SensorPluginPtr> gazebo::sensors::Sensor::plugins` [protected]

All the plugins for the sensor.

10.193.4.10 `math::Pose gazebo::sensors::Sensor::pose` [protected]

Pose of the sensor.

10.193.4.11 `transport::SubscriberPtr gazebo::sensors::Sensor::poseSub` [protected]

Subscribe to pose updates.

10.193.4.12 `gazebo::rendering::ScenePtr gazebo::sensors::Sensor::scene` [protected]

Pointer to the Scene.

10.193.4.13 `sdf::ElementPtr gazebo::sensors::Sensor::sdf` [protected]

Pointer the the SDF element for the sensor.

10.193.4.14 `common::Time gazebo::sensors::Sensor::updatePeriod` [protected]

Desired time between updates, set indirectly by **Sensor::SetUpdateRate** (p. 916).

10.193.4.15 `gazebo::physics::WorldPtr gazebo::sensors::Sensor::world` [protected]

Pointer to the world.

The documentation for this class was generated from the following file:

- **Sensor.hh**

10.194 SensorFactor Class Reference

The sensor factory; the class is just for namespacing purposes.

```
#include <sensors/sensors.hh>
```

10.194.1 Detailed Description

The sensor factory; the class is just for namespacing purposes.

The documentation for this class was generated from the following file:

- **SensorFactory.hh**

10.195 gazebo::sensors::SensorFactory Class Reference

```
#include <SensorFactory.hh>
```

Static Public Member Functions

- static void **GetSensorTypes** (std::vector< std::string > &_types)
Get all the sensor types.
- static **SensorPtr NewSensor** (const std::string &_className)
Create a new instance of a sensor.
- static void **RegisterAll** ()
Register all known sensors.
- static void **RegisterSensor** (const std::string &_className, **SensorFactoryFn** _factoryfn)
Register a sensor class (called by sensor registration function).

10.195.1 Member Function Documentation

10.195.1.1 static void gazebo::sensors::SensorFactory::GetSensorTypes (std::vector< std::string > &_types) [static]

Get all the sensor types.

Parameters

<code>_types</code>	Vector of strings of the sensor types, populated by function
---------------------	--

10.195.1.2 static **SensorPtr** gazebo::sensors::SensorFactory::NewSensor (const std::string &_className) [static]

Create a new instance of a sensor.

Used by the world when reading the world file.

Parameters

<code>in</code>	<code>_className</code>	Name of sensor class
-----------------	-------------------------	----------------------

Returns

Pointer to **Sensor** (p. 907)

10.195.1.3 `static void gazebo::sensors::SensorFactory::RegisterAll () [static]`

Register all known sensors.

- **sensors::CameraSensor** (p. 227)
- **sensors::DepthCameraSensor** (p. 387)
- **sensors::GpuRaySensor** (p. 477)
- **sensors::RaySensor** (p. 842)
- **sensors::ContactSensor** (p. 287)
- **sensors::RFIDSensor** (p. 859)
- **sensors::RFIDTag** (p. 861)
- **sensors::WirelessTransmitter** (p. 1235)
- **sensors::WirelessReceiver** (p. 1230)

10.195.1.4 `static void gazebo::sensors::SensorFactory::RegisterSensor (const std::string & _className, SensorFactoryFn _factoryfn) [static]`

Register a sensor class (called by sensor registration function).

Parameters

in	<code>_className</code>	Name of class of sensor to register.
in	<code>_factoryfn</code>	Function handle for registration.

The documentation for this class was generated from the following file:

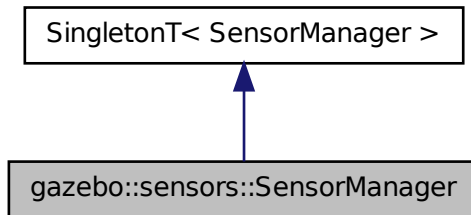
- **SensorFactory.hh**

10.196 gazebo::sensors::SensorManager Class Reference

Class to manage and update all sensors.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::SensorManager:



Classes

- class **ImageSensorContainer**
- class **SensorContainer**

Public Member Functions

- std::string **CreateSensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName) **GAZEBO_DEPRECATED**(2.0)
Deprecated.
- std::string **CreateSensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName, uint32_t _parentId)
Add a sensor from an SDF element.
- void **Fini** ()
Finalize all the sensors.
- **SensorPtr GetSensor** (const std::string &_name) const
Get a sensor.
- **Sensor_V GetSensors** () const
Get all the sensors.
- void **GetSensorTypes** (std::vector< std::string > &_types) const
Get all the sensor types.
- void **Init** ()
Init all the sensors.
- void **RemoveSensor** (const std::string &_name)
Remove a sensor.
- void **RemoveSensors** ()
Remove all sensors.
- void **ResetLastUpdateTimes** ()
Reset last update times in all sensors.

- void **RunThreads** ()
Run sensor updates in separate threads.
- bool **SensorsInitialized** ()
True if SensorManager::initSensors queue is empty i.e.
- void **Stop** ()
Stop the run thread.
- void **Update** (bool _force=false)
Update all the sensors.

Additional Inherited Members

10.196.1 Detailed Description

Class to manage and update all sensors.

10.196.2 Member Function Documentation

10.196.2.1 `std::string gazebo::sensors::SensorManager::CreateSensor (sdf::ElementPtr _elem, const std::string & _worldName, const std::string & _parentName)`

Deprecated.

10.196.2.2 `std::string gazebo::sensors::SensorManager::CreateSensor (sdf::ElementPtr _elem, const std::string & _worldName, const std::string & _parentName, uint32_t _parentId)`

Add a sensor from an SDF element.

This function will also Load and Init the sensor.

Parameters

<code>in</code>	<code>_elem</code>	The SDF element that describes the sensor
<code>in</code>	<code>_worldName</code>	Name of the world in which to create the sensor
<code>in</code>	<code>_parentName</code>	The name of the parent link which the sensor is attached to.

Returns

The name of the sensor

10.196.2.3 `void gazebo::sensors::SensorManager::Fini ()`

Finalize all the sensors.

10.196.2.4 `SensorPtr gazebo::sensors::SensorManager::GetSensor (const std::string & _name) const`

Get a sensor.

Parameters

in	<i>_name</i>	The name of a sensor to find.
----	--------------	-------------------------------

Returns

A pointer to the sensor. NULL if not found.

10.196.2.5 `Sensor_V gazebo::sensors::SensorManager::GetSensors () const`

Get all the sensors.

Returns

Vector of all the sensors.

10.196.2.6 `void gazebo::sensors::SensorManager::GetSensorTypes (std::vector< std::string > & _types) const`

Get all the sensor types.

Parameters

out	<i>All</i>	the sensor types.
-----	------------	-------------------

10.196.2.7 `void gazebo::sensors::SensorManager::Init ()`

Init all the sensors.

10.196.2.8 `void gazebo::sensors::SensorManager::RemoveSensor (const std::string & _name)`

Remove a sensor.

Parameters

in	<i>_name</i>	The name of the sensor to remove.
----	--------------	-----------------------------------

10.196.2.9 `void gazebo::sensors::SensorManager::RemoveSensors ()`

Remove all sensors.

10.196.2.10 `void gazebo::sensors::SensorManager::ResetLastUpdateTimes ()`

Reset last update times in all sensors.

10.196.2.11 `void gazebo::sensors::SensorManager::RunThreads ()`

Run sensor updates in separate threads.

This will only run non-image based sensor updates.

10.196.2.12 `bool gazebo::sensors::SensorManager::SensorsInitialized ()`

True if `SensorManager::initSensors` queue is empty i.e.

all sensors managed by **SensorManager** (p. 920) have been initialized

10.196.2.13 `void gazebo::sensors::SensorManager::Stop ()`

Stop the run thread.

10.196.2.14 `void gazebo::sensors::SensorManager::Update (bool _force = false)`

Update all the sensors.

Checks to see if any sensor need to be initialized first, then updates all sensors once.

Parameters

<code>in</code>	<code><i>_force</i></code>	True force update, false if not
-----------------	----------------------------	---------------------------------

The documentation for this class was generated from the following file:

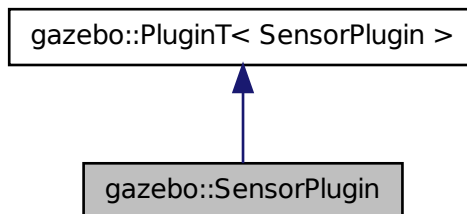
- **SensorManager.hh**

10.197 gazebo::SensorPlugin Class Reference

A plugin with access to `physics::Sensor`.

```
#include <common/common.hh>
```

Inheritance diagram for `gazebo::SensorPlugin`:



Public Member Functions

- **SensorPlugin ()**

Constructor.

- virtual `~SensorPlugin ()`

Destructor.

- virtual void `Init ()`

Override this method for custom plugin initialization behavior.

- virtual void `Load (sensors::SensorPtr _sensor, sdf::ElementPtr _sdf)=0`

Load function.

- virtual void `Reset ()`

Override this method for custom plugin reset behavior.

Additional Inherited Members

10.197.1 Detailed Description

A plugin with access to physics::Sensor.

See [reference](#).

10.197.2 Constructor & Destructor Documentation

10.197.2.1 gazebo::SensorPlugin::SensorPlugin () `[inline]`

Constructor.

References `gazebo::SENSOR_PLUGIN`.

10.197.2.2 virtual gazebo::SensorPlugin::~SensorPlugin () `[inline],[virtual]`

Destructor.

10.197.3 Member Function Documentation

10.197.3.1 virtual void gazebo::SensorPlugin::Init () `[inline],[virtual]`

Override this method for custom plugin initialization behavior.

10.197.3.2 virtual void gazebo::SensorPlugin::Load (sensors::SensorPtr _sensor, sdf::ElementPtr _sdf) `[pure virtual]`

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

<code>in</code>	<code>_sensor</code>	Pointer the Sensor.
<code>in</code>	<code>_sdf</code>	Pointer the the SDF element of the plugin.

10.197.3.3 virtual void gazebo::SensorPlugin::Reset () [inline],[virtual]

Override this method for custom plugin reset behavior.

The documentation for this class was generated from the following file:

- **Plugin.hh**

10.198 gazebo::Server Class Reference

```
#include <Server.hh>
```

Public Member Functions

- **Server** ()
Constructor.
- virtual **~Server** ()
Destructor.
- void **Fini** ()
*Finalize the **Server** (p. 926).*
- bool **GetInitialized** () const
*Get whether the **Server** (p. 926) has been initialized.*
- bool **LoadFile** (const std::string &_filename="worlds/empty.world", const std::string &_physics="")
Load a world file and optionally override physics engine type.
- bool **LoadString** (const std::string &_sdfString)
*Load the **Server** (p. 926) from an SDF string.*
- bool **ParseArgs** (int _argc, char **_argv)
Parse command line arguments.
- bool **PreLoad** ()
Preload the server.
- void **PrintUsage** ()
Output help about gzserver.
- void **Run** ()
*Run the **Server** (p. 926).*
- void **SetParams** (const common::StrStr_M &_params)
Set the parameters.
- void **Stop** ()
*Stop the **Server** (p. 926).*

10.198.1 Constructor & Destructor Documentation

10.198.1.1 gazebo::Server::Server ()

Constructor.

10.198.1.2 virtual gazebo::Server::~Server () [virtual]

Destructor.

10.198.2 Member Function Documentation

10.198.2.1 void gazebo::Server::Fini ()

Finalize the **Server** (p. 926).

10.198.2.2 bool gazebo::Server::GetInitialized () const

Get whether the **Server** (p. 926) has been initialized.

Returns

True if initialized.

10.198.2.3 bool gazebo::Server::LoadFile (const std::string & *_filename* = "worlds/empty.world", const std::string & *_physics* = " ")

Load a world file and optionally override physics engine type.

Parameters

in	<i>_filename</i>	Name of the world file to load.
in	<i>_physics</i>	Physics engine type (ode bullet dart simbody).

Returns

True on success.

10.198.2.4 bool gazebo::Server::LoadString (const std::string & *_sdfString*)

Load the **Server** (p. 926) from an SDF string.

Parameters

in	<i>_sdfString</i>	SDF string from which to load a World.
----	-------------------	--

Returns

True on success.

10.198.2.5 bool gazebo::Server::ParseArgs (int *_argc*, char ** *_argv*)

Parse command line arguments.

Parameters

in	<i>_argc</i>	Number of arguments.
in	<i>_argv</i>	Array of argument values.

Returns

True on success.

10.198.2.6 `bool gazebo::Server::PreLoad ()`

Preload the server.

Returns

True if load was successful.

10.198.2.7 `void gazebo::Server::PrintUsage ()`

Output help about gzserver.

10.198.2.8 `void gazebo::Server::Run ()`

Run the **Server** (p. 926).

10.198.2.9 `void gazebo::Server::SetParams (const common::StrStr_M & _params)`

Set the parameters.

Parameters

<code>in</code>	<code>_params</code>	Map of string parameters
-----------------	----------------------	--------------------------

10.198.2.10 `void gazebo::Server::Stop ()`

Stop the **Server** (p. 926).

The documentation for this class was generated from the following file:

- **Server.hh**

10.199 `gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg` Class Reference

Keeping the CG shader for reference.

```
#include <Heightmap.hh>
```

Public Member Functions

- virtual
Ogre::HighLevelGpuProgramPtr **generateFragmentProgram** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt)

- virtual
Ogre::HighLevelGpuProgramPtr **generateVertexProgram** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt)

Protected Member Functions

- virtual void **defaultVpParams** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, const Ogre::HighLevelGpuProgramPtr &_prog)
- virtual void **generateVertexProgramSource** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpDynamicShadows** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual unsigned int **generateVpDynamicShadowsParams** (unsigned int _texCoordStart, const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpFooter** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpHeader** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)

10.199.1 Detailed Description

Keeping the CG shader for reference.

Utility class to help with generating shaders for Cg / HLSL.

10.199.2 Member Function Documentation

- 10.199.2.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::defaultVpParams (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, const Ogre::HighLevelGpuProgramPtr & _prog)
[protected], [virtual]
- 10.199.2.2 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateFragmentProgram (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt)
[virtual]
- 10.199.2.3 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgram (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt)
[virtual]
- 10.199.2.4 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgramSource (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)
[protected], [virtual]
- 10.199.2.5 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpDynamicShadows (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)
[protected], [virtual]
- 10.199.2.6 virtual unsigned int gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpDynamicShadowsParams (unsigned int _texCoordStart, const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream) [protected], [virtual]

- 10.199.2.7 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpFooter (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)
[protected], [virtual]
- 10.199.2.8 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpHeader (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)
[protected], [virtual]

The documentation for this class was generated from the following file:

- **Heightmap.hh**

10.200 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL Class Reference

Utility class to help with generating shaders for GLSL.

```
#include <Heightmap.hh>
```

Public Member Functions

- virtual
Ogre::HighLevelGpuProgramPtr **generateFragmentProgram** (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt)
- virtual
Ogre::HighLevelGpuProgramPtr **generateVertexProgram** (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt)
- virtual void **updateParams** (const SM2Profile * _prof, const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain, bool _compositeMap)

Protected Member Functions

- virtual void **defaultVpParams** (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, const Ogre::HighLevelGpuProgramPtr & _prog)
- void **generateFpDynamicShadows** (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)
- virtual void **generateFpDynamicShadowsHelpers** (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & _outStream)
- virtual void **generateFpDynamicShadowsParams** (Ogre::uint * _texCoord, Ogre::uint * _sampler, const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)
- virtual void **generateFpFooter** (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & _outStream)
- virtual void **generateFpHeader** (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & _outStream)
- virtual void **generateFpLayer** (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType tt, Ogre::uint _layer, Ogre::StringUtil::StrStreamType & _outStream)
- virtual void **generateFragmentProgramSource** (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)
- virtual void **generateVertexProgramSource** (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)

- virtual void **generateVpDynamicShadows** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual unsigned int **generateVpDynamicShadowsParams** (unsigned int _texCoordStart, const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpFooter** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpHeader** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **updateVpParams** (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, const Ogre::GpuProgramParametersSharedPtr &_params)

10.200.1 Detailed Description

Utility class to help with generating shaders for GLSL.

10.200.2 Member Function Documentation

- 10.200.2.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::defaultVpParams (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, const Ogre::HighLevelGpuProgramPtr & .prog) [protected], [virtual]
- 10.200.2.2 void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadows (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & .outStream) [protected]
- 10.200.2.3 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadowsHelpers (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & .outStream) [protected], [virtual]
- 10.200.2.4 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadowsParams (Ogre::uint *_texCoord, Ogre::uint *_sampler, const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & .outStream) [protected], [virtual]
- 10.200.2.5 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpFooter (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & .outStream) [protected], [virtual]
- 10.200.2.6 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpHeader (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & .outStream) [protected], [virtual]
- 10.200.2.7 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpLayer (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType tt, Ogre::uint _layer, Ogre::StringUtil::StrStreamType & .outStream) [protected], [virtual]
- 10.200.2.8 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFragmentProgram (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt) [virtual]

- 10.200.2.9 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFragmentProgramSource (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.200.2.10 `virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVertexProgram (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt)` [virtual]
- 10.200.2.11 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVertexProgramSource (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.200.2.12 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpDynamicShadows (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.200.2.13 `virtual unsigned int gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpDynamicShadowsParams (unsigned int _texCoordStart, const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.200.2.14 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpFooter (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.200.2.15 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpHeader (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.200.2.16 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::updateParams (const SM2Profile * _prof, const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain, bool _compositeMap)` [virtual]
- 10.200.2.17 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::updateVpParams (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, const Ogre::GpuProgramParametersSharedPtr & _params)` [protected],[virtual]

The documentation for this class was generated from the following file:

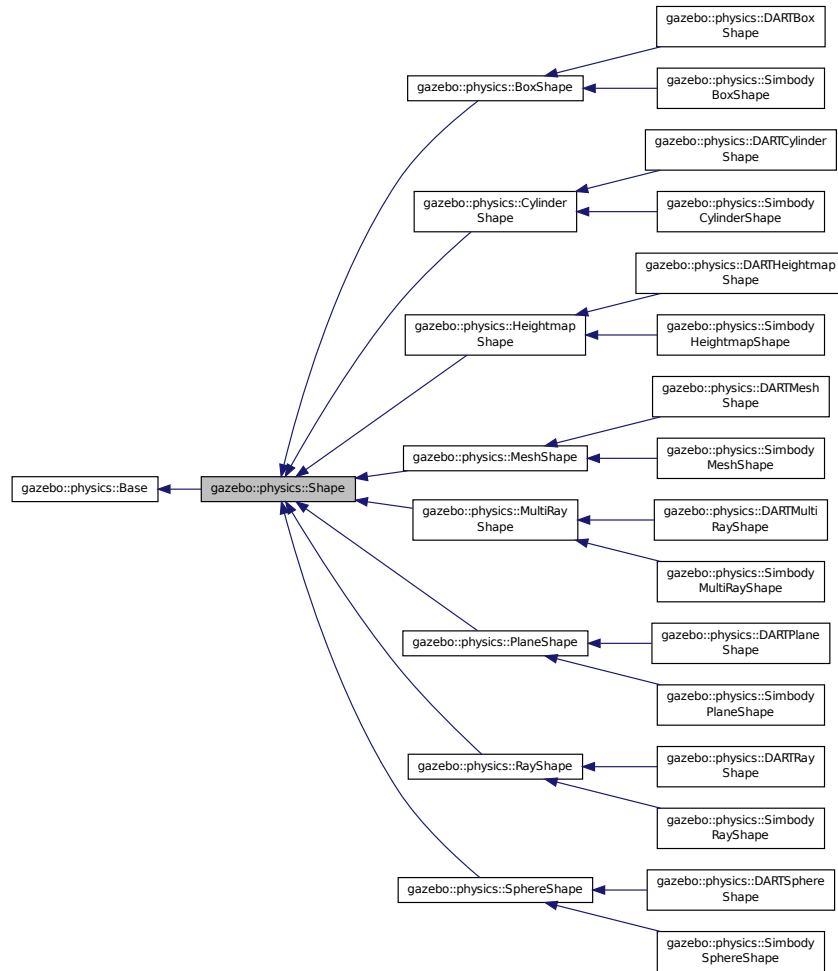
- Heightmap.hh

10.201 gazebo::physics::Shape Class Reference

Base (p. 168) class for all shapes.

```
#include <physics/physics.hh>
```


Inheritance diagram for gazebo::physics::Shape:



Public Member Functions

- **Shape** (**CollisionPtr** _parent)
Constructor.
- virtual **~Shape** ()
Destructor.
- virtual void **FillMsg** (msgs::Geometry &_msg)=0
Fill in the values for a geometry message.
- virtual **math::Vector3** **GetScale** () const
Get the scale of the shape.
- virtual void **Init** ()=0
Initialize the shape.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)=0
Process a geometry message.

- virtual void **SetScale** (const **math::Vector3** &_scale)=0
Set the scale of the shape.

Protected Attributes

- **CollisionPtr collisionParent**
This shape's collision parent.
- **math::Vector3 scale**
This shape's scale;.

Additional Inherited Members

10.201.1 Detailed Description

Base (p. 168) class for all shapes.

10.201.2 Constructor & Destructor Documentation

10.201.2.1 `gazebo::physics::Shape::Shape (CollisionPtr _parent) [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Parent of the shape.
----	----------------------	----------------------

10.201.2.2 `virtual gazebo::physics::Shape::~~Shape () [virtual]`

Destructor.

10.201.3 Member Function Documentation

10.201.3.1 `virtual void gazebo::physics::Shape::FillMsg (msgs::Geometry & _msg) [pure virtual]`

Fill in the values for a geometry message.

Parameters

out	<code>_msg</code>	The geometry message to fill.
-----	-------------------	-------------------------------

Implemented in **gazebo::physics::MultiRayShape** (p. 727), **gazebo::physics::RayShape** (p. 851), **gazebo::physics::HeightmapShape** (p. 509), **gazebo::physics::PlaneShape** (p. 792), **gazebo::physics::MeshShape** (p. 676), **gazebo::physics::CylinderShape** (p. 299), **gazebo::physics::SphereShape** (p. 1056), and **gazebo::physics::BoxShape** (p. 187).

10.201.3.2 `virtual math::Vector3 gazebo::physics::Shape::GetScale () const [virtual]`

Get the scale of the shape.

Returns

Scale of the shape.

10.201.3.3 `virtual void gazebo::physics::Shape::Init () [pure virtual]`

Initialize the shape.

Reimplemented from `gazebo::physics::Base` (p. 177).

Implemented in `gazebo::physics::RayShape` (p. 853), `gazebo::physics::HeightmapShape` (p. 510), `gazebo::physics::MeshShape` (p. 677), `gazebo::physics::MultiRayShape` (p. 730), `gazebo::physics::PlaneShape` (p. 793), `gazebo::physics::SphereShape` (p. 1056), `gazebo::physics::BoxShape` (p. 187), `gazebo::physics::CylinderShape` (p. 300), `gazebo::physics::SimbodyHeightmapShape` (p. 950), `gazebo::physics::SimbodyMeshShape` (p. 982), `gazebo::physics::DARTHeightmapShape` (p. 317), and `gazebo::physics::DARTMeshShape` (p. 349).

10.201.3.4 `virtual void gazebo::physics::Shape::ProcessMsg (const msgs::Geometry & _msg) [pure virtual]`

Process a geometry message.

Parameters

in	_msg	The message to set values from.
----	------	---------------------------------

Implemented in `gazebo::physics::MultiRayShape` (p. 730), `gazebo::physics::RayShape` (p. 853), `gazebo::physics::HeightmapShape` (p. 511), `gazebo::physics::PlaneShape` (p. 793), `gazebo::physics::MeshShape` (p. 677), `gazebo::physics::CylinderShape` (p. 300), `gazebo::physics::SphereShape` (p. 1056), and `gazebo::physics::BoxShape` (p. 187).

10.201.3.5 `virtual void gazebo::physics::Shape::SetScale (const math::Vector3 & _scale) [pure virtual]`

Set the scale of the shape.

Parameters

in	_scale	Scale to set the shape to.
----	--------	----------------------------

Implemented in `gazebo::physics::RayShape` (p. 854), `gazebo::physics::PlaneShape` (p. 794), `gazebo::physics::MeshShape` (p. 678), `gazebo::physics::CylinderShape` (p. 300), `gazebo::physics::HeightmapShape` (p. 511), `gazebo::physics::SphereShape` (p. 1057), `gazebo::physics::BoxShape` (p. 188), and `gazebo::physics::MultiRayShape` (p. 730).

10.201.4 Member Data Documentation

10.201.4.1 `CollisionPtr gazebo::physics::Shape::collisionParent [protected]`

This shape's collision parent.

10.201.4.2 `math::Vector3 gazebo::physics::Shape::scale [protected]`

This shape's scale;

The documentation for this class was generated from the following file:

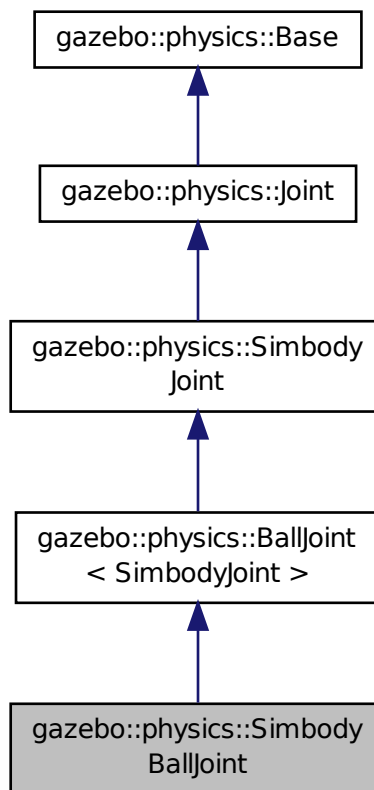
- **Shape.hh**

10.202 gazebo::physics::SimbodyBallJoint Class Reference

SimbodyBallJoint (p. 936) class models a ball joint in Simbody.

```
#include <SimbodyBallJoint.hh>
```

Inheritance diagram for gazebo::physics::SimbodyBallJoint:



Public Member Functions

- **SimbodyBallJoint** (SimTK::MultibodySystem * _world, **BasePtr** _parent)
Simbody Ball Joint (p. 541) *Constructor.*
- virtual **~SimbodyBallJoint** ()
Destructor.
- **math::Vector3 GetAnchor** (unsigned int _index) const

- Get the anchor point.*

 - virtual **math::Angle GetAngleImpl** (unsigned int _index) const

Get the angle of an axis helper function.
- virtual **math::Vector3 GetAxis** (unsigned int) const
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const
- Get the axis of rotation in global coordinate frame.*

 - virtual **math::Angle GetHighStop** (unsigned int _index)

Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (unsigned int _index)
- Get the low stop of an axis(index).*

 - virtual double **GetMaxForce** (unsigned int _index)

Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (unsigned int _index) const
- Get the rotation rate of an axis(index)*

 - virtual void **Load** (sdf::ElementPtr _sdf)

*Template to ::Load the **BallJoint** (p. 167).*
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)
- Set the axis of rotation where axis is specified in local joint frame.*

 - virtual bool **SetHighStop** (unsigned int _index, const **math::Angle** &_angle)

Set the high stop of an axis(index).
- virtual bool **SetLowStop** (unsigned int _index, const **math::Angle** &_angle)
- Set the low stop of an axis(index).*

 - virtual void **SetMaxForce** (unsigned int _index, double _t)

Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (unsigned int _index, double _angle)
- Set the velocity of an axis(index).*

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _torque)
- Set the force applied to this **physics::Joint** (p. 541).*

Additional Inherited Members

10.202.1 Detailed Description

SimbodyBallJoint (p. 936) class models a ball joint in Simbody.

10.202.2 Constructor & Destructor Documentation

10.202.2.1 gazebo::physics::SimbodyBallJoint::SimbodyBallJoint (**SimTK::MultibodySystem** * _world, **BasePtr** _parent)

Simbody Ball **Joint** (p. 541) Constructor.

10.202.2.2 virtual gazebo::physics::SimbodyBallJoint::~SimbodyBallJoint () [virtual]

Destructor.

10.202.3 Member Function Documentation

10.202.3.1 `math::Vector3 gazebo::physics::SimbodyBallJoint::GetAnchor (unsigned int _index) const` [virtual]

Get the anchor point.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Anchor value for the axis.

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 963).

10.202.3.2 `virtual math::Angle gazebo::physics::SimbodyBallJoint::GetAngleImpl (unsigned int _index) const` [virtual]

Get the angle of an axis helper function.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Angle of the axis.

Implements `gazebo::physics::Joint` (p. 550).

10.202.3.3 `virtual math::Vector3 gazebo::physics::SimbodyBallJoint::GetAxis (unsigned int) const` [inline],
[virtual]

10.202.3.4 `virtual math::Vector3 gazebo::physics::SimbodyBallJoint::GetGlobalAxis (unsigned int _index) const`
[virtual]

Get the axis of rotation in global coordinate frame.

Parameters

in	<i>_index</i>	Index of the axis to get.
----	---------------	---------------------------

Returns

Axis value for the provided index.

Implements `gazebo::physics::Joint` (p. 553).

10.202.3.5 `virtual math::Angle gazebo::physics::SimbodyBallJoint::GetHighStop (unsigned int _index)` [virtual]

Get the high stop of an axis(index).

This function is replaced by `GetUpperLimit(unsigned int)`. If you are interested in getting the value of `dParamHiStop*`, use `GetAttribute(hi_stop, _index)`

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Angle of the high stop value.

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 965).

10.202.3.6 `virtual math::Angle gazebo::physics::SimbodyBallJoint::GetLowStop (unsigned int _index) [virtual]`

Get the low stop of an axis(index).

This function is replaced by `GetLowerLimit(unsigned int)`. If you are interested in getting the value of `dParamHiStop*`, use `GetAttribute(hi_stop, _index)`

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Angle of the low stop value.

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 966).

10.202.3.7 `virtual double gazebo::physics::SimbodyBallJoint::GetMaxForce (unsigned int _index) [virtual]`

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

The maximum force.

Implements `gazebo::physics::Joint` (p. 556).

10.202.3.8 `virtual double gazebo::physics::SimbodyBallJoint::GetVelocity (unsigned int _index) const [virtual]`

Get the rotation rate of an axis(index)

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

The rotational velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 558).

10.202.3.9 virtual void gazebo::physics::SimbodyBallJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Template to ::Load the **BallJoint** (p. 167).

Parameters

in	<i>_sdf</i>	SDF to load the joint from.
----	-------------	-----------------------------

Reimplemented from **gazebo::physics::BallJoint**< **SimbodyJoint** > (p. 168).

10.202.3.10 virtual void gazebo::physics::SimbodyBallJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	<i>_index</i>	Index of the axis to set.
in	<i>_axis</i>	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 968).

10.202.3.11 virtual void gazebo::physics::SimbodyBallJoint::SetForceImpl (unsigned int *_index*, double *_force*) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 541)).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_force</i>	Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only.

Implements **gazebo::physics::SimbodyJoint** (p. 968).

10.202.3.12 virtual bool gazebo::physics::SimbodyBallJoint::SetHighStop (unsigned int *_index*, const math::Angle & *_angle*) [virtual]

Set the high stop of an axis(index).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_angle</i>	High stop angle.

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 969).

10.202.3.13 `virtual bool gazebo::physics::SimbodyBallJoint::SetLowStop (unsigned int _index, const math::Angle & _angle)`
`[virtual]`

Set the low stop of an axis(*index*).

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_angle</i></code>	Low stop angle.

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 969).

10.202.3.14 `virtual void gazebo::physics::SimbodyBallJoint::SetMaxForce (unsigned int _index, double _force)` `[virtual]`

Set the max allowed force of an axis(*index*).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_force</i></code>	Maximum force that can be applied to the axis.

Implements `gazebo::physics::Joint` (p. 563).

10.202.3.15 `virtual void gazebo::physics::SimbodyBallJoint::SetVelocity (unsigned int _index, double _vel)` `[virtual]`

Set the velocity of an axis(*index*).

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_vel</i></code>	Velocity.

Implements `gazebo::physics::Joint` (p. 565).

The documentation for this class was generated from the following file:

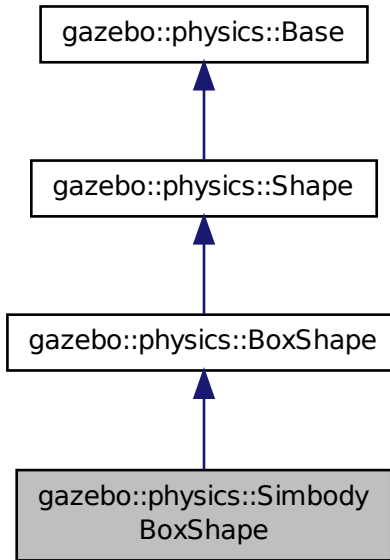
- `SimbodyBallJoint.hh`

10.203 gazebo::physics::SimbodyBoxShape Class Reference

Simbody box collision.

```
#include <SimbodyBoxShape.hh>
```

Inheritance diagram for gazebo::physics::SimbodyBoxShape:



Public Member Functions

- **SimbodyBoxShape** (*CollisionPtr* _parent)
Constructor.
- virtual **~SimbodyBoxShape** ()
Destructor.
- void **SetSize** (const **math::Vector3** &_size)
Set the size of the box.

Additional Inherited Members

10.203.1 Detailed Description

Simbody box collision.

10.203.2 Constructor & Destructor Documentation

10.203.2.1 gazebo::physics::SimbodyBoxShape::SimbodyBoxShape (*CollisionPtr* _parent) [inline]

Constructor.

10.203.2.2 virtual gazebo::physics::SimbodyBoxShape::~~SimbodyBoxShape () [inline],[virtual]

Destructor.

10.203.3 Member Function Documentation

10.203.3.1 void gazebo::physics::SimbodyBoxShape::SetSize (const math::Vector3 & *_size*) [inline],[virtual]

Set the size of the box.

Parameters

in	<i>_size</i>	Size of each side of the box.
----	--------------	-------------------------------

Reimplemented from **gazebo::physics::BoxShape** (p. 188).

References gazebo::math::equal(), gzerr, gzwarn, gazebo::physics::BoxShape::SetSize(), gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

The documentation for this class was generated from the following file:

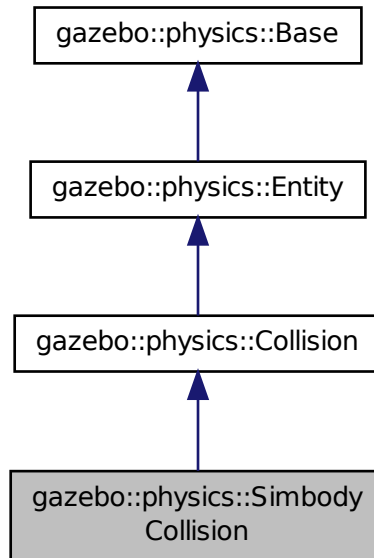
- **SimbodyBoxShape.hh**

10.204 gazebo::physics::SimbodyCollision Class Reference

Simbody collisions.

```
#include <SimbodyCollision.hh>
```

Inheritance diagram for gazebo::physics::SimbodyCollision:



Public Member Functions

- **SimbodyCollision** (LinkPtr _parent)
Constructor.
- virtual **~SimbodyCollision** ()
Destructor.
- virtual **math::Box GetBoundingBox** () const
Get the bounding box for this collision.
- SimTK::ContactGeometry * **GetCollisionShape** () const
Get the simbody collision shape.
- virtual void **Load** (sdf::ElementPtr _ptr)
Load the collision.
- virtual void **OnPoseChange** ()
This function is called when the entity's (or one of its parents) pose of the parent has changed.
- virtual void **SetCategoryBits** (unsigned int _bits)
Set the category bits, used during collision detection.
- virtual void **SetCollideBits** (unsigned int _bits)
Set the collide bits, used during collision detection.
- void **SetCollisionShape** (SimTK::ContactGeometry *_shape)
Set the collision shape.

Additional Inherited Members

10.204.1 Detailed Description

Simbody collisions.

10.204.2 Constructor & Destructor Documentation

10.204.2.1 gazebo::physics::SimbodyCollision::SimbodyCollision (LinkPtr *_parent*)

Constructor.

10.204.2.2 virtual gazebo::physics::SimbodyCollision::~~SimbodyCollision () [virtual]

Destructor.

10.204.3 Member Function Documentation

10.204.3.1 virtual math::Box gazebo::physics::SimbodyCollision::GetBoundingBox () const [virtual]

Get the bounding box for this collision.

Returns

The bounding box.

Implements **gazebo::physics::Collision** (p. 239).

10.204.3.2 SimTK::ContactGeometry* gazebo::physics::SimbodyCollision::GetCollisionShape () const

Get the simbody collision shape.

Returns

SimTK (p. 137) geometry used as the collision shape.

10.204.3.3 virtual void gazebo::physics::SimbodyCollision::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the collision.

Parameters

<i>in</i>	<i>_sdf</i>	SDF to load from.
-----------	-------------	-------------------

Reimplemented from **gazebo::physics::Collision** (p. 243).

10.204.3.4 virtual void gazebo::physics::SimbodyCollision::OnPoseChange () [virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implements **gazebo::physics::Entity** (p. 412).

10.204.3.5 `virtual void gazebo::physics::SimbodyCollision::SetCategoryBits (unsigned int _bits) [virtual]`

Set the category bits, used during collision detection.

Parameters

<code>in</code>	<code><i>_bits</i></code>	The bits to set.
-----------------	---------------------------	------------------

Implements **gazebo::physics::Collision** (p. 243).

10.204.3.6 `virtual void gazebo::physics::SimbodyCollision::SetCollideBits (unsigned int _bits) [virtual]`

Set the collide bits, used during collision detection.

Parameters

<code>in</code>	<code><i>_bits</i></code>	The bits to set.
-----------------	---------------------------	------------------

Implements **gazebo::physics::Collision** (p. 243).

10.204.3.7 `void gazebo::physics::SimbodyCollision::SetCollisionShape (SimTK::ContactGeometry * _shape)`

Set the collision shape.

Parameters

<code>in</code>	<code><i>_shape</i></code>	SimTK (p. 137) geometry to use as the collision SimTK (p. 137) geometry to use as the collision shape.
-----------------	----------------------------	--

The documentation for this class was generated from the following file:

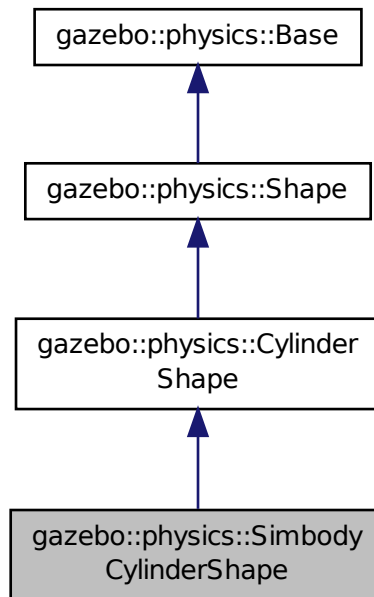
- **SimbodyCollision.hh**

10.205 gazebo::physics::SimbodyCylinderShape Class Reference

Cylinder collision.

```
#include <SimbodyCylinderShape.hh>
```

Inheritance diagram for gazebo::physics::SimbodyCylinderShape:



Public Member Functions

- **SimbodyCylinderShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~SimbodyCylinderShape** ()
Destructor.
- void **SetSize** (double _radius, double _length)
Set the size of the cylinder.

Additional Inherited Members

10.205.1 Detailed Description

Cylinder collision.

10.205.2 Constructor & Destructor Documentation

10.205.2.1 gazebo::physics::SimbodyCylinderShape::SimbodyCylinderShape (**CollisionPtr** _parent) [inline]

Constructor.

10.205.2.2 `virtual gazebo::physics::SimbodyCylinderShape::~~SimbodyCylinderShape () [inline], [virtual]`

Destructor.

10.205.3 Member Function Documentation

10.205.3.1 `void gazebo::physics::SimbodyCylinderShape::SetSize (double _radius, double _length) [inline], [virtual]`

Set the size of the cylinder.

Parameters

<code>in</code>	<code><i>_radius</i></code>	New radius.
<code>in</code>	<code><i>_length</i></code>	New length.

Reimplemented from `gazebo::physics::CylinderShape` (p. 301).

References `gazebo::math::equal()`, `gzerr`, `gzwarn`, and `gazebo::physics::CylinderShape::SetSize()`.

The documentation for this class was generated from the following file:

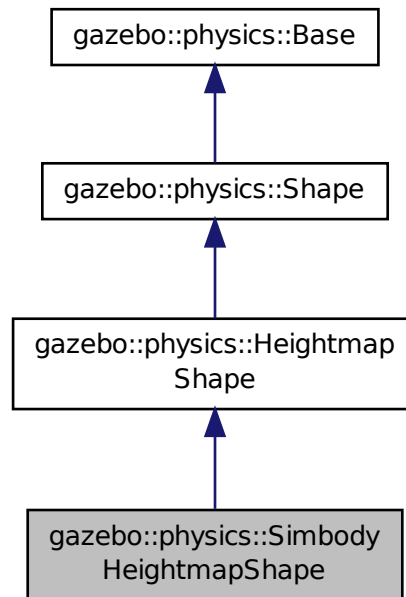
- `SimbodyCylinderShape.hh`

10.206 gazebo::physics::SimbodyHeightmapShape Class Reference

Height map collision.

```
#include <SimbodyHeightmapShape.hh>
```


Inheritance diagram for gazebo::physics::SimbodyHeightmapShape:



Public Member Functions

- **SimbodyHeightmapShape** (*CollisionPtr* _parent)
Constructor.
- virtual **~SimbodyHeightmapShape** ()
Destructor.
- virtual void **Init** ()
Initialize the heightmap.

Additional Inherited Members

10.206.1 Detailed Description

Height map collision.

10.206.2 Constructor & Destructor Documentation

10.206.2.1 gazebo::physics::SimbodyHeightmapShape::SimbodyHeightmapShape (*CollisionPtr* _parent)

Constructor.

10.206.2.2 virtual gazebo::physics::SimbodyHeightmapShape::~~SimbodyHeightmapShape () [virtual]

Destructor.

10.206.3 Member Function Documentation

10.206.3.1 virtual void gazebo::physics::SimbodyHeightmapShape::Init () [virtual]

Initialize the heightmap.

Reimplemented from **gazebo::physics::HeightmapShape** (p. 510).

The documentation for this class was generated from the following file:

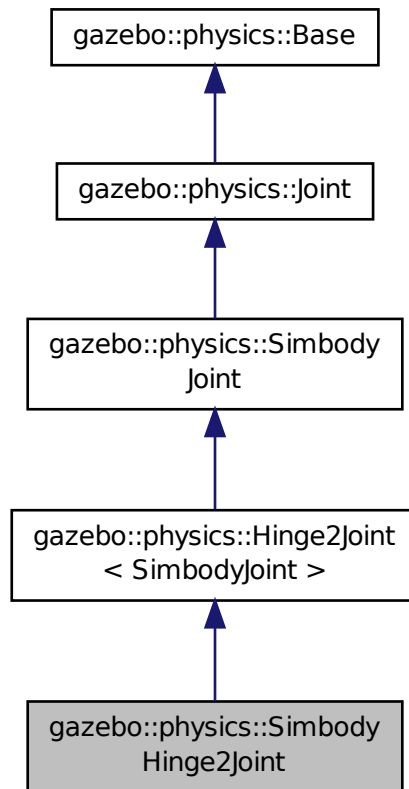
- **SimbodyHeightmapShape.hh**

10.207 gazebo::physics::SimbodyHinge2Joint Class Reference

A two axis hinge joint.

```
#include <SimbodyHinge2Joint.hh>
```

Inheritance diagram for gazebo::physics::SimbodyHinge2Joint:



Public Member Functions

- **SimbodyHinge2Joint** (SimTK::MultibodySystem ***world**, **BasePtr** _parent)
Constructor.
- virtual **~SimbodyHinge2Joint** ()
Destructor.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const
Get the anchor point.
- virtual **math::Vector3 GetAxis** (unsigned int _index) const
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const
Get the axis of rotation in global coordinate frame.
- virtual double **GetMaxForce** (unsigned int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (unsigned int _index) const
Get the rotation rate of an axis(index)
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

Set the axis of rotation where axis is specified in local joint frame.

- virtual void **SetMaxForce** (unsigned int _index, double _t)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (unsigned int _index, double _angle)
Set the velocity of an axis(index).

Protected Member Functions

- virtual **math::Angle GetAngleImpl** (unsigned int _index) const
Get the angle of an axis helper function.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the joint.
- virtual void **SetForceImpl** (unsigned int _index, double _torque)
Set the torque.

Additional Inherited Members

10.207.1 Detailed Description

A two axis hinge joint.

10.207.2 Constructor & Destructor Documentation

10.207.2.1 gazebo::physics::SimbodyHinge2Joint::SimbodyHinge2Joint (SimTK::MultibodySystem * world, BasePtr _parent)

Constructor.

10.207.2.2 virtual gazebo::physics::SimbodyHinge2Joint::~~SimbodyHinge2Joint () [virtual]

Destructor.

10.207.3 Member Function Documentation

10.207.3.1 virtual **math::Vector3** gazebo::physics::SimbodyHinge2Joint::GetAnchor (unsigned int _index) const
[virtual]

Get the anchor point.

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

Anchor value for the axis.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 963).

10.207.3.2 `virtual math::Angle gazebo::physics::SimbodyHinge2Joint::GetAngleImpl (unsigned int _index) const`
`[protected], [virtual]`

Get the angle of an axis helper function.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 550).

10.207.3.3 `virtual math::Vector3 gazebo::physics::SimbodyHinge2Joint::GetAxis (unsigned int _index) const` `[virtual]`

10.207.3.4 `virtual math::Vector3 gazebo::physics::SimbodyHinge2Joint::GetGlobalAxis (unsigned int _index) const`
`[virtual]`

Get the axis of rotation in global coordinate frame.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis to get.
-----------------	----------------------------	---------------------------

Returns

Axis value for the provided index.

Implements **gazebo::physics::Joint** (p. 553).

10.207.3.5 `virtual double gazebo::physics::SimbodyHinge2Joint::GetMaxForce (unsigned int _index)` `[virtual]`

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 556).

10.207.3.6 `virtual double gazebo::physics::SimbodyHinge2Joint::GetVelocity (unsigned int _index) const` `[virtual]`

Get the rotation rate of an axis(index)

Parameters

in	<code>_index</code>	Index of the axis.
----	---------------------	--------------------

Returns

The rotational velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 558).

10.207.3.7 `virtual void gazebo::physics::SimbodyHinge2Joint::Load (sdf::ElementPtr _sdf)` [protected], [virtual]

Load the joint.

Parameters

in	<code>_sdf</code>	SDF values to load from.
----	-------------------	--------------------------

Reimplemented from **gazebo::physics::Hinge2Joint**< **SimbodyJoint** > (p. 513).

10.207.3.8 `virtual void gazebo::physics::SimbodyHinge2Joint::SetAxis (unsigned int _index, const math::Vector3 & _axis)`
[virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	<code>_index</code>	Index of the axis to set.
in	<code>_axis</code>	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 968).

10.207.3.9 `virtual void gazebo::physics::SimbodyHinge2Joint::SetForceImpl (unsigned int _index, double _torque)`
[protected], [virtual]

Set the torque.

Implements **gazebo::physics::SimbodyJoint** (p. 968).

10.207.3.10 `virtual void gazebo::physics::SimbodyHinge2Joint::SetMaxForce (unsigned int _index, double _force)`
[virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_force</code>	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 563).

10.207.3.11 virtual void gazebo::physics::SimbodyHinge2Joint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_vel</i>	Velocity.

Implements **gazebo::physics::Joint** (p. 565).

The documentation for this class was generated from the following file:

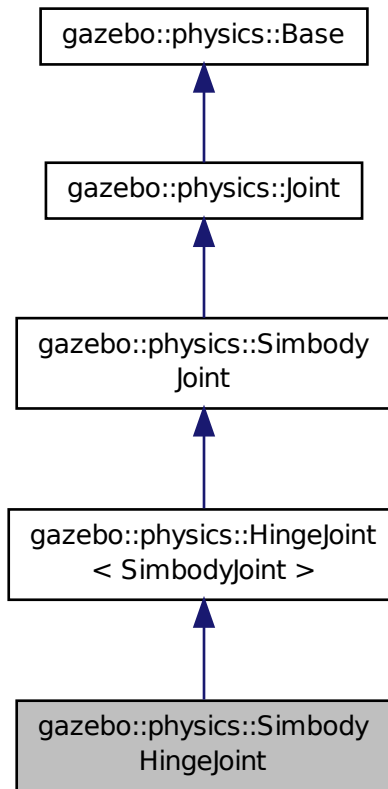
- **SimbodyHinge2Joint.hh**

10.208 gazebo::physics::SimbodyHingeJoint Class Reference

A single axis hinge joint.

```
#include <SimbodyHingeJoint.hh>
```

Inheritance diagram for gazebo::physics::SimbodyHingeJoint:



Public Member Functions

- **SimbodyHingeJoint** (SimTK::MultibodySystem *world, BasePtr _parent)
Constructor.
- virtual ~**SimbodyHingeJoint** ()
Destructor.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const
Get the axis of rotation in global coordinate frame.
- virtual double **GetMaxForce** (unsigned int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (unsigned int _index) const
Get the rotation rate of an axis(index)
- virtual void **RestoreSimbodyState** (SimTK::State &_state)
restore simbody state for spawning
- virtual void **SaveSimbodyState** (const SimTK::State &_state)
save simbody state for spawning

- void **SetAxis** (unsigned int *_index*, const **math::Vector3** & *_axis*)
Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int *_index*, double *_t*)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (unsigned int *_index*, double *_rate*)
Set the velocity of an axis(index).

Protected Member Functions

- virtual **math::Angle** **GetAngleImpl** (unsigned int *_index*) const
Get the angle of an axis helper function.
- virtual void **Load** (sdf::ElementPtr *_sdf*)
Load joint.
- virtual void **SetForceImpl** (unsigned int *_index*, double *_torque*)
*Set the force applied to this **physics::Joint** (p. 541).*

Additional Inherited Members

10.208.1 Detailed Description

A single axis hinge joint.

10.208.2 Constructor & Destructor Documentation

10.208.2.1 gazebo::physics::SimbodyHingeJoint::SimbodyHingeJoint (**SimTK::MultibodySystem** * *world*, **BasePtr** *_parent*)

Constructor.

10.208.2.2 virtual gazebo::physics::SimbodyHingeJoint::~~SimbodyHingeJoint () [virtual]

Destructor.

10.208.3 Member Function Documentation

10.208.3.1 virtual **math::Angle** gazebo::physics::SimbodyHingeJoint::GetAngleImpl (unsigned int *_index*) const
[protected], [virtual]

Get the angle of an axis helper function.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 550).

10.208.3.2 `virtual math::Vector3 gazebo::physics::SimbodyHingeJoint::GetGlobalAxis (unsigned int _index) const`
`[virtual]`

Get the axis of rotation in global coordinate frame.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis to get.
-----------------	----------------------------	---------------------------

Returns

Axis value for the provided index.

Implements `gazebo::physics::Joint` (p. 553).

10.208.3.3 `virtual double gazebo::physics::SimbodyHingeJoint::GetMaxForce (unsigned int _index)` `[virtual]`

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

The maximum force.

Implements `gazebo::physics::Joint` (p. 556).

10.208.3.4 `virtual double gazebo::physics::SimbodyHingeJoint::GetVelocity (unsigned int _index) const` `[virtual]`

Get the rotation rate of an axis(index)

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

The rotational velocity of the joint axis.

Implements `gazebo::physics::Joint` (p. 558).

10.208.3.5 `virtual void gazebo::physics::SimbodyHingeJoint::Load (sdf::ElementPtr _sdf)` `[protected],[virtual]`

Load joint.

Parameters

<code>in</code>	<code><i>_sdf</i></code>	Pointer to SDF element
-----------------	--------------------------	------------------------

Reimplemented from `gazebo::physics::HingeJoint` < `SimbodyJoint` > (p. 514).

10.208.3.6 `virtual void gazebo::physics::SimbodyHingeJoint::RestoreSimbodyState (SimTK::State & _state) [virtual]`

restore simbody state for spawning

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 967).

10.208.3.7 `virtual void gazebo::physics::SimbodyHingeJoint::SaveSimbodyState (const SimTK::State & _state) [virtual]`

save simbody state for spawning

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 967).

10.208.3.8 `void gazebo::physics::SimbodyHingeJoint::SetAxis (unsigned int _index, const math::Vector3 & _axis) [virtual]`

Set the axis of rotation where axis is specified in local joint frame.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to set.
<code>in</code>	<code>_axis</code>	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 968).

10.208.3.9 `virtual void gazebo::physics::SimbodyHingeJoint::SetForceImpl (unsigned int _index, double _force) [protected], [virtual]`

Set the force applied to this `physics::Joint` (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to `SetForceImpl` to the same joint in the same time step will accumulate forces on that `Joint` (p. 541)).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_force</code>	Force value. internal force, e.g. damping forces. This way, <code>Joint::appliedForce</code> keep track of external forces only.

Implements `gazebo::physics::SimbodyJoint` (p. 968).

10.208.3.10 `virtual void gazebo::physics::SimbodyHingeJoint::SetMaxForce (unsigned int _index, double _force) [virtual]`

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_force</code>	Maximum force that can be applied to the axis.

Implements `gazebo::physics::Joint` (p. 563).

10.208.3.11 `virtual void gazebo::physics::SimbodyHingeJoint::SetVelocity (unsigned int _index, double _vel) [virtual]`

Set the velocity of an axis(index).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_vel</code>	Velocity.

Implements `gazebo::physics::Joint` (p. 565).

The documentation for this class was generated from the following file:

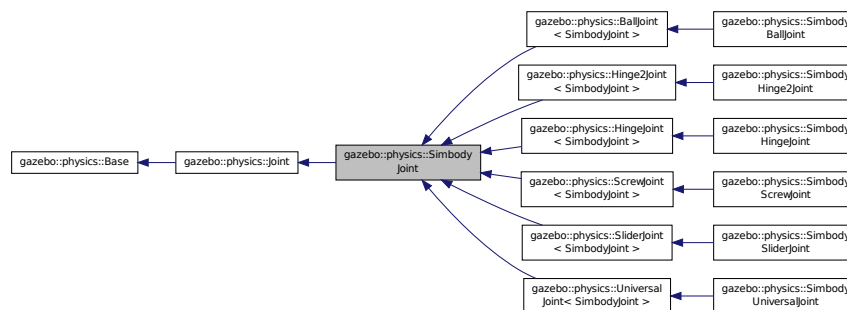
- `SimbodyHingeJoint.hh`

10.209 gazebo::physics::SimbodyJoint Class Reference

Base (p. 168) class for all joints.

```
#include <SimbodyJoint.hh>
```

Inheritance diagram for `gazebo::physics::SimbodyJoint`:



Public Member Functions

- **SimbodyJoint** (**BasePtr** *_parent*)
Constructor.
- virtual `~SimbodyJoint` ()
Destructor.
- virtual bool **AreConnected** (**LinkPtr** *_one*, **LinkPtr** *_two*) const
Determines if the two bodies are connected by a joint.
- virtual void **CacheForceTorque** ()
*Cache **Joint** (p. 541) Force Torque Values if necessary for physics engine.*
- virtual void **Detach** ()
Detach this joint from all links.

- virtual **math::Vector3 GetAnchor** (unsigned int _index) const
Get the anchor point.
- virtual double **GetAttribute** (const std::string &_key, unsigned int _index) **GAZEBO_DEPRECATED(3.0)**
Get a non-generic parameter for the joint.
- virtual double **GetForce** (unsigned int _index)
- virtual **JointWrench GetForceTorque** (unsigned int _index)
get internal force and torque values at a joint.
- virtual **math::Angle GetHighStop** (unsigned int _index)
Get the high stop of an axis(index).
- virtual **LinkPtr GetJointLink** (unsigned int _index) const
Get the link to which the joint is attached according the _index.
- virtual **math::Vector3 GetLinkForce** (unsigned int _index) const
*Get the forces applied to the center of mass of a **physics::Link** (p. 595) due to the existence of this **Joint** (p. 541).*
- virtual **math::Vector3 GetLinkTorque** (unsigned int _index) const
*Get the torque applied to the center of mass of a **physics::Link** (p. 595) due to the existence of this **Joint** (p. 541).*
- virtual **math::Angle GetLowStop** (unsigned int _index)
Get the low stop of an axis(index).
- virtual double **GetParam** (const std::string &_key, unsigned int _index)
Get a non-generic parameter for the joint.
- virtual void **Load** (sdf::ElementPtr _sdf)
*Load **physics::Joint** (p. 541) from a SDF sdf::Element.*
- virtual void **Reset** ()
Reset the joint.
- virtual void **RestoreSimbodyState** (SimTK::State &_state)
- virtual void **SaveSimbodyState** (const SimTK::State &_state)
- virtual void **SetAnchor** (unsigned int _index, const **gazebo::math::Vector3** &_anchor)
Set the anchor point.
- virtual void **SetAttribute** (**Attribute**, unsigned int _index, double _value)
Set a parameter for the joint.
- virtual void **SetAttribute** (const std::string &_key, unsigned int _index, const boost::any &_value) **GAZEBO_DEPRECATED(3.0)**
Set a non-generic parameter for the joint.
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)
Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetDamping** (unsigned int _index, const double _damping)
Set the joint damping.
- virtual void **SetForce** (unsigned int _index, double _force)
*Set the force applied to this **physics::Joint** (p. 541).*
- virtual bool **SetHighStop** (unsigned int _index, const **math::Angle** &_angle)
Set the high stop of an axis(index).
- virtual bool **SetLowStop** (unsigned int _index, const **math::Angle** &_angle)
Set the low stop of an axis(index).
- virtual bool **SetParam** (const std::string &_key, unsigned int _index, const boost::any &_value)
Set a non-generic parameter for the joint.
- virtual void **SetStiffness** (unsigned int _index, const double _stiffness)
Set the joint spring stiffness.
- virtual void **SetStiffnessDamping** (unsigned int _index, double _stiffness, double _damping, double _reference=0)
Set the joint spring stiffness.

Public Attributes

- SimTK::Constraint **constraint**
: *isValid()* if we used a constraint to model this joint.
- SimTK::Force::MobilityLinearDamper **damper** [MAX_JOINT_AXIS]
: for enforcing joint damping forces.
- SimTK::Transform **defxAB**
: default mobilizer pose
- bool **isReversed**
: if mobilizer, did it reverse parent&child? Set when we build the Simbody model.
- SimTK::Force::MobilityLinearStop **limitForce** [MAX_JOINT_AXIS]
: for enforcing joint stops Set when we build the Simbody model.
- SimTK::MobilizedBody **mobod**
: Use *isValid()* if we used a mobilizer Set when we build the Simbody model.
- bool **mustBreakLoopHere**
: Force Simbody to break a loop by using a weld constraint.
- bool **physicsInitialized**
- SimTK::Force::MobilityLinearSpring **spring** [MAX_JOINT_AXIS]
: Spring force element for enforcing joint stiffness.
- SimTK::Transform **xCB**
: child body frame to mobilizer frame
- SimTK::Transform **xPA**
: Normally $A=F$, $B=M$.

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _force)=0
: Set the force applied to this **physics::Joint** (p. 541).

Protected Attributes

- **SimbodyPhysicsPtr simbodyPhysics**
: keep a pointer to the simbody physics engine for convenience
- SimTK::MultibodySystem * **world**
: Simbody Multibody System.

Additional Inherited Members

10.209.1 Detailed Description

Base (p. 168) class for all joints.

10.209.2 Constructor & Destructor Documentation

10.209.2.1 gazebo::physics::SimbodyJoint::SimbodyJoint (BasePtr _parent)

Constructor.

10.209.2.2 virtual gazebo::physics::SimbodyJoint::~~SimbodyJoint () [virtual]

Destructor.

10.209.3 Member Function Documentation

10.209.3.1 virtual bool gazebo::physics::SimbodyJoint::AreConnected (LinkPtr *_one*, LinkPtr *_two*) const [virtual]

Determines if the two bodies are connected by a joint.

Parameters

in	<i>_one</i>	First link.
in	<i>_two</i>	Second link.

Returns

True if the two links are connected by a joint.

Implements **gazebo::physics::Joint** (p. 548).

10.209.3.2 virtual void gazebo::physics::SimbodyJoint::CacheForceTorque () [virtual]

Cache **Joint** (p. 541) Force Torque Values if necessary for physics engine.

Reimplemented from **gazebo::physics::Joint** (p. 548).

10.209.3.3 virtual void gazebo::physics::SimbodyJoint::Detach () [virtual]

Detach this joint from all links.

Reimplemented from **gazebo::physics::Joint** (p. 549).

10.209.3.4 virtual math::Vector3 gazebo::physics::SimbodyJoint::GetAnchor (unsigned int *_index*) const [virtual]

Get the anchor point.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 549).

Reimplemented in **gazebo::physics::SimbodyUniversalJoint** (p. 1018), **gazebo::physics::SimbodyHinge2Joint** (p. 952), and **gazebo::physics::SimbodyBallJoint** (p. 938).

10.209.3.5 virtual double gazebo::physics::SimbodyJoint::GetAttribute (const std::string & *_key*, unsigned int *_index*)
 [virtual]

Get a non-generic parameter for the joint.

Deprecated by GetParam

Parameters

in	<i>_key</i>	String key.
in	<i>_index</i>	Index of the axis.

Implements **gazebo::physics::Joint** (p. 551).

Reimplemented in **gazebo::physics::SimbodyScrewJoint** (p. 1004).

10.209.3.6 virtual double gazebo::physics::SimbodyJoint::GetForce (unsigned int *_index*) [virtual]

Todo : not yet implemented. Get external forces applied at this **Joint** (p. 541). Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

The force applied to an axis.

Reimplemented from **gazebo::physics::Joint** (p. 552).

10.209.3.7 virtual **JointWrench** gazebo::physics::SimbodyJoint::GetForceTorque (unsigned int *_index*) [virtual]

get internal force and torque values at a joint.

The force and torque values are returned in a **JointWrench** (p. 581) data structure. Where **JointWrench.body1Force** (p. 582) contains the force applied by the parent **Link** (p. 595) on the **Joint** (p. 541) specified in the parent **Link** (p. 595) frame, and **JointWrench.body2Force** (p. 583) contains the force applied by the child **Link** (p. 595) on the **Joint** (p. 541) specified in the child **Link** (p. 595) frame. Note that this sign convention is opposite of the reaction forces of the **Joint** (p. 541) on the Links.

FIXME TODO: change name of this function to something like: GetNegatedForceTorqueInLinkFrame and make GetForceTorque call return non-negated reaction forces in perspective **Link** (p. 595) frames.

Note that for ODE you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.

Parameters

in	<i>_index</i>	Not used right now
----	---------------	--------------------

Returns

The force and torque at the joint, see above for details on conventions.

Implements **gazebo::physics::Joint** (p. 553).

10.209.3.8 `virtual math::Angle gazebo::physics::SimbodyJoint::GetHighStop (unsigned int _index) [virtual]`

Get the high stop of an axis(index).

This function is replaced by `GetUpperLimit(unsigned int)`. If you are interested in getting the value of `dParamHiStop*`, use `GetAttribute(hi_stop, _index)`

Parameters

<i>in</i>	<i>_index</i>	Index of the axis.
-----------	---------------	--------------------

Returns

Angle of the high stop value.

Implements `gazebo::physics::Joint` (p. 553).

Reimplemented in `gazebo::physics::SimbodyBallJoint` (p.938), and `gazebo::physics::SimbodyScrewJoint` (p. 1005).

10.209.3.9 `virtual LinkPtr gazebo::physics::SimbodyJoint::GetJointLink (unsigned int _index) const [virtual]`

Get the link to which the joint is attached according the `_index`.

Parameters

<i>in</i>	<i>_index</i>	Index of the link to retrieve.
-----------	---------------	--------------------------------

Returns

Pointer to the request link. NULL if the index was invalid.

Implements `gazebo::physics::Joint` (p. 554).

10.209.3.10 `virtual math::Vector3 gazebo::physics::SimbodyJoint::GetLinkForce (unsigned int _index) const [virtual]`

Get the forces applied to the center of mass of a `physics::Link` (p. 595) due to the existence of this `Joint` (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<i>in</i>	<i>index</i>	The index of the link(0 or 1).
-----------	--------------	--------------------------------

Returns

Force applied to the link.

Implements `gazebo::physics::Joint` (p. 555).

10.209.3.11 `virtual math::Vector3 gazebo::physics::SimbodyJoint::GetLinkTorque (unsigned int _index) const [virtual]`

Get the torque applied to the center of mass of a `physics::Link` (p. 595) due to the existence of this `Joint` (p. 541).

Note that the unit of torque should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code>index</code>	The index of the link(0 or 1)
-----------------	--------------------	-------------------------------

Returns

Torque applied to the link.

Implements **gazebo::physics::Joint** (p. 555).

10.209.3.12 `virtual math::Angle gazebo::physics::SimbodyJoint::GetLowStop (unsigned int index) [virtual]`

Get the low stop of an axis(index).

This function is replaced by `GetLowerLimit(unsigned int)`. If you are interested in getting the value of `dParamHiStop*`, use `GetAttribute(hi_stop, _index)`

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Angle of the low stop value.

Implements **gazebo::physics::Joint** (p. 556).

Reimplemented in **gazebo::physics::SimbodyBallJoint** (p. 939), and **gazebo::physics::SimbodyScrewJoint** (p. 1005).

10.209.3.13 `virtual double gazebo::physics::SimbodyJoint::GetParam (const std::string & key, unsigned int index) [virtual]`

Get a non-generic parameter for the joint.

Parameters

<code>in</code>	<code>_key</code>	String key.
<code>in</code>	<code>_index</code>	Index of the axis.

Implements **gazebo::physics::Joint** (p. 557).

Reimplemented in **gazebo::physics::SimbodyScrewJoint** (p. 1006).

10.209.3.14 `virtual void gazebo::physics::SimbodyJoint::Load (sdf::ElementPtr sdf) [virtual]`

Load **physics::Joint** (p. 541) from a SDF `sdf::Element`.

Parameters

<code>in</code>	<code>_sdf</code>	SDF values to load from.
-----------------	-------------------	--------------------------

Reimplemented from **gazebo::physics::Joint** (p. 560).

Reimplemented in `gazebo::physics::SimbodySliderJoint` (p. 1013), `gazebo::physics::BallJoint` < `SimbodyJoint` > (p. 168), `gazebo::physics::UniversalJoint` < `SimbodyJoint` > (p. 1136), `gazebo::physics::Hinge2Joint` < `SimbodyJoint` > (p. 513), `gazebo::physics::ScrewJoint` < `SimbodyJoint` > (p. 898), `gazebo::physics::HingeJoint` < `SimbodyJoint` > (p. 514), `gazebo::physics::SliderJoint` < `SimbodyJoint` > (p. 1045), `gazebo::physics::SimbodyHingeJoint` (p. 958), `gazebo::physics::SimbodyUniversalJoint` (p. 1019), `gazebo::physics::SimbodyHinge2Joint` (p. 954), `gazebo::physics::SimbodyScrewJoint` (p. 1007), and `gazebo::physics::SimbodyBallJoint` (p. 940).

10.209.3.15 `virtual void gazebo::physics::SimbodyJoint::Reset () [virtual]`

Reset the joint.

Reimplemented from `gazebo::physics::Joint` (p. 560).

10.209.3.16 `virtual void gazebo::physics::SimbodyJoint::RestoreSimbodyState (SimTK::State & _state) [virtual]`

Reimplemented in `gazebo::physics::SimbodyHingeJoint` (p. 959).

10.209.3.17 `virtual void gazebo::physics::SimbodyJoint::SaveSimbodyState (const SimTK::State & _state) [virtual]`

Reimplemented in `gazebo::physics::SimbodyHingeJoint` (p. 959).

10.209.3.18 `virtual void gazebo::physics::SimbodyJoint::SetAnchor (unsigned int _index, const gazebo::math::Vector3 & _anchor) [virtual]`

Set the anchor point.

Parameters

in	<code>_index</code>	Indx of the axis.
in	<code>_anchor</code>	Anchor value.

Implements `gazebo::physics::Joint` (p. 561).

10.209.3.19 `virtual void gazebo::physics::SimbodyJoint::SetAttribute (Attribute , unsigned int _index, double _value) [virtual]`

Set a parameter for the joint.

10.209.3.20 `virtual void gazebo::physics::SimbodyJoint::SetAttribute (const std::string & _key, unsigned int _index, const boost::any & _value) [virtual]`

Set a non-generic parameter for the joint.

replaces `SetAttribute(Attribute, int, double)` Deprecated by `bool SetParam`

Parameters

in	<code>_key</code>	String key.
in	<code>_index</code>	Index of the axis.
in	<code>_value</code>	Value of the attribute.

Implements **gazebo::physics::Joint** (p. 561).

Reimplemented in **gazebo::physics::SimbodyScrewJoint** (p. 1007).

10.209.3.21 `virtual void gazebo::physics::SimbodyJoint::SetAxis (unsigned int _index, const math::Vector3 & _axis)`
`[virtual]`

Set the axis of rotation where axis is specified in local joint frame.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to set.
<code>in</code>	<code>_axis</code>	Vector in local joint frame of axis direction (must have length greater than zero).

Implements **gazebo::physics::Joint** (p. 561).

Reimplemented in **gazebo::physics::SimbodyBallJoint** (p. 940), **gazebo::physics::SimbodyUniversalJoint** (p. 1019), **gazebo::physics::SimbodyHinge2Joint** (p. 954), **gazebo::physics::SimbodyHingeJoint** (p. 959), **gazebo::physics::SimbodyScrewJoint** (p. 1007), and **gazebo::physics::SimbodySliderJoint** (p. 1013).

10.209.3.22 `virtual void gazebo::physics::SimbodyJoint::SetDamping (unsigned int _index, const double _damping)`
`[virtual]`

Set the joint damping.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to set, currently ignored, to be implemented.
<code>in</code>	<code>_damping</code>	Damping value for the axis.

Implements **gazebo::physics::Joint** (p. 562).

10.209.3.23 `virtual void gazebo::physics::SimbodyJoint::SetForce (unsigned int _index, double _effort)` `[virtual]`

Set the force applied to this **physics::Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForce to the same joint in the same time step will accumulate forces on that **Joint** (p. 541)). Forces are truncated by effortLimit before applied.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_effort</code>	Force value.

Implements **gazebo::physics::Joint** (p. 562).

10.209.3.24 `virtual void gazebo::physics::SimbodyJoint::SetForceImpl (unsigned int _index, double _force)` `[protected]`,
`[pure virtual]`

Set the force applied to this **physics::Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 541)).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_force</code>	Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 1008), **gazebo::physics::SimbodyBallJoint** (p. 940), **gazebo::physics::SimbodyUniversalJoint** (p. 1019), **gazebo::physics::SimbodyHinge2Joint** (p. 954), **gazebo::physics::SimbodyHingeJoint** (p. 959), and **gazebo::physics::SimbodySliderJoint** (p. 1013).

10.209.3.25 `virtual bool gazebo::physics::SimbodyJoint::SetHighStop (unsigned int _index, const math::Angle & _angle)`
[virtual]

Set the high stop of an axis(index).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_angle</code>	High stop angle.

Reimplemented from **gazebo::physics::Joint** (p. 562).

Reimplemented in **gazebo::physics::SimbodyBallJoint** (p. 940), and **gazebo::physics::SimbodyScrewJoint** (p. 1008).

10.209.3.26 `virtual bool gazebo::physics::SimbodyJoint::SetLowStop (unsigned int _index, const math::Angle & _angle)`
[virtual]

Set the low stop of an axis(index).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_angle</code>	Low stop angle.

Reimplemented from **gazebo::physics::Joint** (p. 563).

Reimplemented in **gazebo::physics::SimbodyBallJoint** (p. 941), and **gazebo::physics::SimbodyScrewJoint** (p. 1008).

10.209.3.27 `virtual bool gazebo::physics::SimbodyJoint::SetParam (const std::string & _key, unsigned int _index, const boost::any & _value)` [virtual]

Set a non-generic parameter for the joint.

replaces SetAttribute(Attribute, int, double)

Parameters

in	<code>_key</code>	String key.
in	<code>_index</code>	Index of the axis.
in	<code>_value</code>	Value of the attribute.

Implements **gazebo::physics::Joint** (p. 564).

Reimplemented in **gazebo::physics::SimbodyScrewJoint** (p. 1009).

10.209.3.28 `virtual void gazebo::physics::SimbodyJoint::SetStiffness (unsigned int _index, const double _stiffness)`
`[virtual]`

Set the joint spring stiffness.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to set, currently ignored, to be implemented.
<code>in</code>	<code>_stiffness</code>	Spring stiffness value for the axis. : rename to SetSpringStiffness()

Implements **gazebo::physics::Joint** (p. 564).

10.209.3.29 `virtual void gazebo::physics::SimbodyJoint::SetStiffnessDamping (unsigned int _index, double _stiffness, double _damping, double _reference = 0)` `[virtual]`

Set the joint spring stiffness.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to set, currently ignored, to be implemented.
<code>in</code>	<code>_stiffness</code>	Stiffness value for the axis.
<code>in</code>	<code>_reference</code>	Spring zero load reference position. : rename to SetSpringStiffnessDamping()

Implements **gazebo::physics::Joint** (p. 565).

10.209.4 Member Data Documentation

10.209.4.1 `SimTK::Constraint gazebo::physics::SimbodyJoint::constraint`

: isValid() if we used a constraint to model this joint.

Set when we build the Simbody model. How this joint was modeled in the Simbody System. We used either a mobilizer or a constraint, but not both. The type of either one is the same as the joint type above.

10.209.4.2 `SimTK::Force::MobilityLinearDamper gazebo::physics::SimbodyJoint::damper[MAX_JOINT_AXIS]`

: for enforcing joint damping forces.

Set when we build the Simbody model. : Also, consider moving this into individual joint type subclass so we can specify custom dampers for special joints like ball joints.

10.209.4.3 `SimTK::Transform gazebo::physics::SimbodyJoint::defxAB`

default mobilizer pose

10.209.4.4 `bool gazebo::physics::SimbodyJoint::isReversed`

: if mobilizer, did it reverse parent&child? Set when we build the Simbody model.

10.209.4.5 `SimTK::Force::MobilityLinearStop gazebo::physics::SimbodyJoint::limitForce[MAX_JOINT_AXIS]`

: for enforcing joint stops Set when we build the Simbody model.

: Also, consider moving this into individual joint type subclass so we can specify custom dampers for special joints like ball joints. Assuming this is not used for BallJoints it's ok here for now.

10.209.4.6 `SimTK::MobilizedBody gazebo::physics::SimbodyJoint::mobod`

Use `isValid()` if we used a mobilizer Set when we build the Simbody model.

How this joint was modeled in the Simbody System. We used either a mobilizer or a constraint, but not both. The type of either one is the same as the joint type above.

10.209.4.7 `bool gazebo::physics::SimbodyJoint::mustBreakLoopHere`

Force Simbody to break a loop by using a weld constraint.

This flag is needed by `SimbodyPhysics::MultibodyGraphMaker`, so kept public.

10.209.4.8 `bool gazebo::physics::SimbodyJoint::physicsInitialized`**10.209.4.9** `SimbodyPhysicsPtr gazebo::physics::SimbodyJoint::simbodyPhysics` `[protected]`

keep a pointer to the simbody physics engine for convenience

10.209.4.10 `SimTK::Force::MobilityLinearSpring gazebo::physics::SimbodyJoint::spring[MAX_JOINT_AXIS]`

: Spring force element for enforcing joint stiffness.

The element is assigned when constructing Simbody model in `SimbodyPhysics::AddDynamicModelToSimbodySystem`.

: Also, consider moving this into individual joint type subclass so we can specify custom springs for special joints like ball joints.

10.209.4.11 `SimTK::MultibodySystem* gazebo::physics::SimbodyJoint::world` `[protected]`

Simbody Multibody System.

10.209.4.12 `SimTK::Transform gazebo::physics::SimbodyJoint::xCB`

child body frame to mobilizer frame

10.209.4.13 `SimTK::Transform gazebo::physics::SimbodyJoint::xPA`

Normally $A=F$, $B=M$.

But if reversed, then $B=F$, $A=M$. parent body frame to mobilizer frame

The documentation for this class was generated from the following file:

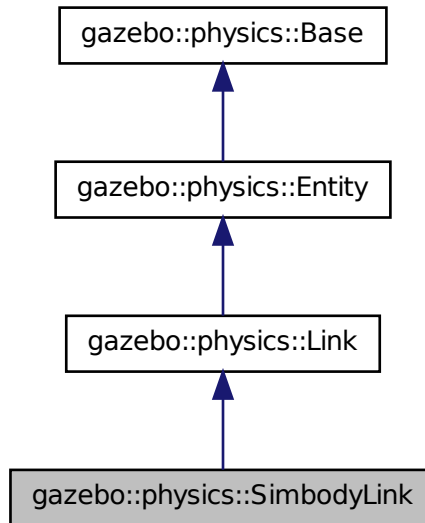
- **SimbodyJoint.hh**

10.210 gazebo::physics::SimbodyLink Class Reference

Simbody **Link** (p. 595) class.

```
#include <SimbodyLink.hh>
```

Inheritance diagram for gazebo::physics::SimbodyLink:



Public Member Functions

- **SimbodyLink** (**EntityPtr** _parent)
Constructor.
- virtual **~SimbodyLink** ()
Destructor.
- virtual void **AddForce** (const **math::Vector3** &_force)
Add a force to the body.
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relpos)
Add a force to the body at position expressed to the body's own frame of reference.
- virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)
Add a force to the body using a global position.
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)
Add a force to the body, components are relative to the body's own frame of reference.
- virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)
Add a torque to the body, components are relative to the body's own frame of reference.
- virtual void **AddTorque** (const **math::Vector3** &_torque)
Add a torque to the body.
- virtual void **Fini** ()

Finalize the body.

- SimTK::MassProperties **GetEffectiveMassProps** (int _numFragments) const
- virtual bool **GetEnabled** () const
 - Get whether this body is enabled in the physics engine.*
- virtual bool **GetGravityMode** () const
 - Get the gravity mode.*
- SimTK::MassProperties **GetMassProperties** () const
 - Convert Gazebo Inertia to Simbody MassProperties Where Simbody MassProperties contains mass, center of mass location, and unit inertia about body origin.*
- virtual **math::Vector3** **GetWorldAngularVel** () const
 - Get the angular velocity of the entity in the world frame.*
- virtual **math::Vector3** **GetWorldCoGLinearVel** () const
 - Get the linear velocity at the body's center of gravity in the world frame.*
- virtual **math::Vector3** **GetWorldForce** () const
 - Get the force applied to the body in the world frame.*
- virtual **math::Vector3** **GetWorldLinearVel** (const **math::Vector3** &_vector3) const
 - Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame.*
- virtual **math::Vector3** **GetWorldLinearVel** (const **math::Vector3** &_offset, const **math::Quaternion** &_q) const
 - Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.*
- virtual **math::Vector3** **GetWorldTorque** () const
 - Get the torque applied to the body in the world frame.*
- virtual void **Init** ()
 - Initialize the body.*
- virtual void **Load** (sdf::ElementPtr _ptr)
 - Load the body based on an SDF element.*
- virtual void **OnPoseChange** ()
 - This function is called when the entity's (or one of its parents) pose of the parent has changed.*
- virtual void **RestoreSimbodyState** (SimTK::State &_state)
- virtual void **SaveSimbodyState** (const SimTK::State &_state)
- virtual void **SetAngularDamping** (double _damping)
 - Set the angular damping factor.*
- virtual void **SetAngularVel** (const **math::Vector3** &_vel)
 - Set the angular velocity of the body.*
- virtual void **SetAutoDisable** (bool _disable)
 - Allow the link to auto disable.*
- void **SetDirtyPose** (const **math::Pose** &_pose)
- virtual void **SetEnabled** (bool enable) const
 - Set whether this body is enabled.*
- virtual void **SetForce** (const **math::Vector3** &_force)
 - Set the force applied to the body.*
- virtual void **SetGravityMode** (bool _mode)
 - Set whether gravity affects this body.*
- virtual void **SetLinearDamping** (double _damping)
 - Set the linear damping factor.*
- virtual void **SetLinearVel** (const **math::Vector3** &_vel)
 - Set the linear velocity of the body.*
- virtual void **SetLinkStatic** (bool _static)

If the inboard body of this link is ground, simply lock the inboard joint to freeze it to ground.

- virtual void **SetSelfCollide** (bool `_collide`)
Set whether this body will collide with others in the model.
- virtual void **SetTorque** (const `math::Vector3` & `_force`)
Set the torque applied to the body.

Public Attributes

- SimTK::MobilizedBody **masterMobod**
- bool **mustBeBaseLink**
: Force this link to be a base body, where its inboard body is the world with 6DOF.
- bool **physicsInitialized**
- std::vector< SimTK::MobilizedBody > **slaveMobods**
- std::vector
< SimTK::Constraint::Weld > **slaveWelds**

Additional Inherited Members

10.210.1 Detailed Description

Simbody **Link** (p. 595) class.

10.210.2 Constructor & Destructor Documentation

10.210.2.1 gazebo::physics::SimbodyLink::SimbodyLink (EntityPtr *_parent*)

Constructor.

10.210.2.2 virtual gazebo::physics::SimbodyLink::~~SimbodyLink () [virtual]

Destructor.

10.210.3 Member Function Documentation

10.210.3.1 virtual void gazebo::physics::SimbodyLink::AddForce (const `math::Vector3` & *_force*) [virtual]

Add a force to the body.

Parameters

<code>in</code>	<code>_force</code>	Force to add.
-----------------	---------------------	---------------

Implements `gazebo::physics::Link` (p. 601).

10.210.3.2 virtual void gazebo::physics::SimbodyLink::AddForceAtRelativePosition (const `math::Vector3` & *_force*, const `math::Vector3` & *_relPos*) [virtual]

Add a force to the body at position expressed to the body's own frame of reference.

Parameters

in	<i>_force</i>	Force to add.
in	<i>_relPos</i>	Position on the link to add the force.

Implements **gazebo::physics::Link** (p. 601).

10.210.3.3 `virtual void gazebo::physics::SimbodyLink::AddForceAtWorldPosition (const math::Vector3 & _force, const math::Vector3 & _pos) [virtual]`

Add a force to the body using a global position.

Parameters

in	<i>_force</i>	Force to add.
in	<i>_pos</i>	Position in global coord frame to add the force.

Implements **gazebo::physics::Link** (p. 602).

10.210.3.4 `virtual void gazebo::physics::SimbodyLink::AddRelativeForce (const math::Vector3 & _force) [virtual]`

Add a force to the body, components are relative to the body's own frame of reference.

Parameters

in	<i>_force</i>	Force to add.
----	---------------	---------------

Implements **gazebo::physics::Link** (p. 602).

10.210.3.5 `virtual void gazebo::physics::SimbodyLink::AddRelativeTorque (const math::Vector3 & _torque) [virtual]`

Add a torque to the body, components are relative to the body's own frame of reference.

Parameters

in	<i>_torque</i>	Torque value to add.
----	----------------	----------------------

Implements **gazebo::physics::Link** (p. 602).

10.210.3.6 `virtual void gazebo::physics::SimbodyLink::AddTorque (const math::Vector3 & _torque) [virtual]`

Add a torque to the body.

Parameters

in	<i>_torque</i>	Torque value to add to the link.
----	----------------	----------------------------------

Implements **gazebo::physics::Link** (p. 602).

10.210.3.7 `virtual void gazebo::physics::SimbodyLink::Fini () [virtual]`

Finalize the body.

Reimplemented from `gazebo::physics::Link` (p. 604).

10.210.3.8 `SimTK::MassProperties gazebo::physics::SimbodyLink::GetEffectiveMassProps (int _numFragments) const`

10.210.3.9 `virtual bool gazebo::physics::SimbodyLink::GetEnabled () const [virtual]`

Get whether this body is enabled in the physics engine.

Returns

True if the link is enabled.

Implements `gazebo::physics::Link` (p. 605).

10.210.3.10 `virtual bool gazebo::physics::SimbodyLink::GetGravityMode () const [virtual]`

Get the gravity mode.

Returns

True if gravity is enabled.

Implements `gazebo::physics::Link` (p. 606).

10.210.3.11 `SimTK::MassProperties gazebo::physics::SimbodyLink::GetMassProperties () const`

Convert Gazebo Inertia to Simbody MassProperties Where Simbody MassProperties contains mass, center of mass location, and unit inertia about body origin.

10.210.3.12 `virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldAngularVel () const [virtual]`

Get the angular velocity of the entity in the world frame.

Returns

A `math::Vector3` (p. 1165) for the velocity.

Reimplemented from `gazebo::physics::Entity` (p. 410).

10.210.3.13 `virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldCoGLinearVel () const [virtual]`

Get the linear velocity at the body's center of gravity in the world frame.

Returns

Linear velocity at the body's center of gravity in the world frame.

Implements `gazebo::physics::Link` (p. 609).

10.210.3.14 `virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldForce () const [virtual]`

Get the force applied to the body in the world frame.

Returns

Force applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 610).

10.210.3.15 `virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldLinearVel (const math::Vector3 & _offset) const [virtual]`

Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame.

If no offset is given, the velocity at the origin of the **Link** (p. 595) frame will be returned.

Parameters

<code>in</code>	<code>_offset</code>	Offset of the point from the origin of the Link (p. 595) frame, expressed in the body-fixed frame.
-----------------	----------------------	---

Returns

Linear velocity of the point on the body

Implements **gazebo::physics::Link** (p. 611).

10.210.3.16 `virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldLinearVel (const math::Vector3 & _offset, const math::Quaternion & _q) const [virtual]`

Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.

Parameters

<code>in</code>	<code>_offset</code>	Offset from the origin of the link frame expressed in a frame defined by <code>_q</code> .
<code>in</code>	<code>_q</code>	Describes the rotation of a reference frame relative to the world reference frame.

Returns

Linear velocity of the point on the body in the world frame.

Implements **gazebo::physics::Link** (p. 611).

10.210.3.17 `virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldTorque () const [virtual]`

Get the torque applied to the body in the world frame.

Returns

Torque applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 611).

10.210.3.18 `virtual void gazebo::physics::SimbodyLink::Init () [virtual]`

Initialize the body.

Reimplemented from **`gazebo::physics::Link`** (p. 611).

10.210.3.19 `virtual void gazebo::physics::SimbodyLink::Load (sdf::ElementPtr _sdf) [virtual]`

Load the body based on an SDF element.

Parameters

<code>in</code>	<code>_sdf</code>	SDF parameters.
-----------------	-------------------	-----------------

Reimplemented from **`gazebo::physics::Link`** (p. 612).

10.210.3.20 `virtual void gazebo::physics::SimbodyLink::OnPoseChange () [virtual]`

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Reimplemented from **`gazebo::physics::Link`** (p. 612).

10.210.3.21 `virtual void gazebo::physics::SimbodyLink::RestoreSimbodyState (SimTK::State & _state) [virtual]`

10.210.3.22 `virtual void gazebo::physics::SimbodyLink::SaveSimbodyState (const SimTK::State & _state) [virtual]`

10.210.3.23 `virtual void gazebo::physics::SimbodyLink::SetAngularDamping (double _damping) [virtual]`

Set the angular damping factor.

Parameters

<code>in</code>	<code>_damping</code>	Angular damping factor.
-----------------	-----------------------	-------------------------

Implements **`gazebo::physics::Link`** (p. 613).

10.210.3.24 `virtual void gazebo::physics::SimbodyLink::SetAngularVel (const math::Vector3 & _vel) [virtual]`

Set the angular velocity of the body.

Parameters

<code>in</code>	<code>_vel</code>	Angular velocity.
-----------------	-------------------	-------------------

Implements **`gazebo::physics::Link`** (p. 613).

10.210.3.25 `virtual void gazebo::physics::SimbodyLink::SetAutoDisable (bool _disable) [virtual]`

Allow the link to auto disable.

Parameters

in	<code>_disable</code>	If true, the link is allowed to auto disable.
----	-----------------------	---

Implements **gazebo::physics::Link** (p. 613).

10.210.3.26 `void gazebo::physics::SimbodyLink::SetDirtyPose (const math::Pose & _pose)`

10.210.3.27 `virtual void gazebo::physics::SimbodyLink::SetEnabled (bool _enable) const` [virtual]

Set whether this body is enabled.

Parameters

in	<code>_enable</code>	True to enable the link in the physics engine.
----	----------------------	--

Implements **gazebo::physics::Link** (p. 614).

10.210.3.28 `virtual void gazebo::physics::SimbodyLink::SetForce (const math::Vector3 & _force)` [virtual]

Set the force applied to the body.

Parameters

in	<code>_force</code>	Force value.
----	---------------------	--------------

Implements **gazebo::physics::Link** (p. 614).

10.210.3.29 `virtual void gazebo::physics::SimbodyLink::SetGravityMode (bool _mode)` [virtual]

Set whether gravity affects this body.

Parameters

in	<code>_mode</code>	True to enable gravity.
----	--------------------	-------------------------

Implements **gazebo::physics::Link** (p. 614).

10.210.3.30 `virtual void gazebo::physics::SimbodyLink::SetLinearDamping (double _damping)` [virtual]

Set the linear damping factor.

Parameters

in	<code>_damping</code>	Linear damping factor.
----	-----------------------	------------------------

Implements **gazebo::physics::Link** (p. 615).

10.210.3.31 `virtual void gazebo::physics::SimbodyLink::SetLinearVel (const math::Vector3 & _vel)` [virtual]

Set the linear velocity of the body.

Parameters

in	<code>_vel</code>	Linear velocity.
----	-------------------	------------------

Implements **gazebo::physics::Link** (p. 615).

10.210.3.32 `virtual void gazebo::physics::SimbodyLink::SetLinkStatic (bool _static) [virtual]`

If the inboard body of this link is ground, simply lock the inboard joint to freeze it to ground. Otherwise, add a weld constraint to simulate freeze to ground effect.

Parameters

in	<code>_static</code>	if true, freeze link to ground. Otherwise unfreeze link.
----	----------------------	--

Implements **gazebo::physics::Link** (p. 615).

10.210.3.33 `virtual void gazebo::physics::SimbodyLink::SetSelfCollide (bool _collide) [virtual]`

Set whether this body will collide with others in the model.

Parameters

in	<code>_collid</code>	True to enable collisions.
----	----------------------	----------------------------

Implements **gazebo::physics::Link** (p. 616).

10.210.3.34 `virtual void gazebo::physics::SimbodyLink::SetTorque (const math::Vector3 & _torque) [virtual]`

Set the torque applied to the body.

Parameters

in	<code>_torque</code>	Torque value.
----	----------------------	---------------

Implements **gazebo::physics::Link** (p. 617).

10.210.4 Member Data Documentation

10.210.4.1 `SimTK::MobilizedBody gazebo::physics::SimbodyLink::masterMobod`

10.210.4.2 `bool gazebo::physics::SimbodyLink::mustBeBaseLink`

: Force this link to be a base body, where its inboard body is the world with 6DOF.

10.210.4.3 `bool gazebo::physics::SimbodyLink::physicsInitialized`

10.210.4.4 `std::vector<SimTK::MobilizedBody> gazebo::physics::SimbodyLink::slaveMobods`

10.210.4.5 `std::vector<SimTK::Constraint::Weld>` gazebo::physics::SimbodyLink::slaveWelds

The documentation for this class was generated from the following file:

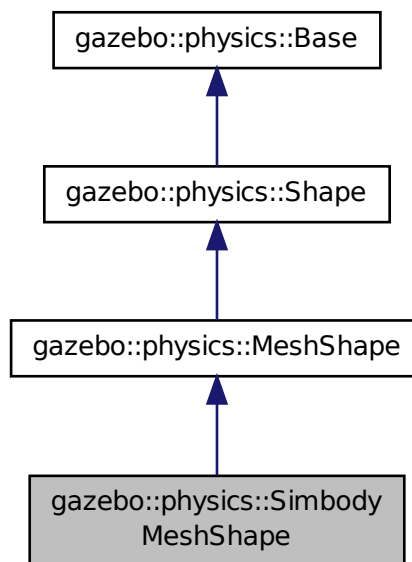
- **SimbodyLink.hh**

10.211 gazebo::physics::SimbodyMeshShape Class Reference

Triangle mesh collision.

```
#include <SimbodyMeshShape.hh>
```

Inheritance diagram for gazebo::physics::SimbodyMeshShape:



Public Member Functions

- **SimbodyMeshShape** (`CollisionPtr _parent`)
Constructor.
- virtual `~SimbodyMeshShape` ()
Destructor.
- virtual void **Load** (`sdf::ElementPtr _sdf`)
Load.

Protected Member Functions

- virtual void **Init** ()

Initialize the shape.

Additional Inherited Members

10.211.1 Detailed Description

Triangle mesh collision.

10.211.2 Constructor & Destructor Documentation

10.211.2.1 `gazebo::physics::SimbodyMeshShape::SimbodyMeshShape (CollisionPtr _parent)`

Constructor.

10.211.2.2 `virtual gazebo::physics::SimbodyMeshShape::~~SimbodyMeshShape () [virtual]`

Destructor.

10.211.3 Member Function Documentation

10.211.3.1 `virtual void gazebo::physics::SimbodyMeshShape::Init () [protected],[virtual]`

Initialize the shape.

Reimplemented from `gazebo::physics::MeshShape` (p. 677).

10.211.3.2 `virtual void gazebo::physics::SimbodyMeshShape::Load (sdf::ElementPtr _sdf) [virtual]`

Load.

Parameters

<code>in</code>	<code>node</code>	Pointer to an SDF parameters
-----------------	-------------------	------------------------------

Reimplemented from `gazebo::physics::Base` (p. 177).

The documentation for this class was generated from the following file:

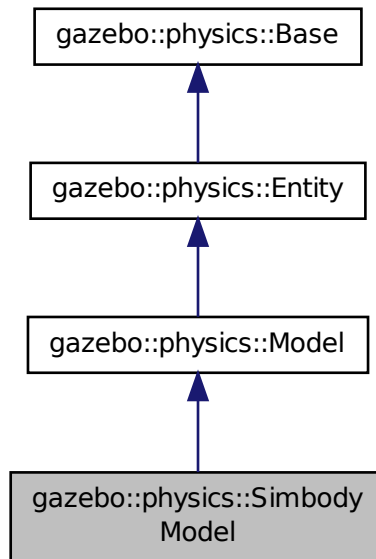
- `SimbodyMeshShape.hh`

10.212 gazebo::physics::SimbodyModel Class Reference

A model is a collection of links, joints, and plugins.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::SimbodyModel:



Public Member Functions

- **SimbodyModel** (**BasePtr** _parent)
Constructor.
- virtual **~SimbodyModel** ()
Destructor.
- virtual void **Init** ()
Initialize the model.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the model.

Additional Inherited Members

10.212.1 Detailed Description

A model is a collection of links, joints, and plugins.

10.212.2 Constructor & Destructor Documentation

10.212.2.1 gazebo::physics::SimbodyModel::SimbodyModel (**BasePtr** _parent) [explicit]

Constructor.

Parameters

in	<code>_parent</code>	Parent object.
----	----------------------	----------------

10.212.2.2 virtual `gazebo::physics::SimbodyModel::~~SimbodyModel()` [virtual]

Destructor.

10.212.3 Member Function Documentation

10.212.3.1 virtual void `gazebo::physics::SimbodyModel::Init()` [virtual]

Initialize the model.

Reimplemented from `gazebo::physics::Model` (p. 688).

10.212.3.2 virtual void `gazebo::physics::SimbodyModel::Load(sdf::ElementPtr _sdf)` [virtual]

Load the model.

Parameters

in	<code>_sdf</code>	SDF parameters to load from.
----	-------------------	------------------------------

Reimplemented from `gazebo::physics::Model` (p. 688).

The documentation for this class was generated from the following file:

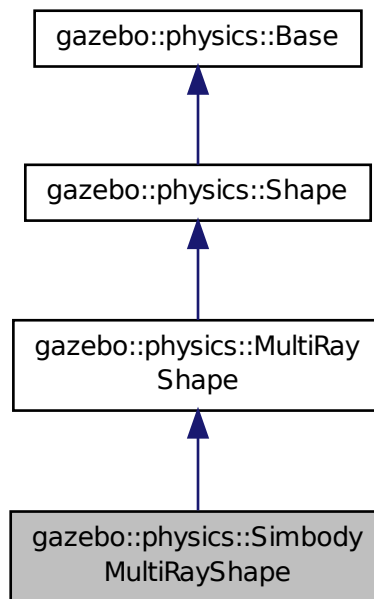
- `SimbodyModel.hh`

10.213 gazebo::physics::SimbodyMultiRayShape Class Reference

Simbody specific version of `MultiRayShape` (p. 723).

```
#include <SimbodyMultiRayShape.hh>
```

Inheritance diagram for gazebo::physics::SimbodyMultiRayShape:



Public Member Functions

- **SimbodyMultiRayShape** (**CollisionPtr** parent)
Constructor.
- virtual **~SimbodyMultiRayShape** ()
Destructor.
- virtual void **UpdateRays** ()
Physics engine specific method for updating the rays.

Protected Member Functions

- virtual void **AddRay** (const **math::Vector3** &_start, const **math::Vector3** &_end)
Add a ray to the collision.

Additional Inherited Members

10.213.1 Detailed Description

Simbody specific version of **MultiRayShape** (p. 723).

10.213.2 Constructor & Destructor Documentation

10.213.2.1 `gazebo::physics::SimbodyMultiRayShape::SimbodyMultiRayShape (CollisionPtr parent)`

Constructor.

10.213.2.2 `virtual gazebo::physics::SimbodyMultiRayShape::~~SimbodyMultiRayShape () [virtual]`

Destructor.

10.213.3 Member Function Documentation

10.213.3.1 `virtual void gazebo::physics::SimbodyMultiRayShape::AddRay (const math::Vector3 & _start, const math::Vector3 & _end) [protected],[virtual]`

Add a ray to the collision.

Parameters

<code>in</code>	<code>_start</code>	Start of the ray.
<code>in</code>	<code>_end</code>	End of the ray.

Reimplemented from `gazebo::physics::MultiRayShape` (p. 726).

10.213.3.2 `virtual void gazebo::physics::SimbodyMultiRayShape::UpdateRays () [virtual]`

Physics engine specific method for updating the rays.

Implements `gazebo::physics::MultiRayShape` (p. 730).

The documentation for this class was generated from the following file:

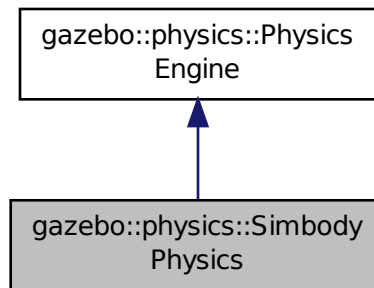
- `SimbodyMultiRayShape.hh`

10.214 gazebo::physics::SimbodyPhysics Class Reference

Simbody physics engine.

```
#include <SimbodyPhysics.hh>
```

Inheritance diagram for gazebo::physics::SimbodyPhysics:



Public Member Functions

- **SimbodyPhysics** (**WorldPtr** _world)
 - Constructor.*
- virtual \sim **SimbodyPhysics** ()
 - Destructor.*
- virtual **CollisionPtr** **CreateCollision** (const std::string &_type, **LinkPtr** _body)
 - Create a collision.*
- virtual **JointPtr** **CreateJoint** (const std::string &_type, **ModelPtr** _parent)
 - Create a new joint.*
- virtual **LinkPtr** **CreateLink** (**ModelPtr** _parent)
 - Create a new body.*
- virtual **ModelPtr** **CreateModel** (**BasePtr** _parent)
 - Create a new model.*
- virtual **ShapePtr** **CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)
 - Create a **physics::Shape** (p. 932) object.*
- virtual void **DebugPrint** () const
 - Debug print out of the physic engine state.*
- virtual void **Fini** ()
 - Finilize the physics engine.*
- SimTK::MultibodySystem * **GetDynamicsWorld** () const
 - Register a joint with the dynamics world.*
- virtual boost::any **GetParam** (const std::string &_key) const
 - Get an parameter of the physics engine.*
- virtual std::string **GetType** () const
 - Return the physics engine type (ode|bullet|dart|simbody).*
- virtual void **Init** ()
 - Initialize the physics engine.*
- virtual void **InitForThread** ()

- Init the engine for threads.*
- void **InitModel** (const **physics::ModelPtr** _model)
 - Add a **Model** (p. 678) to the Simbody system.*
- virtual void **Load** (sdf::ElementPtr _sdf)
 - Load the physics engine.*
- virtual void **Reset** ()
 - Rest the physics engine.*
- virtual void **SetGravity** (const **gazebo::math::Vector3** &_gravity)
 - Set the gavity vector.*
- virtual bool **SetParam** (const std::string &_key, const boost::any &_value)
 - Set a parameter of the physics engine.*
- virtual void **SetSeed** (uint32_t _seed)
 - Set the random number seed for the physics engine.*
- virtual void **UpdateCollision** ()
 - Update the physics engine collision.*
- virtual void **UpdatePhysics** ()
 - Update the physics engine.*

Static Public Member Functions

- static SimTK::Transform **GetPose** (sdf::ElementPtr _element)
 - If the given element contains a <pose> element, return it as a Transform.*
- static std::string **GetTypeString** (unsigned int _type)
 - Convert **Base::GetType()** (p. 176) to string, this is needed by the MultibodyGraphMaker.*
- static std::string **GetTypeString (physics::Base::EntityType _type)**
 - Convert **Base::GetType()** (p. 176) to string, this is needed by the MultibodyGraphMaker.*
- static SimTK::Transform **Pose2Transform** (const **math::Pose** &_pose)
 - Convert the given pose in x,y,z,thetax,thetay,thetaz format to a Simbody Transform.*
- static SimTK::Quaternion **QuadToQuad** (const **math::Quaternion** &_q)
 - Convert **gazebo::math::Quaternion** (p. 824) to SimTK::Quaternion.*
- static **math::Quaternion QuadToQuad** (const SimTK::Quaternion &_q)
 - Convert SimTK::Quaternion to **gazebo::math::Quaternion** (p. 824).*
- static **math::Pose Transform2Pose** (const SimTK::Transform &_xAB)
 - Convert a Simbody transform to a pose in x,y,z, thetax,thetay,thetaz format.*
- static **math::Vector3 Vec3ToVector3** (const SimTK::Vec3 &_v)
 - Convert SimTK::Vec3 to **gazebo::math::Vector3** (p. 1165).*
- static SimTK::Vec3 **Vector3ToVec3** (const **math::Vector3** &_v)
 - Convert **gazebo::math::Vector3** (p. 1165) to SimTK::Vec3.*

Public Attributes

- SimTK::CompliantContactSubsystem **contact**
- SimTK::Force::DiscreteForces **discreteForces**
- SimTK::GeneralForceSubsystem **forces**
- SimTK::Force::Gravity **gravity**
- SimTK::Integrator * **integ**
- SimTK::SimbodyMatterSubsystem **matter**

- bool **simbodyPhysicsInitialized**
true if initialized
- bool **simbodyPhysicsStepped**
- SimTK::MultibodySystem **system**
- SimTK::ContactTrackerSubsystem **tracker**

Protected Member Functions

- virtual void **OnPhysicsMsg** (ConstPhysicsPtr &_msg)
virtual callback for gztopic "~/physics".
- virtual void **OnRequest** (ConstRequestPtr &_msg)
virtual callback for gztopic "~/request".

Additional Inherited Members

10.214.1 Detailed Description

Simbody physics engine.

10.214.2 Constructor & Destructor Documentation

10.214.2.1 gazebo::physics::SimbodyPhysics::SimbodyPhysics (WorldPtr _world)

Constructor.

10.214.2.2 virtual gazebo::physics::SimbodyPhysics::~~SimbodyPhysics () [virtual]

Destructor.

10.214.3 Member Function Documentation

10.214.3.1 virtual CollisionPtr gazebo::physics::SimbodyPhysics::CreateCollision (const std::string & _shapeType, LinkPtr _link) [virtual]

Create a collision.

Parameters

in	<code>_shapeType</code>	Type of collision to create.
in	<code>_link</code>	Parent link.

Implements **gazebo::physics::PhysicsEngine** (p. 770).

10.214.3.2 virtual JointPtr gazebo::physics::SimbodyPhysics::CreateJoint (const std::string & _type, ModelPtr _parent) [virtual]

Create a new joint.

Parameters

in	<code>_type</code>	Type of joint to create.
in	<code>_parent</code>	Model (p. 678) parent.

Implements **gazebo::physics::PhysicsEngine** (p. 771).

10.214.3.3 virtual `LinkPtr gazebo::physics::SimbodyPhysics::CreateLink (ModelPtr _parent)` [virtual]

Create a new body.

Parameters

in	<code>_parent</code>	Parent model for the link.
----	----------------------	----------------------------

Implements **gazebo::physics::PhysicsEngine** (p. 771).

10.214.3.4 virtual `ModelPtr gazebo::physics::SimbodyPhysics::CreateModel (BasePtr _base)` [virtual]

Create a new model.

Parameters

in	<code>_base</code>	Boost shared pointer to a new model.
----	--------------------	--------------------------------------

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 771).

10.214.3.5 virtual `ShapePtr gazebo::physics::SimbodyPhysics::CreateShape (const std::string & _shapeType, CollisionPtr _collision)` [virtual]

Create a **physics::Shape** (p. 932) object.

Parameters

in	<code>_shapeType</code>	Type of shape to create.
in	<code>_collision</code>	Collision (p. 235) parent.

Implements **gazebo::physics::PhysicsEngine** (p. 771).

10.214.3.6 virtual `void gazebo::physics::SimbodyPhysics::DebugPrint () const` [virtual]

Debug print out of the physic engine state.

Implements **gazebo::physics::PhysicsEngine** (p. 772).

10.214.3.7 virtual `void gazebo::physics::SimbodyPhysics::Fini ()` [virtual]

Finalize the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 772).

10.214.3.8 `SimTK::MultibodySystem* gazebo::physics::SimbodyPhysics::GetDynamicsWorld () const`

Register a joint with the dynamics world.

10.214.3.9 `virtual boost::any gazebo::physics::SimbodyPhysics::GetParam (const std::string & _key) const [virtual]`

Get an parameter of the physics engine.

Parameters

<code>in</code>	<code>_attr</code>	String key
-----------------	--------------------	------------

See Also

SetParam (p. 994)

Returns

The value of the parameter

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 773).

10.214.3.10 `static SimTK::Transform gazebo::physics::SimbodyPhysics::GetPose (sdf::ElementPtr _element) [static]`

If the given element contains a <pose> element, return it as a Transform.

Otherwise return the identity Transform. If there is more than one <pose> element, only the first one is processed.

10.214.3.11 `virtual std::string gazebo::physics::SimbodyPhysics::GetType () const [virtual]`

Return the physics engine type (ode|bullet|dart|simbody).

Returns

Type of the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 774).

10.214.3.12 `static std::string gazebo::physics::SimbodyPhysics::GetTypeString (unsigned int _type) [static]`

Convert **Base::GetType()** (p. 176) to string, this is needed by the MultibodyGraphMaker.

Parameters

<code>in</code>	<code>_type</code>	Joint (p. 541) type returned by Joint::GetType() (p. 176).
-----------------	--------------------	--

Returns

a hard-coded string needed by the MultibodyGraphMaker.

10.214.3.13 `static std::string gazebo::physics::SimbodyPhysics::GetTypeString (physics::Base::EntityType _type)`
`[static]`

Convert **Base::GetType()** (p. 176) to string, this is needed by the MultibodyGraphMaker.

Parameters

<code>in</code>	<code>_type</code>	Joint (p. 541) type returned by Joint::GetType() (p. 176).
-----------------	--------------------	--

Returns

a hard-coded string needed by the MultibodyGraphMaker.

10.214.3.14 `virtual void gazebo::physics::SimbodyPhysics::Init ()` `[virtual]`

Initialize the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 775).

10.214.3.15 `virtual void gazebo::physics::SimbodyPhysics::InitForThread ()` `[virtual]`

Init the engine for threads.

Implements **gazebo::physics::PhysicsEngine** (p. 775).

10.214.3.16 `void gazebo::physics::SimbodyPhysics::InitModel (const physics::ModelPtr _model)`

Add a **Model** (p. 678) to the Simbody system.

Parameters

<code>in</code>	<code>_model</code>	Pointer to the model to add into Simbody.
-----------------	---------------------	---

10.214.3.17 `virtual void gazebo::physics::SimbodyPhysics::Load (sdf::ElementPtr _sdf)` `[virtual]`

Load the physics engine.

Parameters

<code>in</code>	<code>_sdf</code>	Pointer to the SDF parameters.
-----------------	-------------------	--------------------------------

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 775).

10.214.3.18 `virtual void gazebo::physics::SimbodyPhysics::OnPhysicsMsg (ConstPhysicsPtr & _msg)` `[protected]`,
`[virtual]`

virtual callback for gztopic "~/physics".

Parameters

in	<code>_msg</code>	Physics message.
----	-------------------	------------------

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 776).

10.214.3.19 `virtual void gazebo::physics::SimbodyPhysics::OnRequest (ConstRequestPtr & _msg) [protected], [virtual]`

virtual callback for gztopic "~/request".

Parameters

in	<code>_msg</code>	Request message.
----	-------------------	------------------

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 776).

10.214.3.20 `static SimTK::Transform gazebo::physics::SimbodyPhysics::Pose2Transform (const math::Pose & _pose) [static]`

Convert the given pose in x,y,z,thetax,thetay,thetaz format to a Simbody Transform.

The rotation angles are interpreted as a body-fixed sequence, meaning we rotation about x, then about the new y, then about the now twice-rotated z.

Parameters

in	<code>_pose</code>	Gazebo's math::Pose (p. 797) object
----	--------------------	--

Returns

Simbody's SimTK::Transform object

10.214.3.21 `static SimTK::Quaternion gazebo::physics::SimbodyPhysics::QuadToQuad (const math::Quaternion & _q) [static]`

Convert **gazebo::math::Quaternion** (p. 824) to SimTK::Quaternion.

Parameters

in	<code>_q</code>	Gazebo's math::Quaternion (p. 824) object
----	-----------------	--

Returns

Simbody's SimTK::Quaternion object

10.214.3.22 `static math::Quaternion gazebo::physics::SimbodyPhysics::QuadToQuad (const SimTK::Quaternion & _q) [static]`

Convert SimTK::Quaternion to **gazebo::math::Quaternion** (p. 824).

Parameters

in	<code>_q</code>	Simbody's SimTK::Quaternion object
----	-----------------	------------------------------------

Returns

Gazebo's **math::Quaternion** (p. 824) object

10.214.3.23 `virtual void gazebo::physics::SimbodyPhysics::Reset () [virtual]`

Rest the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 776).

10.214.3.24 `virtual void gazebo::physics::SimbodyPhysics::SetGravity (const gazebo::math::Vector3 & _gravity) [virtual]`

Set the gravity vector.

Parameters

in	<code>_gravity</code>	New gravity vector.
----	-----------------------	---------------------

Implements **gazebo::physics::PhysicsEngine** (p. 777).

10.214.3.25 `virtual bool gazebo::physics::SimbodyPhysics::SetParam (const std::string & _key, const boost::any & _value) [virtual]`

Set a parameter of the physics engine.

See SetParam documentation for descriptions of duplicate parameters.

Parameters

in	_key	String key Below is a list of _key parameter definitions: <ol style="list-style-type: none"> 1. "solver_type" (string) - returns solver used by engine, e.g. "sequential_impulse" for Bullet, "quick" for ODE "Featherstone and Lemkes" for DART and "Spatial Algebra and Elastic Foundation" for Simbody. -# "type" (string) - deprecated, use keyword "solver_type". -# "cfm" (double) - global CFM -# "erp" (double) - global ERP -# "precon_iters" (bool) - precondition iterations (experimental). 2. "iters" (int) - number of LCP PGS iterations. If sor_lcp_tolerance is negative, full iteration count is executed. Otherwise, PGS may stop iteration early if sor_lcp_tolerance is satisfied by the total RMS residual. 3. "sor" (double) - relaxation parameter for Projected Gauss-Seidel (PGS) updates. 4. "contact_max_correcting_vel" (double) - truncates correction impulses from ERP by this value. 5. "contact_surface_layer" (double) - ERP is 0 for interpenetration depths below this value. 6. "max_contacts" (int) - max number of contact constraints between any pair of collision bodies. 7. "min_step_size" (double) - minimum internal step size. (defined but not used in ode). 8. "max_step_size" (double) - maximum physics step size when physics update step must return.
in	_value	The value to set to

Returns

true if SetParam is successful, false if operation fails.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 777).

10.214.3.26 virtual void gazebo::physics::SimbodyPhysics::SetSeed (uint32_t _seed) [virtual]

Set the random number seed for the physics engine.

Parameters

in	_seed	The random number seed.
----	-------	-------------------------

Implements **gazebo::physics::PhysicsEngine** (p. 778).

10.214.3.27 static math::Pose gazebo::physics::SimbodyPhysics::Transform2Pose (const SimTK::Transform & _xAB)
[static]

Convert a Simbody transform to a pose in x,y,z, thetax,thetay,thetaz format.

Parameters

in	_xAB	Simbody's SimTK::Transform object
----	------	-----------------------------------

Returns

Gazebo's **math::Pose** (p. 797) object

10.214.3.28 `virtual void gazebo::physics::SimbodyPhysics::UpdateCollision () [virtual]`

Update the physics engine collision.

Implements **gazebo::physics::PhysicsEngine** (p. 780).

10.214.3.29 `virtual void gazebo::physics::SimbodyPhysics::UpdatePhysics () [virtual]`

Update the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 780).

10.214.3.30 `static math::Vector3 gazebo::physics::SimbodyPhysics::Vec3ToVector3 (const SimTK::Vec3 & _v) [static]`

Convert SimTK::Vec3 to **gazebo::math::Vector3** (p. 1165).

Parameters

in	_v	Simbody's SimTK::Vec3 object
----	----	------------------------------

Returns

Gazebo's **math::Vector3** (p. 1165) object

10.214.3.31 `static SimTK::Vec3 gazebo::physics::SimbodyPhysics::Vector3ToVec3 (const math::Vector3 & _v) [static]`

Convert **gazebo::math::Vector3** (p. 1165) to SimTK::Vec3.

Parameters

in	_v	Gazebo's math::Vector3 (p. 1165) object
----	----	--

Returns

Simbody's SimTK::Vec3 object

10.214.4 Member Data Documentation

10.214.4.1 `SimTK::CompliantContactSubsystem gazebo::physics::SimbodyPhysics::contact`

10.214.4.2 `SimTK::Force::DiscreteForces gazebo::physics::SimbodyPhysics::discreteForces`

10.214.4.3 SimTK::GeneralForceSubsystem gazebo::physics::SimbodyPhysics::forces

10.214.4.4 SimTK::Force::Gravity gazebo::physics::SimbodyPhysics::gravity

10.214.4.5 SimTK::Integrator* gazebo::physics::SimbodyPhysics::integ

10.214.4.6 SimTK::SimbodyMatterSubsystem gazebo::physics::SimbodyPhysics::matter

10.214.4.7 bool gazebo::physics::SimbodyPhysics::simbodyPhysicsInitialized

true if initialized

10.214.4.8 bool gazebo::physics::SimbodyPhysics::simbodyPhysicsStepped

10.214.4.9 SimTK::MultibodySystem gazebo::physics::SimbodyPhysics::system

10.214.4.10 SimTK::ContactTrackerSubsystem gazebo::physics::SimbodyPhysics::tracker

The documentation for this class was generated from the following file:

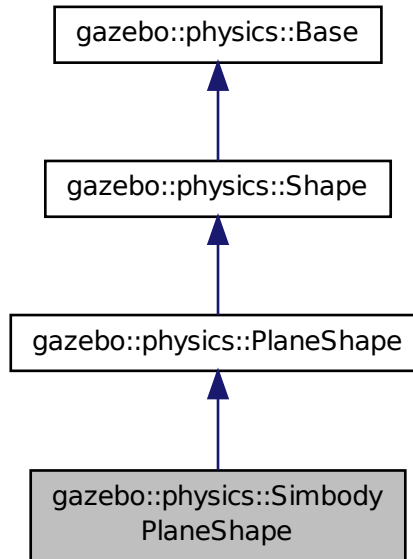
- **SimbodyPhysics.hh**

10.215 gazebo::physics::SimbodyPlaneShape Class Reference

Simbody collision for an infinite plane.

```
#include <SimbodyPlaneShape.hh>
```

Inheritance diagram for gazebo::physics::SimbodyPlaneShape:



Public Member Functions

- **SimbodyPlaneShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~SimbodyPlaneShape** ()
Destructor.
- virtual void **CreatePlane** ()
Create the plane.
- virtual void **SetAltitude** (const **math::Vector3** &_pos)
Set the altitude of the plane.

Additional Inherited Members

10.215.1 Detailed Description

Simbody collision for an infinite plane.

10.215.2 Constructor & Destructor Documentation

10.215.2.1 gazebo::physics::SimbodyPlaneShape::SimbodyPlaneShape (**CollisionPtr** _parent)

Constructor.

10.215.2.2 virtual gazebo::physics::SimbodyPlaneShape::~~SimbodyPlaneShape () [virtual]

Destructor.

10.215.3 Member Function Documentation

10.215.3.1 virtual void gazebo::physics::SimbodyPlaneShape::CreatePlane () [virtual]

Create the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 792).

10.215.3.2 virtual void gazebo::physics::SimbodyPlaneShape::SetAltitude (const math::Vector3 & _pos) [virtual]

Set the altitude of the plane.

Parameters

<code>in</code>	<code>_pos</code>	Position of the plane.
-----------------	-------------------	------------------------

Reimplemented from **gazebo::physics::PlaneShape** (p. 793).

The documentation for this class was generated from the following file:

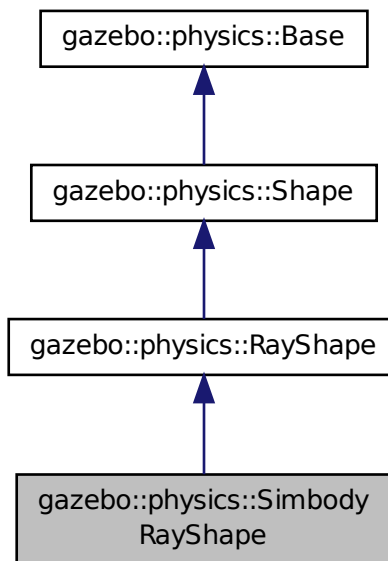
- **SimbodyPlaneShape.hh**

10.216 gazebo::physics::SimbodyRayShape Class Reference

Ray shape for simbody.

```
#include <SimbodyRayShape.hh>
```

Inheritance diagram for gazebo::physics::SimbodyRayShape:



Public Member Functions

- **SimbodyRayShape** (**PhysicsEnginePtr** _physicsEngine)
Constructor.
- **SimbodyRayShape** (**CollisionPtr** _collision)
Constructor.
- virtual **~SimbodyRayShape** ()
Destructor.
- virtual void **GetIntersection** (double &_dist, std::string &_entity)
Get the nearest intersection.
- virtual void **SetPoints** (const **math::Vector3** &_posStart, const **math::Vector3** &_posEnd)
Set the ray based on starting and ending points relative to the body.
- virtual void **Update** ()
Update the ray collision.

Additional Inherited Members

10.216.1 Detailed Description

Ray shape for simbody.

10.216.2 Constructor & Destructor Documentation

10.216.2.1 gazebo::physics::SimbodyRayShape::SimbodyRayShape (PhysicsEnginePtr *_physicsEngine*)

Constructor.

Parameters

in	<i>_physicsEngine</i>	Pointer to the physics engine.
----	-----------------------	--------------------------------

10.216.2.2 gazebo::physics::SimbodyRayShape::SimbodyRayShape (CollisionPtr *_collision*)

Constructor.

Parameters

in	<i>_collision</i>	Collision (p. 235) the ray is attached to.
----	-------------------	---

10.216.2.3 virtual gazebo::physics::SimbodyRayShape::~~SimbodyRayShape () [virtual]

Destructor.

10.216.3 Member Function Documentation

10.216.3.1 virtual void gazebo::physics::SimbodyRayShape::GetIntersection (double & *_dist*, std::string & *_entity*) [virtual]

Get the nearest intersection.

Parameters

out	<i>_dist</i>	Distance to the intersection.
out	<i>_entity</i>	Name of the entity the ray intersected with.

Implements **gazebo::physics::RayShape** (p. 852).

10.216.3.2 virtual void gazebo::physics::SimbodyRayShape::SetPoints (const math::Vector3 & *_posStart*, const math::Vector3 & *_posEnd*) [virtual]

Set the ray based on starting and ending points relative to the body.

Parameters

in	<i>_posStart</i>	Start position, relative the body.
in	<i>_posEnd</i>	End position, relative to the body.

Reimplemented from **gazebo::physics::RayShape** (p. 853).

10.216.3.3 `virtual void gazebo::physics::SimbodyRayShape::Update () [virtual]`

Update the ray collision.

Implements `gazebo::physics::RayShape` (p. 854).

The documentation for this class was generated from the following file:

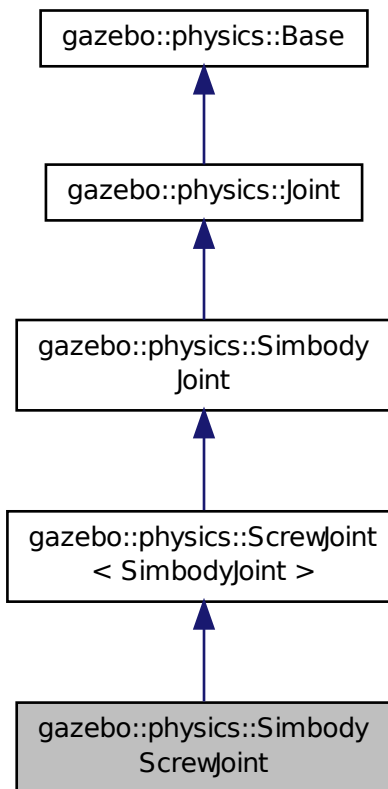
- `SimbodyRayShape.hh`

10.217 `gazebo::physics::SimbodyScrewJoint` Class Reference

A screw joint.

```
#include <SimbodyScrewJoint.hh>
```

Inheritance diagram for `gazebo::physics::SimbodyScrewJoint`:



Public Member Functions

- `SimbodyScrewJoint` (`SimTK::MultibodySystem *_world`, `BasePtr _parent`)

Constructor.

- virtual `~SimbodyScrewJoint ()`

Destructor.

- virtual `math::Angle GetAngleImpl (unsigned int _index) const`
Get the angle of an axis helper function.
- virtual `double GetAttribute (const std::string &_key, unsigned int _index) GAZEBO_DEPRECATED(3.0)`
Get a non-generic parameter for the joint.
- virtual `math::Vector3 GetGlobalAxis (unsigned int _index) const`
Get the axis of rotation in global coordinate frame.
- virtual `math::Angle GetHighStop (unsigned int _index)`
Get the high stop of an axis(index).
- virtual `math::Angle GetLowStop (unsigned int _index)`
Get the low stop of an axis(index).
- virtual `double GetMaxForce (unsigned int _index)`
Get the max allowed force of an axis(index).
- virtual `double GetParam (const std::string &_key, unsigned int _index)`
Get a non-generic parameter for the joint.
- virtual `double GetThreadPitch (unsigned int)`
Get screw joint thread pitch.
- virtual `double GetThreadPitch ()`
Get screw joint thread pitch.
- virtual `double GetVelocity (unsigned int _index) const`
Get the rotation rate of an axis(index)
- virtual `void SetAttribute (const std::string &_key, unsigned int _index, const boost::any &_value) GAZEBO_DEPRECATED(3.0)`
Set a non-generic parameter for the joint.
- virtual `void SetAxis (unsigned int _index, const math::Vector3 &_axis)`
Set the axis of rotation where axis is specified in local joint frame.
- virtual `bool SetHighStop (unsigned int _index, const math::Angle &_angle)`
Set the high stop of an axis(index).
- virtual `bool SetLowStop (unsigned int _index, const math::Angle &_angle)`
Set the low stop of an axis(index).
- virtual `void SetMaxForce (unsigned int _index, double _t)`
Set the max allowed force of an axis(index).
- virtual `bool SetParam (const std::string &_key, unsigned int _index, const boost::any &_value)`
Set a non-generic parameter for the joint.
- virtual `void SetThreadPitch (unsigned int _index, double _threadPitch)`
Set screw joint thread pitch.
- virtual `void SetThreadPitch (double _threadPitch)`
Set screw joint thread pitch.
- virtual `void SetVelocity (unsigned int _index, double _angle)`
Set the velocity of an axis(index).

Protected Member Functions

- virtual `void Load (sdf::ElementPtr _sdf)`
*Load a **ScrewJoint** (p. 896).*
- virtual `void SetForceImpl (unsigned int _index, double _force)`
*Set the force applied to this **physics::Joint** (p. 541).*

Additional Inherited Members

10.217.1 Detailed Description

A screw joint.

10.217.2 Constructor & Destructor Documentation

10.217.2.1 gazebo::physics::SimbodyScrewJoint::SimbodyScrewJoint (SimTK::MultibodySystem * *_world*, BasePtr *_parent*)

Constructor.

Parameters

in	<i>_world</i>	Pointer to the Simbody world.
in	<i>_parent</i>	Parent of the screw joint.

10.217.2.2 virtual gazebo::physics::SimbodyScrewJoint::~~SimbodyScrewJoint () [virtual]

Destructor.

10.217.3 Member Function Documentation

10.217.3.1 virtual math::Angle gazebo::physics::SimbodyScrewJoint::GetAngleImpl (unsigned int *_index*) const [virtual]

Get the angle of an axis helper function.

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 550).

10.217.3.2 virtual double gazebo::physics::SimbodyScrewJoint::GetAttribute (const std::string & *_key*, unsigned int *_index*) [virtual]

Get a non-generic parameter for the joint.

Deprecated by GetParam

Parameters

in	<i>_key</i>	String key.
in	<i>_index</i>	Index of the axis.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 964).

10.217.3.3 `virtual math::Vector3 gazebo::physics::SimbodyScrewJoint::GetGlobalAxis (unsigned int _index) const`
`[virtual]`

Get the axis of rotation in global coordinate frame.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis to get.
-----------------	----------------------------	---------------------------

Returns

Axis value for the provided index.

Implements `gazebo::physics::Joint` (p. 553).

10.217.3.4 `virtual math::Angle gazebo::physics::SimbodyScrewJoint::GetHighStop (unsigned int _index)` `[virtual]`

Get the high stop of an axis(index).

This function is replaced by `GetUpperLimit(unsigned int)`. If you are interested in getting the value of `dParamHiStop*`, use `GetAttribute(hi_stop, _index)`

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Angle of the high stop value.

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 965).

10.217.3.5 `virtual math::Angle gazebo::physics::SimbodyScrewJoint::GetLowStop (unsigned int _index)` `[virtual]`

Get the low stop of an axis(index).

This function is replaced by `GetLowerLimit(unsigned int)`. If you are interested in getting the value of `dParamHiStop*`, use `GetAttribute(hi_stop, _index)`

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Angle of the low stop value.

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 966).

10.217.3.6 `virtual double gazebo::physics::SimbodyScrewJoint::GetMaxForce (unsigned int _index)` `[virtual]`

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	<code>_index</code>	Index of the axis.
----	---------------------	--------------------

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 556).

10.217.3.7 `virtual double gazebo::physics::SimbodyScrewJoint::GetParam (const std::string & _key, unsigned int _index)` [virtual]

Get a non-generic parameter for the joint.

Parameters

in	<code>_key</code>	String key.
in	<code>_index</code>	Index of the axis.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 966).

10.217.3.8 `virtual double gazebo::physics::SimbodyScrewJoint::GetThreadPitch (unsigned int _index)` [virtual]

Get screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class

Parameters

in	<code>_index</code>	Index of the axis.
----	---------------------	--------------------

Returns

`_threadPitch` Thread pitch value.

Implements **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 898).

10.217.3.9 `virtual double gazebo::physics::SimbodyScrewJoint::GetThreadPitch ()` [virtual]

Get screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class

Returns

`_threadPitch` Thread pitch value.

Implements **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 898).

10.217.3.10 virtual double gazebo::physics::SimbodyScrewJoint::GetVelocity (unsigned int *_index*) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 558).

10.217.3.11 virtual void gazebo::physics::SimbodyScrewJoint::Load (sdf::ElementPtr *_sdf*) [protected],[virtual]

Load a **ScrewJoint** (p. 896).

Parameters

in	<i>_sdf</i>	SDF value to load from
----	-------------	------------------------

Reimplemented from **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 898).

10.217.3.12 virtual void gazebo::physics::SimbodyScrewJoint::SetAttribute (const std::string & *_key*, unsigned int *_index*, const boost::any & *_value*) [virtual]

Set a non-generic parameter for the joint.

replaces SetAttribute(Attribute, int, double) Deprecated by bool SetParam

Parameters

in	<i>_key</i>	String key.
in	<i>_index</i>	Index of the axis.
in	<i>_value</i>	Value of the attribute.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 967).

10.217.3.13 virtual void gazebo::physics::SimbodyScrewJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	<i>_index</i>	Index of the axis to set.
in	<i>_axis</i>	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 968).

10.217.3.14 `virtual void gazebo::physics::SimbodyScrewJoint::SetForceImpl (unsigned int _index, double _force)`
`[protected], [virtual]`

Set the force applied to this **physics::Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 541)).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_force</code>	Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only.

Implements **gazebo::physics::SimbodyJoint** (p. 968).

10.217.3.15 `virtual bool gazebo::physics::SimbodyScrewJoint::SetHighStop (unsigned int _index, const math::Angle & _angle)`
`[virtual]`

Set the high stop of an axis(index).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_angle</code>	High stop angle.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 969).

10.217.3.16 `virtual bool gazebo::physics::SimbodyScrewJoint::SetLowStop (unsigned int _index, const math::Angle & _angle)`
`[virtual]`

Set the low stop of an axis(index).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_angle</code>	Low stop angle.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 969).

10.217.3.17 `virtual void gazebo::physics::SimbodyScrewJoint::SetMaxForce (unsigned int _index, double _force)` `[virtual]`

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_force</code>	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 563).

10.217.3.18 `virtual bool gazebo::physics::SimbodyScrewJoint::SetParam (const std::string & _key, unsigned int _index, const boost::any & _value) [virtual]`

Set a non-generic parameter for the joint.

replaces SetAttribute(Attribute, int, double)

Parameters

in	<i>_key</i>	String key.
in	<i>_index</i>	Index of the axis.
in	<i>_value</i>	Value of the attribute.

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 969).

10.217.3.19 `virtual void gazebo::physics::SimbodyScrewJoint::SetThreadPitch (unsigned int _index, double _threadPitch) [virtual]`

Set screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class `Deprecated`, please use the index-less version in the future: `virtual void SetThreadPitch(double _threadPitch)` (p. 1009) = 0;

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_threadPitch</i>	Thread pitch value.

Implements `gazebo::physics::ScrewJoint< SimbodyJoint >` (p. 899).

10.217.3.20 `virtual void gazebo::physics::SimbodyScrewJoint::SetThreadPitch (double _threadPitch) [virtual]`

Set screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class To clarify direction, these are modeling right handed threads with positive `thread_pitch`, i.e. the child `Link` (p. 595) is the nut (interior threads) while the parent `Link` (p. 595) is the bolt/screw (exterior threads).

Parameters

in	<i>_threadPitch</i>	Thread pitch value.
----	---------------------	---------------------

Implements `gazebo::physics::ScrewJoint< SimbodyJoint >` (p. 899).

10.217.3.21 `virtual void gazebo::physics::SimbodyScrewJoint::SetVelocity (unsigned int _index, double _vel) [virtual]`

Set the velocity of an axis(index).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_vel</i>	Velocity.

Implements **gazebo::physics::Joint** (p. 565).

The documentation for this class was generated from the following file:

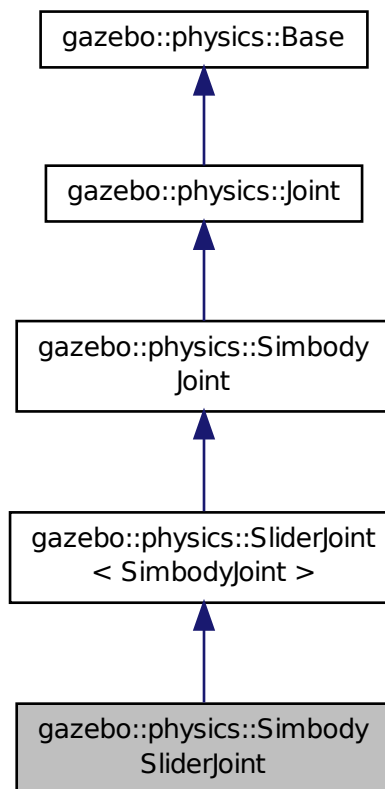
- **SimbodyScrewJoint.hh**

10.218 gazebo::physics::SimbodySliderJoint Class Reference

A slider joint.

```
#include <SimbodySliderJoint.hh>
```

Inheritance diagram for gazebo::physics::SimbodySliderJoint:



Public Member Functions

- **SimbodySliderJoint** (SimTK::MultibodySystem *world, BasePtr _parent)
Constructor.
- virtual ~**SimbodySliderJoint** ()

Destructor.

- virtual **math::Angle GetAngleImpl** (unsigned int *_index*) const
Get the angle of an axis helper function.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int *_index*) const
Get the axis of rotation in global coordinate frame.
- virtual double **GetMaxForce** (unsigned int *_index*)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (unsigned int *_index*) const
Get the rotation rate of an axis(index)
- virtual void **SetAxis** (unsigned int *_index*, const **math::Vector3** &*_axis*)
Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int *_index*, double *_t*)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (unsigned int *_index*, double *_rate*)
Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr *_sdf*)
*Load a **SliderJoint** (p. 1044).*
- virtual void **SetForceImpl** (unsigned int *_index*, double *_force*)
*Set the force applied to this **physics::Joint** (p. 541).*

Additional Inherited Members

10.218.1 Detailed Description

A slider joint.

10.218.2 Constructor & Destructor Documentation

10.218.2.1 gazebo::physics::SimbodySliderJoint::SimbodySliderJoint (**SimTK::MultibodySystem** * *world*, **BasePtr** *_parent*)

Constructor.

Parameters

<i>in</i>	<i>_world</i>	Pointer to the Simbody world.
<i>in</i>	<i>_parent</i>	Parent of the screw joint.

10.218.2.2 virtual gazebo::physics::SimbodySliderJoint::~~SimbodySliderJoint () [virtual]

Destructor.

10.218.3 Member Function Documentation

10.218.3.1 `virtual math::Angle gazebo::physics::SimbodySliderJoint::GetAngleImpl (unsigned int _index) const`
`[virtual]`

Get the angle of an axis helper function.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Angle of the axis.

Implements `gazebo::physics::Joint` (p. 550).

10.218.3.2 `virtual math::Vector3 gazebo::physics::SimbodySliderJoint::GetGlobalAxis (unsigned int _index) const`
`[virtual]`

Get the axis of rotation in global coordinate frame.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis to get.
-----------------	----------------------------	---------------------------

Returns

Axis value for the provided index.

Implements `gazebo::physics::Joint` (p. 553).

10.218.3.3 `virtual double gazebo::physics::SimbodySliderJoint::GetMaxForce (unsigned int _index)` `[virtual]`

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

The maximum force.

Implements `gazebo::physics::Joint` (p. 556).

10.218.3.4 `virtual double gazebo::physics::SimbodySliderJoint::GetVelocity (unsigned int _index) const` `[virtual]`

Get the rotation rate of an axis(index)

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

The rotational velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 558).

10.218.3.5 virtual void gazebo::physics::SimbodySliderJoint::Load (sdf::ElementPtr *_sdf*) [protected],[virtual]

Load a **SliderJoint** (p. 1044).

Parameters

in	<i>_sdf</i>	SDF values to load from
----	-------------	-------------------------

Reimplemented from **gazebo::physics::SliderJoint**< **SimbodyJoint** > (p. 1045).

10.218.3.6 virtual void gazebo::physics::SimbodySliderJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	<i>_index</i>	Index of the axis to set.
in	<i>_axis</i>	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 968).

10.218.3.7 virtual void gazebo::physics::SimbodySliderJoint::SetForceImpl (unsigned int *_index*, double *_force*) [protected],[virtual]

Set the force applied to this **physics::Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 541)).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_force</i>	Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only.

Implements **gazebo::physics::SimbodyJoint** (p. 968).

10.218.3.8 virtual void gazebo::physics::SimbodySliderJoint::SetMaxForce (unsigned int *_index*, double *_force*) [virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_force</i>	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 563).

10.218.3.9 virtual void gazebo::physics::SimbodySliderJoint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_vel</i>	Velocity.

Implements **gazebo::physics::Joint** (p. 565).

The documentation for this class was generated from the following file:

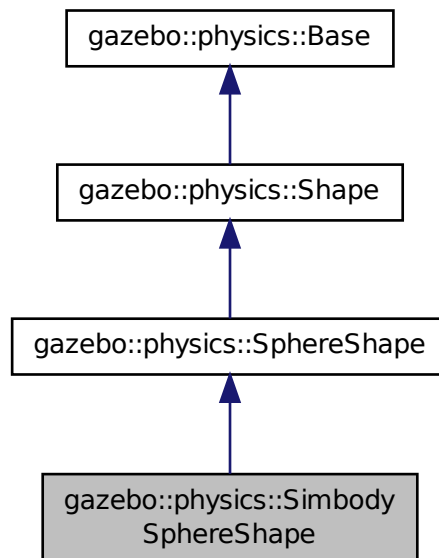
- **SimbodySliderJoint.hh**

10.219 gazebo::physics::SimbodySphereShape Class Reference

Simbody sphere collision.

```
#include <SimbodySphereShape.hh>
```

Inheritance diagram for gazebo::physics::SimbodySphereShape:



Public Member Functions

- **SimbodySphereShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~SimbodySphereShape** ()
Destructor.
- virtual void **SetRadius** (double _radius)
Set the size.

Additional Inherited Members

10.219.1 Detailed Description

Simbody sphere collision.

10.219.2 Constructor & Destructor Documentation

10.219.2.1 gazebo::physics::SimbodySphereShape::SimbodySphereShape (**CollisionPtr** _parent) [inline]

Constructor.

Parameters

in	_parent	Collision (p. 235) parent pointer
----	---------	--

10.219.2.2 virtual gazebo::physics::SimbodySphereShape::~~SimbodySphereShape () [inline],[virtual]

Destructor.

10.219.3 Member Function Documentation

10.219.3.1 virtual void gazebo::physics::SimbodySphereShape::SetRadius (double _radius) [inline],[virtual]

Set the size.

Parameters

in	_radius	Radius of the sphere.
----	---------	-----------------------

Reimplemented from **gazebo::physics::SphereShape** (p. 1057).

References gazebo::math::equal(), gzerr, gzwarn, and gazebo::physics::SphereShape::SetRadius().

The documentation for this class was generated from the following file:

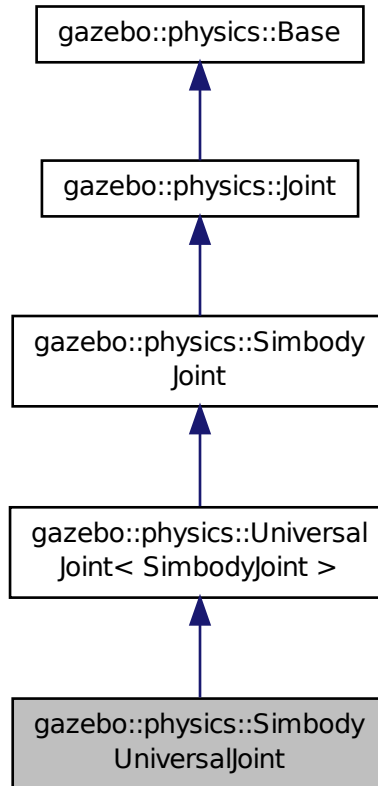
- **SimbodySphereShape.hh**

10.220 gazebo::physics::SimbodyUniversalJoint Class Reference

A simbody universal joint class.

```
#include <SimbodyUniversalJoint.hh>
```

Inheritance diagram for gazebo::physics::SimbodyUniversalJoint:



Public Member Functions

- **SimbodyUniversalJoint** (SimTK::MultibodySystem * _world, **BasePtr** _parent)
Constructor.
- virtual **~SimbodyUniversalJoint** ()
Destructor.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const
Get the anchor point.
- virtual **math::Vector3 GetAxis** (unsigned int _index) const
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const
Get the axis of rotation in global coordinate frame.

- virtual double **GetMaxForce** (unsigned int *_index*)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (unsigned int *_index*) const
Get the rotation rate of an axis(index)
- virtual void **Load** (sdf::ElementPtr *_sdf*)
*Load a **UniversalJoint** (p. 1135).*
- virtual void **SetAxis** (unsigned int *_index*, const **math::Vector3** &*_axis*)
Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int *_index*, double *_t*)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (unsigned int *_index*, double *_rate*)
Set the velocity of an axis(index).

Protected Member Functions

- virtual **math::Angle** **GetAngleImpl** (unsigned int *_index*) const
Get the angle of an axis helper function.
- virtual void **SetForceImpl** (unsigned int *_index*, double *_torque*)
*Set the force applied to this **physics::Joint** (p. 541).*

Additional Inherited Members

10.220.1 Detailed Description

A simbody universal joint class.

10.220.2 Constructor & Destructor Documentation

10.220.2.1 gazebo::physics::SimbodyUniversalJoint::SimbodyUniversalJoint (**SimTK::MultibodySystem** * *_world*, **BasePtr** *_parent*)

Constructor.

Parameters

<i>in</i>	<i>_world</i>	Pointer to the Simbody world.
<i>in</i>	<i>_parent</i>	Parent of the screw joint.

10.220.2.2 virtual gazebo::physics::SimbodyUniversalJoint::~~SimbodyUniversalJoint () [virtual]

Destuctor.

10.220.3 Member Function Documentation

10.220.3.1 `virtual math::Vector3 gazebo::physics::SimbodyUniversalJoint::GetAnchor (unsigned int _index) const`
`[virtual]`

Get the anchor point.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Anchor value for the axis.

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 963).

10.220.3.2 `virtual math::Angle gazebo::physics::SimbodyUniversalJoint::GetAngleImpl (unsigned int _index) const`
`[protected], [virtual]`

Get the angle of an axis helper function.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Angle of the axis.

Implements `gazebo::physics::Joint` (p. 550).

10.220.3.3 `virtual math::Vector3 gazebo::physics::SimbodyUniversalJoint::GetAxis (unsigned int _index) const`
`[virtual]`

10.220.3.4 `virtual math::Vector3 gazebo::physics::SimbodyUniversalJoint::GetGlobalAxis (unsigned int _index) const`
`[virtual]`

Get the axis of rotation in global coordinate frame.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis to get.
-----------------	----------------------------	---------------------------

Returns

Axis value for the provided index.

Implements `gazebo::physics::Joint` (p. 553).

10.220.3.5 `virtual double gazebo::physics::SimbodyUniversalJoint::GetMaxForce (unsigned int _index)` `[virtual]`

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	<code>_index</code>	Index of the axis.
----	---------------------	--------------------

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 556).

10.220.3.6 virtual double gazebo::physics::SimbodyUniversalJoint::GetVelocity (unsigned int `_index`) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	<code>_index</code>	Index of the axis.
----	---------------------	--------------------

Returns

The rotational velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 558).

10.220.3.7 virtual void gazebo::physics::SimbodyUniversalJoint::Load (sdf::ElementPtr `_sdf`) [virtual]

Load a **UniversalJoint** (p. 1135).

Parameters

in	<code>_sdf</code>	SDF values to load from.
----	-------------------	--------------------------

Reimplemented from **gazebo::physics::UniversalJoint< SimbodyJoint >** (p. 1136).

10.220.3.8 virtual void gazebo::physics::SimbodyUniversalJoint::SetAxis (unsigned int `_index`, const math::Vector3 & `_axis`) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	<code>_index</code>	Index of the axis to set.
in	<code>_axis</code>	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 968).

10.220.3.9 virtual void gazebo::physics::SimbodyUniversalJoint::SetForceImpl (unsigned int `_index`, double `_force`) [protected], [virtual]

Set the force applied to this **physics::Joint** (p. 541).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to

SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 541)).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_force</code>	Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only.

Implements **gazebo::physics::SimbodyJoint** (p. 968).

10.220.3.10 `virtual void gazebo::physics::SimbodyUniversalJoint::SetMaxForce (unsigned int _index, double _force)`
`[virtual]`

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_force</code>	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 563).

10.220.3.11 `virtual void gazebo::physics::SimbodyUniversalJoint::SetVelocity (unsigned int _index, double _vel)` `[virtual]`

Set the velocity of an axis(index).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_vel</code>	Velocity.

Implements **gazebo::physics::Joint** (p. 565).

The documentation for this class was generated from the following file:

- **SimbodyUniversalJoint.hh**

10.221 gazebo::sensors::SimTimeEvent Class Reference

```
#include <SensorManager.hh>
```

Public Attributes

- `boost::condition_variable * condition`
The condition to notify.
- `common::Time time`
The time at which to trigger the condition.

10.221.1 Detailed Description

A simulation time event

10.221.2 Member Data Documentation

10.221.2.1 boost::condition_variable* gazebo::sensors::SimTimeEvent::condition

The condition to notify.

10.221.2.2 common::Time gazebo::sensors::SimTimeEvent::time

The time at which to trigger the condition.

The documentation for this class was generated from the following file:

- **SensorManager.hh**

10.222 gazebo::sensors::SimTimeEventHandler Class Reference

Monitors simulation time, and notifies conditions when a specified time has been reached.

```
#include <SensorManager.hh>
```

Public Member Functions

- **SimTimeEventHandler** ()
Constructor.
- virtual **~SimTimeEventHandler** ()
Destructor.
- void **AddRelativeEvent** (const **common::Time** &_time, boost::condition_variable *_var)
Add a new event to the handler.

10.222.1 Detailed Description

Monitors simulation time, and notifies conditions when a specified time has been reached.

10.222.2 Constructor & Destructor Documentation

10.222.2.1 gazebo::sensors::SimTimeEventHandler::SimTimeEventHandler ()

Constructor.

10.222.2.2 virtual gazebo::sensors::SimTimeEventHandler::~~SimTimeEventHandler () [virtual]

Destructor.

10.222.3 Member Function Documentation

10.222.3.1 `void gazebo::sensors::SimTimeEventHandler::AddRelativeEvent (const common::Time & _time, boost::condition_variable * _var)`

Add a new event to the handler.

Parameters

<code>in</code>	<code><i>_time</i></code>	Time of the new event. The current sim time will be add to this time.
<code>in</code>	<code><i>_var</i></code>	Condition to notify when the time has been reached.

The documentation for this class was generated from the following file:

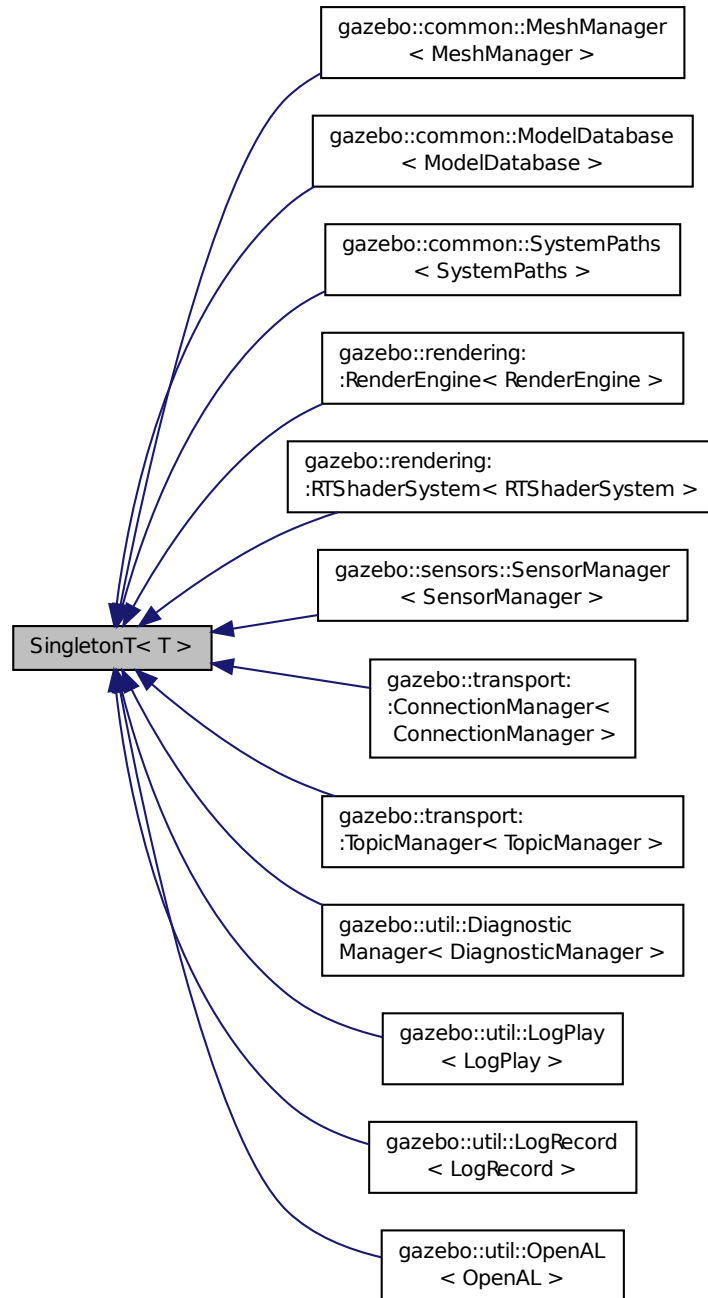
- **SensorManager.hh**

10.223 SingletonT< T > Class Template Reference

Singleton template class.

```
#include <common/common.hh>
```

Inheritance diagram for SingletonT< T >:



Static Public Member Functions

- static T * Instance ()

Get an instance of the singleton.

Protected Member Functions

- **SingletonT** ()
Constructor.
- virtual **~SingletonT** ()
Destructor.

10.223.1 Detailed Description

```
template<class T>class SingletonT< T >
```

Singleton template class.

10.223.2 Constructor & Destructor Documentation

10.223.2.1 `template<class T> SingletonT< T >::SingletonT ()` [inline],[protected]

Constructor.

10.223.2.2 `template<class T> virtual SingletonT< T >::~~SingletonT ()` [inline],[protected],[virtual]

Destructor.

10.223.3 Member Function Documentation

10.223.3.1 `template<class T> static T* SingletonT< T >::Instance ()` [inline],[static]

Get an instance of the singleton.

Referenced by gazebo::transport::TopicManager::Advertise(), gazebo::transport::Node::Advertise(), gazebo::PluginT< ModelPlugin >::Create(), and gazebo::transport::Node::Subscribe().

The documentation for this class was generated from the following file:

- **SingletonT.hh**

10.224 gazebo::common::Skeleton Class Reference

A skeleton.

```
#include <common/common.hh>
```

Public Member Functions

- **Skeleton** ()
Constructor.

- **Skeleton** (**SkeletonNode** *_root)
Constructor.
- virtual ~**Skeleton** ()
Destructor.
- void **AddAnimation** (**SkeletonAnimation** *_anim)
Add an animation.
- void **AddVertNodeWeight** (unsigned int _vertex, std::string _node, double _weight)
Add a new weight to a node (bone)
- **SkeletonAnimation** * **GetAnimation** (const unsigned int _i)
Find animation.
- **math::Matrix4** **GetBindShapeTransform** ()
Return bind pose skeletal transform.
- **SkeletonNode** * **GetNodeByHandle** (unsigned int _handle)
Find or create node with handle.
- **SkeletonNode** * **GetNodeById** (std::string _id)
Find node by index.
- **SkeletonNode** * **GetNodeByName** (std::string _name)
Find a node.
- **NodeMap** **GetNodes** ()
Get a copy or the node dictionary.
- unsigned int **GetNumAnimations** ()
Returns the number of animations.
- unsigned int **GetNumJoints** ()
Returns the number of joints.
- unsigned int **GetNumNodes** ()
Returns the node count.
- unsigned int **GetNumVertNodeWeights** (unsigned int _vertex)
Returns the number of bone weights for a vertex.
- **SkeletonNode** * **GetRootNode** ()
Return the root.
- std::pair< std::string, double > **GetVertNodeWeight** (unsigned int _v, unsigned int _i)
Weight of a bone for a vertex.
- void **PrintTransforms** ()
Outputs the transforms to std::err stream.
- void **Scale** (double _scale)
Scale all nodes, transforms and animation data.
- void **SetBindShapeTransform** (**math::Matrix4** _trans)
Set the bind pose skeletal transform.
- void **SetNumVertAttached** (unsigned int _vertices)
Resizes the raw node weight array.
- void **SetRootNode** (**SkeletonNode** *_node)
Change the root node.

Protected Member Functions

- void **BuildNodeMap** ()
Initializes the handle numbers for each node in the map using breadth first traversal.

Protected Attributes

- `std::vector< SkeletonAnimation * > anims`
the array of animations
- `math::Matrix4 bindShapeTransform`
the bind pose skeletal transform
- **NodeMap nodes**
The dictionary of nodes, indexed by name.
- **RawNodeWeights rawNW**
the node weight table
- **SkeletonNode * root**
the root node

10.224.1 Detailed Description

A skeleton.

10.224.2 Constructor & Destructor Documentation

10.224.2.1 gazebo::common::Skeleton::Skeleton ()

Constructor.

10.224.2.2 gazebo::common::Skeleton::Skeleton (**SkeletonNode** * *_root*)

Constructor.

Parameters

in	<i>_root</i>	node
----	--------------	------

10.224.2.3 virtual gazebo::common::Skeleton::~Skeleton () [virtual]

Destructor.

10.224.3 Member Function Documentation

10.224.3.1 void gazebo::common::Skeleton::AddAnimation (**SkeletonAnimation** * *_anim*)

Add an animation.

The skeleton does not take ownership of the animation

Parameters

in	<i>_anim</i>	the animation to add
----	--------------	----------------------

10.224.3.2 void gazebo::common::Skeleton::AddVertNodeWeight (unsigned int *_vertex*, std::string *_node*, double *_weight*)

Add a new weight to a node (bone)

Parameters

in	<i>_vertex</i>	index of the vertex
in	<i>_node</i>	name of the bone
in	<i>_weight</i>	the new weight (range 0 to 1)

10.224.3.3 void gazebo::common::Skeleton::BuildNodeMap () [protected]

Initializes the handle numbers for each node in the map using breadth first traversal.

10.224.3.4 SkeletonAnimation* gazebo::common::Skeleton::GetAnimation (const unsigned int *_i*)

Find animation.

Parameters

in	<i>_i</i>	the animation index
----	-----------	---------------------

Returns

the animation, or NULL if *_i* is out of bounds

10.224.3.5 math::Matrix4 gazebo::common::Skeleton::GetBindShapeTransform ()

Return bind pose skeletal transform.

Returns

a matrix

10.224.3.6 SkeletonNode* gazebo::common::Skeleton::GetNodeByHandle (unsigned int *_handle*)

Find or create node with handle.

Parameters

in	<i>_handle</i>	
----	----------------	--

Returns

the node. A new node is created if it didn't exist

10.224.3.7 SkeletonNode* gazebo::common::Skeleton::GetNodeById (std::string *_id*)

Find node by index.

Parameters

<code>in</code>	<code>_id</code>	the index
-----------------	------------------	-----------

Returns

the node, or NULL if not found

10.224.3.8 `SkeletonNode*` `gazebo::common::Skeleton::GetNodeByName (std::string _name)`

Find a node.

Parameters

<code>in</code>	<code>_name</code>	the name of the node to look for
-----------------	--------------------	----------------------------------

Returns

the node, or NULL if not found

10.224.3.9 `NodeMap` `gazebo::common::Skeleton::GetNodes ()`

Get a copy of the node dictionary.

10.224.3.10 `unsigned int` `gazebo::common::Skeleton::GetNumAnimations ()`

Returns the number of animations.

Returns

the count

10.224.3.11 `unsigned int` `gazebo::common::Skeleton::GetNumJoints ()`

Returns the number of joints.

Returns

the count

10.224.3.12 `unsigned int` `gazebo::common::Skeleton::GetNumNodes ()`

Returns the node count.

Returns

the count

10.224.3.13 `unsigned int gazebo::common::Skeleton::GetNumVertNodeWeights (unsigned int _vertex)`

Returns the number of bone weights for a vertex.

Parameters

<code>in</code>	<code><i>_vertex</i></code>	the index of the vertex
-----------------	-----------------------------	-------------------------

Returns

the count

10.224.3.14 `SkeletonNode* gazebo::common::Skeleton::GetRootNode ()`

Return the root.

Returns

the root

10.224.3.15 `std::pair<std::string, double> gazebo::common::Skeleton::GetVertNodeWeight (unsigned int _v, unsigned int _i)`

Weight of a bone for a vertex.

Parameters

<code>in</code>	<code><i>_v</i></code>	the index of the vertex
<code>in</code>	<code><i>_i</i></code>	the index of the weight for that vertex

Returns

a pair containing the name of the node and the weight

10.224.3.16 `void gazebo::common::Skeleton::PrintTransforms ()`

Outputs the transforms to `std::err` stream.

10.224.3.17 `void gazebo::common::Skeleton::Scale (double _scale)`

Scale all nodes, transforms and animation data.

Parameters

<code>in</code>	<code><i>the</i></code>	scaling factor
-----------------	-------------------------	----------------

10.224.3.18 `void gazebo::common::Skeleton::SetBindShapeTransform (math::Matrix4 _trans)`

Set the bind pose skeletal transform.

Parameters

in	<i>_trans</i>	the transform
----	---------------	---------------

10.224.3.19 void gazebo::common::Skeleton::SetNumVertAttached (unsigned int *_vertices*)

Resizes the raw node weight array.

Parameters

in	<i>_vertices</i>	the new size
----	------------------	--------------

10.224.3.20 void gazebo::common::Skeleton::SetRootNode (SkeletonNode * *_node*)

Change the root node.

Parameters

in	<i>_node</i>	the new node
----	--------------	--------------

10.224.4 Member Data Documentation

10.224.4.1 std::vector<SkeletonAnimation*> gazebo::common::Skeleton::anim [protected]

the array of animations

10.224.4.2 math::Matrix4 gazebo::common::Skeleton::bindShapeTransform [protected]

the bind pose skeletal transform

10.224.4.3 NodeMap gazebo::common::Skeleton::nodes [protected]

The dictionary of nodes, indexed by name.

10.224.4.4 RawNodeWeights gazebo::common::Skeleton::rawNW [protected]

the node weight table

10.224.4.5 SkeletonNode* gazebo::common::Skeleton::root [protected]

the root node

The documentation for this class was generated from the following file:

- **Skeleton.hh**

10.225 gazebo::common::SkeletonAnimation Class Reference

Skeleton (p. 1024) animation.

```
#include <SkeletonAnimation.hh>
```

Public Member Functions

- **SkeletonAnimation** (const std::string &_name)
The Constructor.
- **~SkeletonAnimation** ()
The destructor.
- void **AddKeyFrame** (const std::string &_node, const double _time, const **math::Matrix4** &_mat)
Adds or replaces a named key frame at a specific time.
- void **AddKeyFrame** (const std::string &_node, const double _time, const **math::Pose** &_pose)
Adds or replaces a named key frame at a specific time.
- double **GetLength** () const
Returns the duration of the animations.
- std::string **GetName** () const
Returns the name.
- unsigned int **GetNodeCount** () const
Returns the number of animation nodes.
- **math::Matrix4** **GetNodePoseAt** (const std::string &_node, const double _time, const bool _loop=true)
Returns the key frame transformation for a named animation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.
- std::map< std::string, **math::Matrix4** > **GetPoseAt** (const double _time, const bool _loop=true) const
Returns a dictionary of transformations indexed by name at a specific time if a node does not exist at that specific time (with tolerance of 1e-6 sec), the transformation is interpolated.
- std::map< std::string, **math::Matrix4** > **GetPoseAtX** (const double _x, const std::string &_node, const bool _loop=true) const
Returns a dictionary of transformations indexed by name where a named node transformation's translational value along the X axis is equal to _x.
- bool **HasNode** (const std::string &_node) const
Looks for a node with a specific name in the animations.
- void **Scale** (const double _scale)
Scales every animation in the animations list.
- void **SetName** (const std::string &_name)
Changes the name.

Protected Attributes

- std::map< std::string, **NodeAnimation** * > **animations**
a dictionary of node animations
- double **length**
the duration of the longest animation
- std::string **name**
the node name

10.225.1 Detailed Description

Skeleton (p. 1024) animation.

10.225.2 Constructor & Destructor Documentation

10.225.2.1 gazebo::common::SkeletonAnimation::SkeletonAnimation (const std::string & *_name*)

The Constructor.

Parameters

in	<i>_name</i>	the name of the animation
----	--------------	---------------------------

10.225.2.2 gazebo::common::SkeletonAnimation::~~SkeletonAnimation ()

The destructor.

Clears the list without destroying the animations

10.225.3 Member Function Documentation

10.225.3.1 void gazebo::common::SkeletonAnimation::AddKeyFrame (const std::string & *_node*, const double *_time*, const math::Matrix4 & *_mat*)

Adds or replaces a named key frame at a specific time.

Parameters

in	<i>_node</i>	the name of the new or existing node
in	<i>_time</i>	the time
in	<i>_mat</i>	the key frame transformation

10.225.3.2 void gazebo::common::SkeletonAnimation::AddKeyFrame (const std::string & *_node*, const double *_time*, const math::Pose & *_pose*)

Adds or replaces a named key frame at a specific time.

Parameters

in	<i>_node</i>	the name of the new or existing node
in	<i>_time</i>	the time
in	<i>_pose</i>	the key frame transformation as a math::Pose (p. 797)

10.225.3.3 double gazebo::common::SkeletonAnimation::GetLength () const

Returns the duration of the animations.

Returns

the duration in seconds

10.225.3.4 `std::string gazebo::common::SkeletonAnimation::GetName () const`

Returns the name.

Returns

the name

10.225.3.5 `unsigned int gazebo::common::SkeletonAnimation::GetNodeCount () const`

Returns the number of animation nodes.

Returns

the count

10.225.3.6 `math::Matrix4 gazebo::common::SkeletonAnimation::GetNodePoseAt (const std::string & _node, const double _time, const bool _loop = true)`

Returns the key frame transformation for a named animation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.

Parameters

in	<code>_node</code>	the name of the animation node
in	<code>_time</code>	the time
in	<code>_loop</code>	when true, the time is divided by the duration (see <code>GetLength</code>)

Returns

the transformation

10.225.3.7 `std::map<std::string, math::Matrix4> gazebo::common::SkeletonAnimation::GetPoseAt (const double _time, const bool _loop = true) const`

Returns a dictionary of transformations indexed by name at a specific time if a node does not exist at that specific time (with tolerance of 1e-6 sec), the transformation is interpolated.

Parameters

in	<code>_time</code>	the time
in	<code>_loop</code>	when true, the time is divided by the duration (see <code>GetLength</code>)

Returns

the transformation for every node

10.225.3.8 `std::map<std::string, math::Matrix4> gazebo::common::SkeletonAnimation::GetPoseAtX (const double _x, const std::string & _node, const bool _loop = true) const`

Returns a dictionary of transformations indexed by name where a named node transformation's translational value along the X axis is equal to *_x*.

Parameters

in	<i>_x</i>	the value along x. You must ensure that <i>_x</i> is within a valid range.
in	<i>_node</i>	the name of the animation node
in	<i>_loop</i>	when true, the time is divided by the duration (see GetLength)

10.225.3.9 `bool gazebo::common::SkeletonAnimation::HasNode (const std::string & _node) const`

Looks for a node with a specific name in the animations.

Parameters

in	<i>_node</i>	the name of the node
----	--------------	----------------------

Returns

true if the node exists

10.225.3.10 `void gazebo::common::SkeletonAnimation::Scale (const double _scale)`

Scales every animation in the animations list.

Parameters

in	<i>_scale</i>	the scaling factor
----	---------------	--------------------

10.225.3.11 `void gazebo::common::SkeletonAnimation::SetName (const std::string & _name)`

Changes the name.

Parameters

in	<i>_name</i>	the new name
----	--------------	--------------

10.225.4 Member Data Documentation

10.225.4.1 `std::map<std::string, NodeAnimation*> gazebo::common::SkeletonAnimation::animations` `[protected]`

a dictionary of node animations

10.225.4.2 double gazebo::common::SkeletonAnimation::length [protected]

the duration of the longest animation

10.225.4.3 std::string gazebo::common::SkeletonAnimation::name [protected]

the node name

The documentation for this class was generated from the following file:

- **SkeletonAnimation.hh**

10.226 gazebo::common::SkeletonNode Class Reference

A skeleton node.

```
#include <common/common.hh>
```

Public Types

- enum **SkeletonNodeType** { **NODE**, **JOINT** }
enumeration of node types

Public Member Functions

- **SkeletonNode** (**SkeletonNode** *_parent)
Constructor.
- **SkeletonNode** (**SkeletonNode** *_parent, std::string _name, std::string _id, **SkeletonNodeType** _type=**JOINT**)
Constructor.
- virtual ~**SkeletonNode** ()
Destructor.
- void **AddChild** (**SkeletonNode** *_child)
Add a new child.
- void **AddRawTransform** (**NodeTransform** _t)
Add a raw transform.
- **SkeletonNode** * **GetChild** (unsigned int _index)
Find a child by index.
- **SkeletonNode** * **GetChildById** (std::string _id)
Get child by string id.
- **SkeletonNode** * **GetChildByName** (std::string _name)
Get child by name.
- unsigned int **GetChildCount** ()
Returns the children count.
- unsigned int **GetHandle** ()
Get the handle index.
- std::string **GetId** ()
Returns the index.

- **math::Matrix4 GetInverseBindTransform ()**
Retrieve the inverse of the bind pose skeletal transform.
- **math::Matrix4 GetModelTransform ()**
Retrieve the model transform.
- **std::string GetName ()**
Returns the name.
- **unsigned int GetNumRawTrans ()**
Return the raw transformations count.
- **SkeletonNode * GetParent ()**
Returns the parent node.
- **NodeTransform GetRawTransform (unsigned int _i)**
Find a raw transformation.
- **std::vector< NodeTransform > GetRawTransforms ()**
Retrieve the raw transformations.
- **math::Matrix4 GetTransform ()**
Get transform relative to parent.
- **std::vector< NodeTransform > GetTransforms ()**
Returns a copy of the array of transformations.
- **bool IsJoint ()**
Is a joint query.
- **bool IsRootNode ()**
Queries wether a node has no parent parent.
- **void Reset (bool _resetChildren)**
Reset the transformation to the initial transformation.
- **void SetHandle (unsigned int _h)**
Assign a handle number.
- **void SetId (std::string _id)**
Change the id string.
- **void SetInitialTransform (math::Matrix4 _tras)**
Sets the initial transformation.
- **void SetInverseBindTransform (math::Matrix4 _invBM)**
Assign the inverse of the bind pose skeletal transform.
- **void SetModelTransform (math::Matrix4 _trans, bool _updateChildren=true)**
Set the model transformation.
- **void SetName (std::string _name)**
Change the name.
- **void SetParent (SkeletonNode *_parent)**
Set the parent node.
- **void SetTransform (math::Matrix4 _trans, bool _updateChildren=true)**
Set a transformation.
- **void SetType (SkeletonNodeType _type)**
Change the skeleton node type.
- **void UpdateChildrenTransforms ()**
Apply model transformations in order for each node in the tree.

Protected Attributes

- `std::vector< SkeletonNode * > children`
the children nodes
- `unsigned int handle`
handle index number
- `std::string id`
a string identifier
- `math::Matrix4 initialTransform`
the initial transformation
- `math::Matrix4 invBindTransform`
the inverse of the bind pose skeletal transform
- `math::Matrix4 modelTransform`
the model transformation
- `std::string name`
the name of the skeletal node
- `SkeletonNode * parent`
the parent node
- `std::vector< NodeTransform > rawTransforms`
the raw transformation
- `math::Matrix4 transform`
the transform
- `SkeletonNodeType type`
the type fo node

10.226.1 Detailed Description

A skeleton node.

10.226.2 Member Enumeration Documentation

10.226.2.1 enum gazebo::common::SkeletonNode::SkeletonNodeType

enumeration of node types

Enumerator:

NODE

JOINT

10.226.3 Constructor & Destructor Documentation

10.226.3.1 gazebo::common::SkeletonNode::SkeletonNode (**SkeletonNode** * *_parent*)

Constructor.

Parameters

<code>in</code>	<code><i>_parent</i></code>	The parent node
-----------------	-----------------------------	-----------------

10.226.3.2 `gazebo::common::SkeletonNode::SkeletonNode (SkeletonNode * _parent, std::string _name, std::string _id, SkeletonNodeType _type = JOINT)`

Constructor.

Parameters

in	<code>_parent</code>	the parent node
in	<code>_name</code>	name of node
in	<code>_id</code>	Id of node
in	<code>_type</code>	The type of this node

10.226.3.3 `virtual gazebo::common::SkeletonNode::~~SkeletonNode () [virtual]`

Destructor.

10.226.4 Member Function Documentation

10.226.4.1 `void gazebo::common::SkeletonNode::AddChild (SkeletonNode * _child)`

Add a new child.

Parameters

in	<code>_child</code>	a child
----	---------------------	---------

10.226.4.2 `void gazebo::common::SkeletonNode::AddRawTransform (NodeTransform _t)`

Add a raw transform.

Parameters

in	<code>_t</code>	the transform
----	-----------------	---------------

10.226.4.3 `SkeletonNode* gazebo::common::SkeletonNode::GetChild (unsigned int _index)`

Find a child by index.

Parameters

in	<code>_index</code>	the index
----	---------------------	-----------

Returns

the child skeleton. NO BOUNDS CHECKING

10.226.4.4 `SkeletonNode* gazebo::common::SkeletonNode::GetChildById (std::string _id)`

Get child by string id.

Parameters

in	<code>_id</code>	the string id
----	------------------	---------------

Returns

the child skeleton or NULL if not found

10.226.4.5 SkeletonNode* gazebo::common::SkeletonNode::GetChildByName (std::string *_name*)

Get child by name.

Parameters

in	<code>_name</code>	the name of the child skeleton
----	--------------------	--------------------------------

Returns

the skeleton, or NULL if not found

10.226.4.6 unsigned int gazebo::common::SkeletonNode::GetChildCount ()

Returns the children count.

Returns

the count

10.226.4.7 unsigned int gazebo::common::SkeletonNode::GetHandle ()

Get the handle index.

Returns

the handle index

10.226.4.8 std::string gazebo::common::SkeletonNode::GetId ()

Returns the index.

Returns

the id string

10.226.4.9 math::Matrix4 gazebo::common::SkeletonNode::GetInverseBindTransform ()

Retrieve the inverse of the bind pose skeletal transform.

Returns

the transform

10.226.4.10 `math::Matrix4 gazebo::common::SkeletonNode::GetModelTransform ()`

Retrieve the model transform.

Returns

the transform

10.226.4.11 `std::string gazebo::common::SkeletonNode::GetName ()`

Returns the name.

Returns

the name

10.226.4.12 `unsigned int gazebo::common::SkeletonNode::GetNumRawTrans ()`

Return the raw transformations count.

Returns

the count

10.226.4.13 `SkeletonNode* gazebo::common::SkeletonNode::GetParent ()`

Returns the parent node.

Returns

the parent

10.226.4.14 `NodeTransform gazebo::common::SkeletonNode::GetRawTransform (unsigned int i)`

Find a raw transformation.

Parameters

<code><i>i</i></code>	<code><i>i</i></code>	the index of the transformation
-----------------------	-----------------------	---------------------------------

Returns

the node transform. NO BOUNDS CHECKING PERFORMED

10.226.4.15 `std::vector<NodeTransform> gazebo::common::SkeletonNode::GetRawTransforms ()`

Retrieve the raw transformations.

Returns

an array of transformations

10.226.4.16 `math::Matrix4 gazebo::common::SkeletonNode::GetTransform ()`

Get transform relative to parent.

10.226.4.17 `std::vector<NodeTransform> gazebo::common::SkeletonNode::GetTransforms ()`

Returns a copy of the array of transformations.

Returns

the array of transform (These are the same as the raw trans)

10.226.4.18 `bool gazebo::common::SkeletonNode::IsJoint ()`

Is a joint query.

Returns

true if the skeleton type is a joint, false otherwise

10.226.4.19 `bool gazebo::common::SkeletonNode::IsRootNode ()`

Queries whether a node has no parent parent.

Returns

true if the node has no parent, false otherwise

10.226.4.20 `void gazebo::common::SkeletonNode::Reset (bool _resetChildren)`

Reset the transformation to the initial transformation.

Parameters

<code>in</code>	<code><i>_resetChildren</i></code>	when true, performs the operation for every node in the tree
-----------------	------------------------------------	--

10.226.4.21 `void gazebo::common::SkeletonNode::SetHandle (unsigned int _h)`

Assign a handle number.

Parameters

<code>in</code>	<code><i>_h</i></code>	the handle
-----------------	------------------------	------------

10.226.4.22 void gazebo::common::SkeletonNode::SetId (std::string *_id*)

Change the id string.

Parameters

in	<i>_id</i>	the new id string
----	------------	-------------------

10.226.4.23 void gazebo::common::SkeletonNode::SetInitialTransform (math::Matrix4 *_tras*)

Sets the initial transformation.

Parameters

in	<i>_tras</i>	the transformation matrix
----	--------------	---------------------------

10.226.4.24 void gazebo::common::SkeletonNode::SetInverseBindTransform (math::Matrix4 *_invBM*)

Assign the inverse of the bind pose skeletal transform.

Parameters

in	<i>_invBM</i>	the transform
----	---------------	---------------

10.226.4.25 void gazebo::common::SkeletonNode::SetModelTransform (math::Matrix4 *_trans*, bool *_updateChildren = true*)

Set the model transformation.

Parameters

in	<i>_trans</i>	the transformation
in	<i>_updateChildren</i>	when true the UpdateChildrenTransforms operation is performed

10.226.4.26 void gazebo::common::SkeletonNode::SetName (std::string *_name*)

Change the name.

Parameters

in	<i>_name</i>	the new name
----	--------------	--------------

10.226.4.27 void gazebo::common::SkeletonNode::SetParent (SkeletonNode * *_parent*)

Set the parent node.

Parameters

in	<i>_parent</i>	the new parent
----	----------------	----------------

10.226.4.28 void gazebo::common::SkeletonNode::SetTransform (math::Matrix4 *_trans*, bool *_updateChildren = true*)

Set a transformation.

Parameters

in	<i>_trans</i>	the transformation
in	<i>_updateChildren</i>	when true the UpdateChildrenTransforms operation is performed

10.226.4.29 void gazebo::common::SkeletonNode::SetType (SkeletonNodeType *_type*)

Change the skeleton node type.

Parameters

in	<i>_type</i>	the new type
----	--------------	--------------

10.226.4.30 void gazebo::common::SkeletonNode::UpdateChildrenTransforms ()

Apply model transformations in order for each node in the tree.

10.226.5 Member Data Documentation

10.226.5.1 std::vector<SkeletonNode*> gazebo::common::SkeletonNode::children [protected]

the children nodes

10.226.5.2 unsigned int gazebo::common::SkeletonNode::handle [protected]

handle index number

10.226.5.3 std::string gazebo::common::SkeletonNode::id [protected]

a string identifier

10.226.5.4 math::Matrix4 gazebo::common::SkeletonNode::initialTransform [protected]

the initial transformation

10.226.5.5 math::Matrix4 gazebo::common::SkeletonNode::invBindTransform [protected]

the inverse of the bind pose skeletal transform

10.226.5.6 math::Matrix4 gazebo::common::SkeletonNode::modelTransform [protected]

the model transformation

10.226.5.7 `std::string gazebo::common::SkeletonNode::name` [protected]

the name of the skeletal node

10.226.5.8 `SkeletonNode* gazebo::common::SkeletonNode::parent` [protected]

the parent node

10.226.5.9 `std::vector<NodeTransform> gazebo::common::SkeletonNode::rawTransforms` [protected]

the raw transformation

10.226.5.10 `math::Matrix4 gazebo::common::SkeletonNode::transform` [protected]

the transform

10.226.5.11 `SkeletonNodeType gazebo::common::SkeletonNode::type` [protected]

the type fo node

The documentation for this class was generated from the following file:

- **Skeleton.hh**

10.227 gazebo::physics::SliderJoint< T > Class Template Reference

A slider joint.

```
#include <physics/physics.hh>
```

Public Member Functions

- **SliderJoint** (**BasePtr** _parent)
Constructor.
- virtual **~SliderJoint** ()
Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (sdf::ElementPtr _sdf)
*Load a **SliderJoint** (p. 1044).*

10.227.1 Detailed Description

```
template<class T>class gazebo::physics::SliderJoint< T >
```

A slider joint.

10.227.2 Constructor & Destructor Documentation

10.227.2.1 `template<class T> gazebo::physics::SliderJoint< T >::SliderJoint (BasePtr _parent) [inline], [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Parent of the joint.
----	----------------------	----------------------

10.227.2.2 `template<class T> virtual gazebo::physics::SliderJoint< T >::~~SliderJoint () [inline], [virtual]`

Destructor.

10.227.3 Member Function Documentation

10.227.3.1 `template<class T> virtual unsigned int gazebo::physics::SliderJoint< T >::GetAngleCount () const [inline], [virtual]`

10.227.3.2 `template<class T> virtual void gazebo::physics::SliderJoint< T >::Load (sdf::ElementPtr _sdf) [inline], [virtual]`

Load a **SliderJoint** (p. 1044).

Parameters

in	<code>_sdf</code>	SDF values to load from
----	-------------------	-------------------------

Reimplemented in **gazebo::physics::SimbodySliderJoint** (p. 1013), and **gazebo::physics::DARTSliderJoint** (p. 374).

The documentation for this class was generated from the following file:

- **SliderJoint.hh**

10.228 gazebo::rendering::GzTerrainMatGen::SM2Profile Class Reference

Shader model 2 profile target.

```
#include <Heightmap.hh>
```

Classes

- class **ShaderHelperCg**
Keeping the CG shader for reference.
- class **ShaderHelperGLSL**
Utility class to help with generating shaders for GLSL.

Public Member Functions

- **SM2Profile** (Ogre::TerrainMaterialGenerator *_parent, const Ogre::String &_name, const Ogre::String &_desc)
Constructor.
- virtual ~**SM2Profile** ()
Destructor.
- Ogre::MaterialPtr **generate** (const Ogre::Terrain *_terrain)
- Ogre::MaterialPtr **generateForCompositeMap** (const Ogre::Terrain *_terrain)
- void **UpdateParams** (const Ogre::MaterialPtr &_mat, const Ogre::Terrain *_terrain)
- void **UpdateParamsForCompositeMap** (const Ogre::MaterialPtr &_mat, const Ogre::Terrain *_terrain)

Protected Member Functions

- virtual void **addTechnique** (const Ogre::MaterialPtr &_mat, const Ogre::Terrain *_terrain, TechniqueType _tt)

10.228.1 Detailed Description

Shader model 2 profile target.

10.228.2 Constructor & Destructor Documentation

- 10.228.2.1 gazebo::rendering::GzTerrainMatGen::SM2Profile::SM2Profile (Ogre::TerrainMaterialGenerator *_parent, const Ogre::String & _name, const Ogre::String & _desc)

Constructor.

- 10.228.2.2 virtual gazebo::rendering::GzTerrainMatGen::SM2Profile::~SM2Profile () [virtual]

Destructor.

10.228.3 Member Function Documentation

- 10.228.3.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::addTechnique (const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain, TechniqueType _tt) [protected],[virtual]
- 10.228.3.2 Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generate (const Ogre::Terrain * _terrain)
- 10.228.3.3 Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generateForCompositeMap (const Ogre::Terrain * _terrain)
- 10.228.3.4 void gazebo::rendering::GzTerrainMatGen::SM2Profile::UpdateParams (const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain)
- 10.228.3.5 void gazebo::rendering::GzTerrainMatGen::SM2Profile::UpdateParamsForCompositeMap (const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain)

The documentation for this class was generated from the following file:

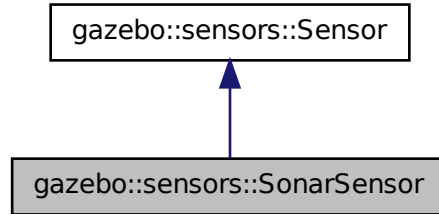
- **Heightmap.hh**

10.229 gazebo::sensors::SonarSensor Class Reference

Sensor (p. 907) with sonar cone.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::SonarSensor:



Public Member Functions

- **SonarSensor** ()
Constructor.
- virtual **~SonarSensor** ()
Destructor.
- template<typename T >
event::ConnectionPtr ConnectUpdate (T _subscriber)
Connect a to the new update signal.
- void **DisconnectUpdate** (**event::ConnectionPtr** &_conn)
Disconnect from the update signal.
- double **GetRadius** () const
Get the radius of the sonar cone at maximum range.
- double **GetRange** ()
Get detected range for a sonar.
- double **GetRangeMax** () const
Get the minimum range of the sonar.
- double **GetRangeMin** () const
Get the minimum range of the sonar.
- virtual std::string **GetTopic** () const
Returns the topic name as set in SDF.
- virtual void **Init** ()
Initialize the sensor.
- virtual bool **IsActive** ()
Returns true if sensor generation is active.
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.

Protected Member Functions

- virtual void **Fini** ()
Finalize the sensor.
- virtual bool **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Protected Attributes

- **event::EventT**< void(msgs::SonarStamped)> **update**
Update event.

10.229.1 Detailed Description

Sensor (p. 907) with sonar cone.

This sensor uses a cone .

10.229.2 Constructor & Destructor Documentation

10.229.2.1 gazebo::sensors::SonarSensor::SonarSensor ()

Constructor.

10.229.2.2 virtual gazebo::sensors::SonarSensor::~~SonarSensor () [virtual]

Destructor.

10.229.3 Member Function Documentation

10.229.3.1 template<typename T > event::ConnectionPtr gazebo::sensors::SonarSensor::ConnectUpdate (T _subscriber) [inline]

Connect a to the new update signal.

Parameters

in	<code>_subscriber</code>	Callback function.
----	--------------------------	--------------------

Returns

The connection, which must be kept in scope.

10.229.3.2 void gazebo::sensors::SonarSensor::DisconnectUpdate (event::ConnectionPtr & _conn) [inline]

Disconnect from the update signal.

Parameters

in	<code>_conn</code>	Connection to remove.
----	--------------------	-----------------------

10.229.3.3 virtual void gazebo::sensors::SonarSensor::Fini () [protected],[virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 912).

10.229.3.4 double gazebo::sensors::SonarSensor::GetRadius () const

Get the radius of the sonar cone at maximum range.

Returns

The radius of the sonar cone at max range.

10.229.3.5 double gazebo::sensors::SonarSensor::GetRange ()

Get detected range for a sonar.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Returns

Returns DBL_MAX for no detection.

10.229.3.6 double gazebo::sensors::SonarSensor::GetRangeMax () const

Get the minimum range of the sonar.

Returns

The sonar's maximum range.

10.229.3.7 double gazebo::sensors::SonarSensor::GetRangeMin () const

Get the minimum range of the sonar.

Returns

The sonar's minimum range.

10.229.3.8 `virtual std::string gazebo::sensors::SonarSensor::GetTopic () const` [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented from `gazebo::sensors::Sensor` (p. 914).

10.229.3.9 `virtual void gazebo::sensors::SonarSensor::Init ()` [virtual]

Initialize the sensor.

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.229.3.10 `virtual bool gazebo::sensors::SonarSensor::IsActive ()` [virtual]

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.229.3.11 `virtual void gazebo::sensors::SonarSensor::Load (const std::string & _worldName)` [virtual]

Load the sensor with default parameters.

Parameters

in	<code>_worldName</code>	Name of world to load from.
----	-------------------------	-----------------------------

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

10.229.3.12 `virtual bool gazebo::sensors::SonarSensor::UpdateImpl (bool)` [protected],[virtual]

This gets overwritten by derived sensor types.

This function is called during `Sensor::Update`.
And in turn, `Sensor::Update` is called by
`SensorManager::Update`

Parameters

in	<code>_force</code>	True if update is forced, false if not
----	---------------------	--

Returns

True if the sensor was updated.

Reimplemented from `gazebo::sensors::Sensor` (p. 917).

10.229.4 Member Data Documentation

10.229.4.1 `event::EventT<void(msgs::SonarStamped)> gazebo::sensors::SonarSensor::update` [protected]

Update event.

The documentation for this class was generated from the following file:

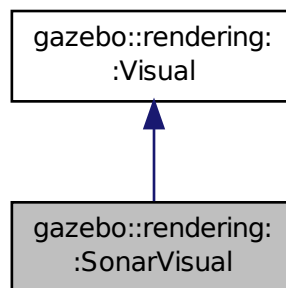
- **SonarSensor.hh**

10.230 gazebo::rendering::SonarVisual Class Reference

Visualization for sonar data.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for `gazebo::rendering::SonarVisual`:

**Public Member Functions**

- **SonarVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)
Constructor.
- virtual **~SonarVisual** ()
Destructor.
- virtual void **Load** ()
Load the visual with default parameters.

Additional Inherited Members

10.230.1 Detailed Description

Visualization for sonar data.

10.230.2 Constructor & Destructor Documentation

10.230.2.1 `gazebo::rendering::SonarVisual::SonarVisual (const std::string & _name, VisualPtr _vis, const std::string & _topicName)`

Constructor.

Parameters

<code>in</code>	<code><i>_name</i></code>	Name of the visual.
<code>in</code>	<code><i>_vis</i></code>	Pointer to the parent Visual (p. 1196).
<code>in</code>	<code><i>_topicName</i></code>	Name of the topic that has sonar data.

10.230.2.2 `virtual gazebo::rendering::SonarVisual::~~SonarVisual () [virtual]`

Destructor.

10.230.3 Member Function Documentation

10.230.3.1 `virtual void gazebo::rendering::SonarVisual::Load () [virtual]`

Load the visual with default parameters.

Reimplemented from `gazebo::rendering::Visual` (p. 1210).

The documentation for this class was generated from the following file:

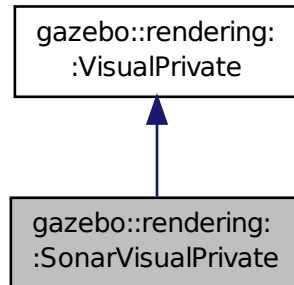
- `SonarVisual.hh`

10.231 gazebo::rendering::SonarVisualPrivate Class Reference

Private data for the Sonar **Visual** (p. 1196) class.

```
#include <SonarVisualPrivate.hh>
```


Inheritance diagram for gazebo::rendering::SonarVisualPrivate:



Public Attributes

- Ogre::SceneNode * **coneNode**
Renders the sonar cone.
- std::vector< **event::ConnectionPtr** > **connections**
All the event connections.
- boost::mutex **mutex**
Mutex to protect the contact message.
- **transport::NodePtr** **node**
Pointer to a node that handles communication.
- bool **receivedMsg**
True if we have received a message.
- boost::shared_ptr
< msgs::SonarStamped const > **sonarMsg**
The current sonar message.
- **DynamicLines** * **sonarRay**
Renders the sonar data reading.
- **transport::SubscriberPtr** **sonarSub**
Subscription to the sonar data.

Additional Inherited Members

10.231.1 Detailed Description

Private data for the Sonar **Visual** (p. 1196) class.

10.231.2 Member Data Documentation

10.231.2.1 `Ogre::SceneNode*` `gazebo::rendering::SonarVisualPrivate::coneNode`

Renders the sonar cone.

10.231.2.2 `std::vector<event::ConnectionPtr>` `gazebo::rendering::SonarVisualPrivate::connections`

All the event connections.

10.231.2.3 `boost::mutex` `gazebo::rendering::SonarVisualPrivate::mutex`

Mutex to protect the contact message.

10.231.2.4 `transport::NodePtr` `gazebo::rendering::SonarVisualPrivate::node`

Pointer to a node that handles communication.

10.231.2.5 `bool` `gazebo::rendering::SonarVisualPrivate::receivedMsg`

True if we have received a message.

10.231.2.6 `boost::shared_ptr<msgs::SonarStamped const>` `gazebo::rendering::SonarVisualPrivate::sonarMsg`

The current sonar message.

10.231.2.7 `DynamicLines*` `gazebo::rendering::SonarVisualPrivate::sonarRay`

Renders the sonar data reading.

10.231.2.8 `transport::SubscriberPtr` `gazebo::rendering::SonarVisualPrivate::sonarSub`

Subscription to the sonar data.

The documentation for this class was generated from the following file:

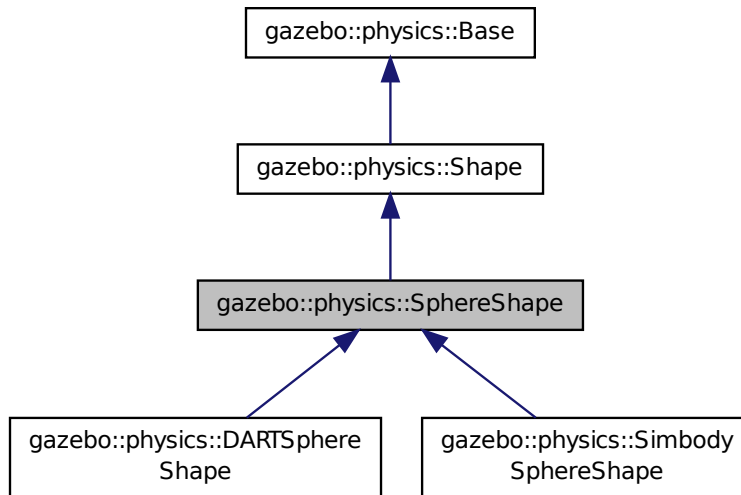
- `SonarVisualPrivate.hh`

10.232 `gazebo::physics::SphereShape` Class Reference

Sphere collision shape.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::SphereShape:



Public Member Functions

- **SphereShape** (*CollisionPtr* _parent)
Constructor.
- virtual \sim **SphereShape** ()
Destructor.
- virtual void **FillMsg** (*msgs::Geometry* &_msg)
Fill in the values for a geometry message.
- double **GetRadius** () const
Get the sphere's radius.
- virtual void **Init** ()
Initialize the sphere.
- virtual void **ProcessMsg** (const *msgs::Geometry* &_msg)
Process a geometry message.
- virtual void **SetRadius** (double _radius)
Set the size.
- virtual void **SetScale** (const *math::Vector3* &_scale)
Set the scale of the sphere.

Additional Inherited Members

10.232.1 Detailed Description

Sphere collision shape.

10.232.2 Constructor & Destructor Documentation

10.232.2.1 `gazebo::physics::SphereShape::SphereShape (CollisionPtr _parent) [explicit]`

Constructor.

Parameters

in	_parent	Parent collision object.
----	---------	--------------------------

10.232.2.2 `virtual gazebo::physics::SphereShape::~~SphereShape () [virtual]`

Destructor.

10.232.3 Member Function Documentation

10.232.3.1 `virtual void gazebo::physics::SphereShape::FillMsg (msgs::Geometry & _msg) [virtual]`

Fill in the values for a geometry message.

Parameters

out	_msg	The geometry message to fill.
-----	------	-------------------------------

Implements **gazebo::physics::Shape** (p. 934).

10.232.3.2 `double gazebo::physics::SphereShape::GetRadius () const`

Get the sphere's radius.

Returns

Radius of the sphere.

10.232.3.3 `virtual void gazebo::physics::SphereShape::Init () [virtual]`

Initialize the sphere.

Implements **gazebo::physics::Shape** (p. 935).

10.232.3.4 `virtual void gazebo::physics::SphereShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]`

Process a geometry message.

Parameters

in	_msg	The message to set values from.
----	------	---------------------------------

Implements **gazebo::physics::Shape** (p. 935).

10.232.3.5 virtual void gazebo::physics::SphereShape::SetRadius (double *_radius*) [virtual]

Set the size.

Parameters

in	<i>_radius</i>	Radius of the sphere.
----	----------------	-----------------------

Reimplemented in **gazebo::physics::SimbodySphereShape** (p. 1015), and **gazebo::physics::DARTSphereShape** (p. 377).

Referenced by gazebo::physics::DARTSphereShape::SetRadius(), and gazebo::physics::SimbodySphereShape::SetRadius().

10.232.3.6 virtual void gazebo::physics::SphereShape::SetScale (const math::Vector3 & *_scale*) [virtual]

Set the scale of the sphere.

Parameters

in	<i>_scale</i>	Scale to set the sphere to.
----	---------------	-----------------------------

Implements **gazebo::physics::Shape** (p. 935).

The documentation for this class was generated from the following file:

- **SphereShape.hh**

10.233 gazebo::common::SphericalCoordinates Class Reference

Convert spherical coordinates for planetary surfaces.

```
#include <common/common.hh>
```

Public Types

- enum **SurfaceType** { **EARTH_WGS84** = 1 }
Unique identifiers for planetary surface models.

Public Member Functions

- **SphericalCoordinates** ()
Constructor.
- **SphericalCoordinates** (const **SurfaceType** *_type*)
Constructor with surface type input.
- **SphericalCoordinates** (const **SurfaceType** *_type*, const **math::Angle** & *_latitude*, const **math::Angle** & *_longitude*, double *_elevation*, const **math::Angle** & *_heading*)
Constructor with surface type, angle, and elevation inputs.
- **~SphericalCoordinates** ()
Destructor.

- double **GetElevationReference** () const
Get reference elevation in meters.
- **math::Angle GetHeadingOffset** () const
Get heading offset for gazebo reference frame, expressed as angle from East to gazebo x-axis, or equivalently from North to gazebo y-axis.
- **math::Angle GetLatitudeReference** () const
Get reference geodetic latitude.
- **math::Angle GetLongitudeReference** () const
Get reference longitude.
- **SurfaceType GetSurfaceType** () const
Get SurfaceType currently in use.
- **math::Vector3 GlobalFromLocal** (const **math::Vector3** &_xyz) const
Convert a Cartesian velocity vector in the local gazebo frame to a global Cartesian frame with components East, North, Up.
- void **SetElevationReference** (double _elevation)
Set reference elevation above sea level in meters.
- void **SetHeadingOffset** (const **math::Angle** &_angle)
Set heading angle offset for gazebo frame.
- void **SetLatitudeReference** (const **math::Angle** &_angle)
Set reference geodetic latitude.
- void **SetLongitudeReference** (const **math::Angle** &_angle)
Set reference longitude.
- void **SetSurfaceType** (const **SurfaceType** &_type)
Set SurfaceType for planetary surface model.
- **math::Vector3 SphericalFromLocal** (const **math::Vector3** &_xyz) const
Convert a Cartesian position vector to geodetic coordinates.

Static Public Member Functions

- static **SurfaceType Convert** (const std::string &_str)
Convert a string to a SurfaceType.
- static double **Distance** (const **math::Angle** &_latA, const **math::Angle** &_lonA, const **math::Angle** &_latB, const **math::Angle** &_lonB)
Get the distance between two points expressed in geographic latitude and longitude.

10.233.1 Detailed Description

Convert spherical coordinates for planetary surfaces.

10.233.2 Member Enumeration Documentation

10.233.2.1 enum gazebo::common::SphericalCoordinates::SurfaceType

Unique identifiers for planetary surface models.

Enumerator:

EARTH_WGS84 Model of reference ellipsoid for earth, based on WGS 84 standard. see wikipedia: World_Geodetic_System

10.233.3 Constructor & Destructor Documentation

10.233.3.1 gazebo::common::SphericalCoordinates::SphericalCoordinates ()

Constructor.

10.233.3.2 gazebo::common::SphericalCoordinates::SphericalCoordinates (const SurfaceType *_type*)

Constructor with surface type input.

Parameters

in	<i>_type</i>	SurfaceType specification.
----	--------------	----------------------------

10.233.3.3 gazebo::common::SphericalCoordinates::SphericalCoordinates (const SurfaceType *_type*, const math::Angle & *_latitude*, const math::Angle & *_longitude*, double *_elevation*, const math::Angle & *_heading*)

Constructor with surface type, angle, and elevation inputs.

Parameters

in	<i>_type</i>	SurfaceType specification.
in	<i>_latitude</i>	Reference latitude.
in	<i>_longitude</i>	Reference longitude.
in	<i>_elevation</i>	Reference elevation.
in	<i>_heading</i>	Heading offset.

10.233.3.4 gazebo::common::SphericalCoordinates::~~SphericalCoordinates ()

Destructor.

10.233.4 Member Function Documentation

10.233.4.1 static SurfaceType gazebo::common::SphericalCoordinates::Convert (const std::string & *_str*) [static]

Convert a string to a SurfaceType.

Parameters

in	<i>_str</i>	String to convert.
----	-------------	--------------------

Returns

Conversion to SurfaceType.

10.233.4.2 static double gazebo::common::SphericalCoordinates::Distance (const math::Angle & *_latA*, const math::Angle & *_lonA*, const math::Angle & *_latB*, const math::Angle & *_lonB*) [static]

Get the distance between two points expressed in geographic latitude and longitude.

It assumes that both points are at sea level. Example: `_latA = 38.0016667` and `_lonA = -123.0016667` represents the point with latitude 38d 0'6.00"N and longitude 123d 0'6.00"W.

Parameters

<code>in</code>	<code>_latA</code>	Latitude of point A.
<code>in</code>	<code>_longA</code>	Longitude of point A.
<code>in</code>	<code>_latB</code>	Latitude of point B.
<code>in</code>	<code>_longB</code>	Longitude of point B.

Returns

Distance in meters.

10.233.4.3 `double gazebo::common::SphericalCoordinates::GetElevationReference () const`

Get reference elevation in meters.

Returns

Reference elevation.

10.233.4.4 `math::Angle gazebo::common::SphericalCoordinates::GetHeadingOffset () const`

Get heading offset for gazebo reference frame, expressed as angle from East to gazebo x-axis, or equivalently from North to gazebo y-axis.

Returns

Heading offset of gazebo reference frame.

10.233.4.5 `math::Angle gazebo::common::SphericalCoordinates::GetLatitudeReference () const`

Get reference geodetic latitude.

Returns

Reference geodetic latitude.

10.233.4.6 `math::Angle gazebo::common::SphericalCoordinates::GetLongitudeReference () const`

Get reference longitude.

Returns

Reference longitude.

10.233.4.7 **SurfaceType** gazebo::common::SphericalCoordinates::GetSurfaceType () const

Get SurfaceType currently in use.

Returns

Current SurfaceType value.

10.233.4.8 **math::Vector3** gazebo::common::SphericalCoordinates::GlobalFromLocal (const math::Vector3 & *_xyz*) const

Convert a Cartesian velocity vector in the local gazebo frame to a global Cartesian frame with components East, North, Up.

Parameters

<i>in</i>	<i>_xyz</i>	Cartesian vector in gazebo's world frame.
-----------	-------------	---

Returns

Rotated vector with components (x,y,z): (East, North, Up).

10.233.4.9 void gazebo::common::SphericalCoordinates::SetElevationReference (double *_elevation*)

Set reference elevation above sea level in meters.

Parameters

<i>in</i>	<i>_elevation</i>	Reference elevation.
-----------	-------------------	----------------------

10.233.4.10 void gazebo::common::SphericalCoordinates::SetHeadingOffset (const math::Angle & *_angle*)

Set heading angle offset for gazebo frame.

Parameters

<i>in</i>	<i>_angle</i>	Heading offset for gazebo frame.
-----------	---------------	----------------------------------

10.233.4.11 void gazebo::common::SphericalCoordinates::SetLatitudeReference (const math::Angle & *_angle*)

Set reference geodetic latitude.

Parameters

<i>in</i>	<i>_angle</i>	Reference geodetic latitude.
-----------	---------------	------------------------------

10.233.4.12 void gazebo::common::SphericalCoordinates::SetLongitudeReference (const math::Angle & *_angle*)

Set reference longitude.

Parameters

in	_angle	Reference longitude.
----	--------	----------------------

10.233.4.13 `void gazebo::common::SphericalCoordinates::SetSurfaceType (const SurfaceType & _type)`

Set SurfaceType for planetary surface model.

Parameters

in	_type	SurfaceType value.
----	-------	--------------------

10.233.4.14 `math::Vector3 gazebo::common::SphericalCoordinates::SphericalFromLocal (const math::Vector3 & _xyz) const`

Convert a Cartesian position vector to geodetic coordinates.

Parameters

in	_xyz	Cartesian position vector in gazebo's world frame.
----	------	--

Returns

Coordinates: geodetic latitude (deg), longitude (deg), altitude above sea level (m).

The documentation for this class was generated from the following file:

- **SphericalCoordinates.hh**

10.234 gazebo::common::SphericalCoordinatesPrivate Class Reference

common/common.hh

```
#include <SphericalCoordinatesPrivate.hh>
```

Public Attributes

- double **elevationReference**
Elevation of reference point relative to sea level in meters.
- **math::Angle headingOffset**
Heading offset, expressed as angle from East to gazebo x-axis, or equivalently from North to gazebo y-axis.
- **math::Angle latitudeReference**
Latitude of reference point.
- **math::Angle longitudeReference**
Longitude of reference point.
- **SphericalCoordinates::SurfaceType surfaceType**
Type of surface being used.

10.234.1 Detailed Description

common/common.hh

Private data for the **SphericalCoordinates** (p. 1057) class.

10.234.2 Member Data Documentation

10.234.2.1 double gazebo::common::SphericalCoordinatesPrivate::elevationReference

Elevation of reference point relative to sea level in meters.

10.234.2.2 math::Angle gazebo::common::SphericalCoordinatesPrivate::headingOffset

Heading offset, expressed as angle from East to gazebo x-axis, or equivalently from North to gazebo y-axis.

10.234.2.3 math::Angle gazebo::common::SphericalCoordinatesPrivate::latitudeReference

Latitude of reference point.

10.234.2.4 math::Angle gazebo::common::SphericalCoordinatesPrivate::longitudeReference

Longitude of reference point.

10.234.2.5 SphericalCoordinates::SurfaceType gazebo::common::SphericalCoordinatesPrivate::surfaceType

Type of surface being used.

The documentation for this class was generated from the following file:

- **SphericalCoordinatesPrivate.hh**

10.235 gazebo::math::Spline Class Reference

Splines.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Spline** ()
constructor
- **~Spline** ()
destructor
- void **AddPoint** (const **Vector3** &_pt)
Adds a control point to the end of the spline.
- void **Clear** ()
Clears all the points in the spline.

- **Vector3 GetPoint** (unsigned int `_index`) const
Gets the detail of one of the control points of the spline.
- unsigned int **GetPointCount** () const
Gets the number of control points in the spline.
- **Vector3 GetTangent** (unsigned int `_index`) const
Get the tangent value for a point.
- double **GetTension** () const
Get the tension value.
- **Vector3 Interpolate** (double `_t`) const
Returns an interpolated point based on a parametric value over the whole series.
- **Vector3 Interpolate** (unsigned int `_fromIndex`, double `_t`) const
Interpolates a single segment of the spline given a parametric value.
- void **RecalcTangents** ()
Recalculates the tangents associated with this spline.
- void **SetAutoCalculate** (bool `_autoCalc`)
Tells the spline whether it should automatically calculate tangents on demand as points are added.
- void **SetTension** (double `_t`)
Set the tension parameter.
- void **UpdatePoint** (unsigned int `_index`, const **Vector3** &`_value`)
Updates a single point in the spline.

Protected Attributes

- bool **autoCalc**
when true, the tangents are recalculated when the control point change
- **Matrix4 coeffs**
Matrix of coefficients.
- std::vector< **Vector3** > **points**
control points
- std::vector< **Vector3** > **tangents**
tangents
- double **tension**
Tension of 0 = Catmull-Rom spline, otherwise a Cardinal spline.

10.235.1 Detailed Description

Splines.

10.235.2 Constructor & Destructor Documentation

10.235.2.1 gazebo::math::Spline::Spline ()

constructor

10.235.2.2 gazebo::math::Spline::~~Spline ()

destructor

10.235.3 Member Function Documentation

10.235.3.1 void gazebo::math::Spline::AddPoint (const Vector3 & _pt)

Adds a control point to the end of the spline.

Parameters

in	<i>_pt</i>	point to add
----	------------	--------------

10.235.3.2 void gazebo::math::Spline::Clear ()

Clears all the points in the spline.

10.235.3.3 Vector3 gazebo::math::Spline::GetPoint (unsigned int *_index*) const

Gets the detail of one of the control points of the spline.

Parameters

in	<i>_index</i>	the control point index
----	---------------	-------------------------

Returns

the control point, or [0,0,0] and a message on the error stream

10.235.3.4 unsigned int gazebo::math::Spline::GetPointCount () const

Gets the number of control points in the spline.

Returns

the count

10.235.3.5 Vector3 gazebo::math::Spline::GetTangent (unsigned int *_index*) const

Get the tangent value for a point.

Parameters

in	<i>_index</i>	the control point index
----	---------------	-------------------------

10.235.3.6 double gazebo::math::Spline::GetTension () const

Get the tension value.

Returns

The value of the tension, which is between 0.0 and 1.0

10.235.3.7 Vector3 gazebo::math::Spline::Interpolate (double *_t*) const

Returns an interpolated point based on a parametric value over the whole series.

Parameters

<i>in</i>	<i>_t</i>	parameter (range 0 to 1)
-----------	-----------	--------------------------

10.235.3.8 Vector3 gazebo::math::Spline::Interpolate (unsigned int *_fromIndex*, double *_t*) const

Interpolates a single segment of the spline given a parametric value.

Parameters

<i>in</i>	<i>_fromIndex</i>	The point index to treat as t = 0. fromIndex + 1 is deemed to be t = 1
<i>in</i>	<i>_t</i>	Parametric value

10.235.3.9 void gazebo::math::Spline::RecalcTangents ()

Recalculates the tangents associated with this spline.

Remarks

If you tell the spline not to update on demand by calling `setAutoCalculate(false)` then you must call this after completing your updates to the spline points.

10.235.3.10 void gazebo::math::Spline::SetAutoCalculate (bool *_autoCalc*)

Tells the spline whether it should automatically calculate tangents on demand as points are added.

Remarks

The spline calculates tangents at each point automatically based on the input points. Normally it does this every time a point changes. However, if you have a lot of points to add in one go, you probably don't want to incur this overhead and would prefer to defer the calculation until you are finished setting all the points. You can do this by calling this method with a parameter of 'false'. Just remember to manually call the `recalcTangents` method when you are done.

Parameters

<i>in</i>	<i>_autoCalc</i>	If true, tangents are calculated for you whenever a point changes. If false, you must call <code>recalcTangents</code> to recalculate them when it best suits.
-----------	------------------	--

10.235.3.11 void gazebo::math::Spline::SetTension (double *_t*)

Set the tension parameter.

A value of 0 = Catmull-Rom spline.

Parameters

<i>in</i>	<i>_t</i>	Tension value between 0.0 and 1.0
-----------	-----------	-----------------------------------

10.235.3.12 void gazebo::math::Spline::UpdatePoint (unsigned int *_index*, const Vector3 & *_value*)

Updates a single point in the spline.

Remarks

an error to the error stream is printed when the index is out of bounds

Parameters

<i>in</i>	<i>_index</i>	the control point index
<i>in</i>	<i>_value</i>	the new position

10.235.4 Member Data Documentation

10.235.4.1 bool gazebo::math::Spline::autoCalc [protected]

when true, the tangents are recalculated when the control point change

10.235.4.2 Matrix4 gazebo::math::Spline::coeffs [protected]

Matrix of coefficients.

10.235.4.3 std::vector<Vector3> gazebo::math::Spline::points [protected]

control points

10.235.4.4 std::vector<Vector3> gazebo::math::Spline::tangents [protected]

tangents

10.235.4.5 double gazebo::math::Spline::tension [protected]

Tension of 0 = Catmull-Rom spline, otherwise a Cardinal spline.

The documentation for this class was generated from the following file:

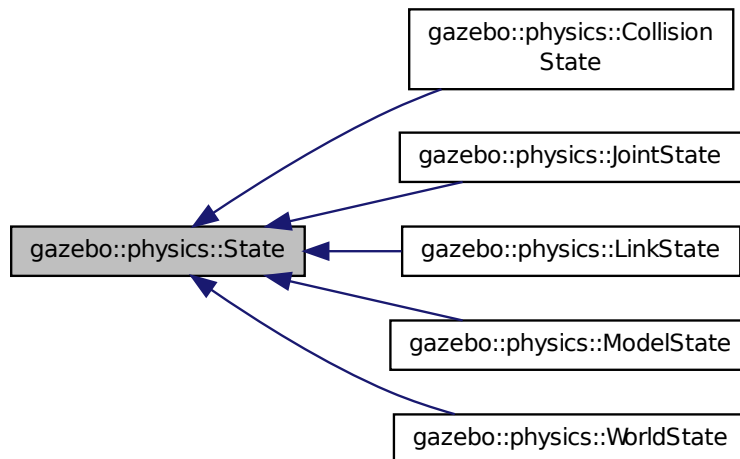
- **Spline.hh**

10.236 gazebo::physics::State Class Reference

State (p. 1068) of an entity.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::State:



Public Member Functions

- **State** ()
Default constructor.
- **State** (const std::string &_name, const **common::Time** &_realTime, const **common::Time** &_simTime)
Constructor.
- virtual ~**State** ()
Destructor.
- std::string **GetName** () const
*Get the name associated with this **State** (p. 1068).*
- **common::Time** **GetRealTime** () const
Get the real time when this state was generated.
- **common::Time** **GetSimTime** () const
Get the sim time when this state was generated.
- **common::Time** **GetWallTime** () const
Get the wall time when this state was generated.
- virtual void **Load** (const sdf::ElementPtr _elem)
Load state from SDF element.
- **State operator-** (const **State** &_state) const
Subtraction operator.
- **State & operator=** (const **State** &_state)

Assignment operator.

- void **SetName** (const std::string &_name)
*Set the name associated with this **State** (p. 1068).*
- virtual void **SetRealTime** (const **common::Time** &_time)
Set the real time when this state was generated.
- virtual void **SetSimTime** (const **common::Time** &_time)
Set the sim time when this state was generated.
- virtual void **SetWallTime** (const **common::Time** &_time)
Set the wall time when this state was generated.

Protected Attributes

- std::string **name**
*Name associated with this **State** (p. 1068).*
- **common::Time** **realTime**
- **common::Time** **simTime**
- **common::Time** **wallTime**
Times for the state data.

10.236.1 Detailed Description

State (p. 1068) of an entity.

This is the base class for all **State** (p. 1068) information.

10.236.2 Constructor & Destructor Documentation

10.236.2.1 gazebo::physics::State::State ()

Default constructor.

10.236.2.2 gazebo::physics::State::State (const std::string & _name, const **common::Time** & _realTime, const **common::Time** & _simTime)

Constructor.

Construct a **State** (p. 1068) object using some basic information.

Parameters

<code>_name</code>	Name associated with the State (p. 1068) information. This is typically the name of an Entity (p. 404). <code>_realTime</code> Clock time since simulation started.
<code>_simTime</code>	Simulation time associated with this State (p. 1068) info.

10.236.2.3 virtual gazebo::physics::State::~~State () [virtual]

Destructor.

10.236.3 Member Function Documentation

10.236.3.1 `std::string gazebo::physics::State::GetName () const`

Get the name associated with this **State** (p. 1068).

Returns

Name associated with this state information. Typically a name of an **Entity** (p. 404).

10.236.3.2 `common::Time gazebo::physics::State::GetRealTime () const`

Get the real time when this state was generated.

Returns

Clock time since simulation was stated.

10.236.3.3 `common::Time gazebo::physics::State::GetSimTime () const`

Get the sim time when this state was generated.

Returns

Simulation time when the data was recorded.

10.236.3.4 `common::Time gazebo::physics::State::GetWallTime () const`

Get the wall time when this state was generated.

Returns

The absolute clock time when the **State** (p. 1068) data was recorded.

10.236.3.5 `virtual void gazebo::physics::State::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Populates the **State** (p. 1068) information from data stored in an `SDF::Element`

Parameters

<code><i>_elem</i></code>	Pointer to the <code>SDF::Element</code>
---------------------------	--

Reimplemented in `gazebo::physics::ModelState` (p. 704), `gazebo::physics::LinkState` (p. 623), `gazebo::physics::WorldState` (p. 1257), `gazebo::physics::JointState` (p. 577), and `gazebo::physics::CollisionState` (p. 248).

10.236.3.6 State gazebo::physics::State::operator- (const State & *_state*) const

Subtraction operator.

Parameters

in	<i>_pt</i>	A state to subtract.
----	------------	----------------------

Returns

The resulting state.

10.236.3.7 State& gazebo::physics::State::operator= (const State & *_state*)

Assignment operator.

Parameters

in	<i>_state</i>	State (p. 1068) value
----	---------------	------------------------------

Returns

this

10.236.3.8 void gazebo::physics::State::SetName (const std::string & *_name*)

Set the name associated with this **State** (p. 1068).

Parameters

in	<i>_name</i>	Name associated with this state information. Typically the name of an Entity (p. 404).
----	--------------	---

10.236.3.9 virtual void gazebo::physics::State::SetRealTime (const common::Time & *_time*) [virtual]

Set the real time when this state was generated.

Parameters

in	<i>_time</i>	Clock time since simulation was stated.
----	--------------	---

Reimplemented in **gazebo::physics::ModelState** (p. 705), **gazebo::physics::LinkState** (p. 624), and **gazebo::physics::WorldState** (p. 1258).

10.236.3.10 virtual void gazebo::physics::State::SetSimTime (const common::Time & *_time*) [virtual]

Set the sim time when this state was generated.

Parameters

in	_time	Simulation time when the data was recorded.
----	-------	---

Reimplemented in **gazebo::physics::ModelState** (p. 705), **gazebo::physics::LinkState** (p. 624), and **gazebo::physics::WorldState** (p. 1258).

10.236.3.11 `virtual void gazebo::physics::State::SetWallTime (const common::Time & _time) [virtual]`

Set the wall time when this state was generated.

Parameters

in	_time	The absolute clock time when the State (p. 1068) data was recorded.
----	-------	--

Reimplemented in **gazebo::physics::ModelState** (p. 706), **gazebo::physics::LinkState** (p. 625), and **gazebo::physics::WorldState** (p. 1258).

10.236.4 Member Data Documentation

10.236.4.1 `std::string gazebo::physics::State::name [protected]`

Name associated with this **State** (p. 1068).

10.236.4.2 `common::Time gazebo::physics::State::realTime [protected]`

10.236.4.3 `common::Time gazebo::physics::State::simTime [protected]`

10.236.4.4 `common::Time gazebo::physics::State::wallTime [protected]`

Times for the state data.

The documentation for this class was generated from the following file:

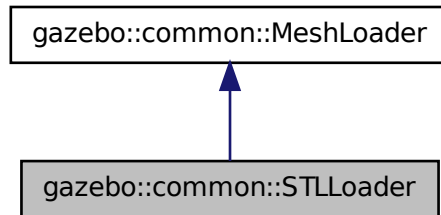
- **State.hh**

10.237 gazebo::common::STLLoader Class Reference

Class used to load STL mesh files.

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::common::STLloader:



Public Member Functions

- **STLloader** ()
Constructor.
- virtual **~STLloader** ()
Destructor.
- virtual **Mesh * Load** (const std::string &_filename)
Creates a new mesh and loads the data from a file.

10.237.1 Detailed Description

Class used to load STL mesh files.

10.237.2 Constructor & Destructor Documentation

10.237.2.1 gazebo::common::STLloader::STLloader ()

Constructor.

10.237.2.2 virtual gazebo::common::STLloader::~~STLloader () [virtual]

Destructor.

10.237.3 Member Function Documentation

10.237.3.1 virtual Mesh* gazebo::common::STLloader::Load (const std::string &_filename) [virtual]

Creates a new mesh and loads the data from a file.

Parameters

in	_filename	the mesh file
----	-----------	---------------

Implements `gazebo::common::MeshLoader` (p. 670).

The documentation for this class was generated from the following file:

- `STLloader.hh`

10.238 gazebo::common::SubMesh Class Reference

A child mesh.

```
#include <Mesh.hh>
```

Public Types

- enum `PrimitiveType` {
POINTS, LINES, LINSTRIPS, TRIANGLES,
TRIFANS, TRISTRIPS }
An enumeration of the geometric mesh primitives.

Public Member Functions

- `SubMesh ()`
Constructor.
- `SubMesh (const SubMesh *_mesh)`
Copy Constructor.
- `virtual ~SubMesh ()`
Destructor.
- `void AddIndex (unsigned int _i)`
Add an index to the mesh.
- `void AddNodeAssignment (unsigned int _vertex, unsigned int _node, float _weight)`
Add a vertex - skeleton node assignment.
- `void AddNormal (const math::Vector3 &_n)`
Add a normal to the mesh.
- `void AddNormal (double _x, double _y, double _z)`
Add a normal to the mesh.
- `void AddTexCoord (double _u, double _v)`
Add a texture coord to the mesh.
- `void AddVertex (const math::Vector3 &_v)`
Add a vertex to the mesh.
- `void AddVertex (double _x, double _y, double _z)`
Add a vertex to the mesh.
- `void Center (const math::Vector3 &_center=math::Vector3::Zero)`
Move the center of the submesh to the given coordinate.
- `void CopyNormals (const std::vector< math::Vector3 > &_norms)`
Copy normals from a vector.
- `void CopyVertices (const std::vector< math::Vector3 > &_verts)`
Copy vertices from a vector.
- `void FillArrays (float **_vertArr, int **_indArr) const`

Put all the data into flat arrays.

- void **GenSphericalTexCoord** (const **math::Vector3** &_center)
Generate texture coordinates using spherical projection from center.
- unsigned int **GetIndex** (unsigned int _i) const
Get an index.
- unsigned int **GetIndexCount** () const
Return the number of indicies.
- unsigned int **GetMaterialIndex** () const
Get the material index.
- **math::Vector3** **GetMax** () const
Get the maximum X, Y, Z values.
- unsigned int **GetMaxIndex** () const
Get the highest index value.
- **math::Vector3** **GetMin** () const
Get the minimum X, Y, Z values.
- std::string **GetName** () const
Get the name of this mesh.
- **NodeAssignment** **GetNodeAssignment** (unsigned int _i) const
Get a vertex - skeleton node assignment.
- unsigned int **GetNodeAssignmentsCount** () const
Return the number of vertex - skeleton node assignments.
- **math::Vector3** **GetNormal** (unsigned int _i) const
Get a normal.
- unsigned int **GetNormalCount** () const
Return the number of normals.
- **PrimitiveType** **GetPrimitiveType** () const
Get the primitive type.
- **math::Vector2d** **GetTexCoord** (unsigned int _i) const
Get a tex coord.
- unsigned int **GetTexCoordCount** () const
Return the number of texture coordinates.
- **math::Vector3** **GetVertex** (unsigned int _i) const
Get a vertex.
- unsigned int **GetVertexCount** () const
Return the number of vertices.
- unsigned int **GetVertexIndex** (const **math::Vector3** &_v) const
Get the index of the vertex.
- bool **HasVertex** (const **math::Vector3** &_v) const
Return true if this submesh has the vertex.
- void **RecalculateNormals** ()
Recalculate all the normals.
- void **Scale** (double _factor)
Scale all vertices by _factor.
- void **SetIndexCount** (unsigned int _count)
Resize the index array.
- void **SetMaterialIndex** (unsigned int _index)
Set the material index.

- void **SetName** (const std::string &_n)
Set the name of this mesh.
- void **SetNormal** (unsigned int _i, const **math::Vector3** &_n)
Set a normal.
- void **SetNormalCount** (unsigned int _count)
Resize the normal array.
- void **SetPrimitiveType** (**PrimitiveType** _type)
Set the primitive type.
- void **SetScale** (const **math::Vector3** &_factor)
Scale all vertices by the _factor vector.
- void **SetSubMeshCenter** (**math::Vector3** _center)
Reset mesh center to geometric center.
- void **SetTexCoord** (unsigned int _i, const **math::Vector2d** &_t)
Set a tex coord.
- void **SetTexCoordCount** (unsigned int _count)
Resize the texture coordinate array.
- void **SetVertex** (unsigned int _i, const **math::Vector3** &_v)
Set a vertex.
- void **SetVertexCount** (unsigned int _count)
Resize the vertex array.
- void **Translate** (const **math::Vector3** &_vec)
Move all vertices by _vec.

10.238.1 Detailed Description

A child mesh.

10.238.2 Member Enumeration Documentation

10.238.2.1 enum gazebo::common::SubMesh::PrimitiveType

An enumeration of the geometric mesh primitives.

Enumerator:

POINTS
LINES
LINESTRIPS
TRIANGLES
TRIFANS
TRISTRIPS

10.238.3 Constructor & Destructor Documentation

10.238.3.1 gazebo::common::SubMesh::SubMesh ()

Constructor.

10.238.3.2 gazebo::common::SubMesh::SubMesh (const SubMesh * *_mesh*)

Copy Constructor.

10.238.3.3 virtual gazebo::common::SubMesh::~~SubMesh () [virtual]

Destructor.

10.238.4 Member Function Documentation

10.238.4.1 void gazebo::common::SubMesh::AddIndex (unsigned int *_i*)

Add an index to the mesh.

Parameters

in	<i>_i</i>	the new vertex index
----	-----------	----------------------

10.238.4.2 void gazebo::common::SubMesh::AddNodeAssignment (unsigned int *_vertex*, unsigned int *_node*, float *_weight*)

Add a vertex - skeleton node assignment.

Parameters

in	<i>_vertex</i>	the vertex index
in	<i>_node</i>	the node index
in	<i>_weight</i>	the weight (between 0 and 1)

10.238.4.3 void gazebo::common::SubMesh::AddNormal (const math::Vector3 & *_n*)

Add a normal to the mesh.

Parameters

in	<i>_n</i>	the normal
----	-----------	------------

10.238.4.4 void gazebo::common::SubMesh::AddNormal (double *_x*, double *_y*, double *_z*)

Add a normal to the mesh.

Parameters

in	<i>_x</i>	position along x
in	<i>_y</i>	position along y
in	<i>_z</i>	position along z

10.238.4.5 void gazebo::common::SubMesh::AddTexCoord (double *_u*, double *_v*)

Add a texture coord to the mesh.

Parameters

in	<i>_u</i>	position along u
in	<i>_v</i>	position along v

10.238.4.6 void gazebo::common::SubMesh::AddVertex (const math::Vector3 & *_v*)

Add a vertex to the mesh.

Parameters

in	<i>_v</i>	the new position
----	-----------	------------------

10.238.4.7 void gazebo::common::SubMesh::AddVertex (double *_x*, double *_y*, double *_z*)

Add a vertex to the mesh.

Parameters

in	<i>_x</i>	position along x
in	<i>_y</i>	position along y
in	<i>_z</i>	position along z

10.238.4.8 void gazebo::common::SubMesh::Center (const math::Vector3 & *_center* = math::Vector3::Zero)

Move the center of the submesh to the given coordinate.

This will move all the vertices.

Parameters

in	<i>_center</i>	Location of the mesh center.
----	----------------	------------------------------

10.238.4.9 void gazebo::common::SubMesh::CopyNormals (const std::vector< math::Vector3 > & *_norms*)

Copy normals from a vector.

Parameters

in	<i>_norms</i>	to copy from
----	---------------	--------------

10.238.4.10 void gazebo::common::SubMesh::CopyVertices (const std::vector< math::Vector3 > & *_verts*)

Copy vertices from a vector.

Parameters

in	<code>_verts</code>	the vertices to copy from
----	---------------------	---------------------------

10.238.4.11 `void gazebo::common::SubMesh::FillArrays (float ** _vertArr, int ** _indArr) const`

Put all the data into flat arrays.

Parameters

in	<code>_verArr</code>	
in	<code>_indArr</code>	

10.238.4.12 `void gazebo::common::SubMesh::GenSphericalTexCoord (const math::Vector3 & _center)`

Generate texture coordinates using spherical projection from center.

Parameters

in	<code>_center</code>	
----	----------------------	--

10.238.4.13 `unsigned int gazebo::common::SubMesh::GetIndex (unsigned int _i) const`

Get an index.

Parameters

in	<code>_i</code>	
----	-----------------	--

10.238.4.14 `unsigned int gazebo::common::SubMesh::GetIndexCount () const`

Return the number of indicies.

10.238.4.15 `unsigned int gazebo::common::SubMesh::GetMaterialIndex () const`

Get the material index.

10.238.4.16 `math::Vector3 gazebo::common::SubMesh::GetMax () const`

Get the maximum X, Y, Z values.

Returns

10.238.4.17 `unsigned int gazebo::common::SubMesh::GetMaxIndex () const`

Get the highest index value.

10.238.4.18 `math::Vector3 gazebo::common::SubMesh::GetMin () const`

Get the minimum X, Y, Z values.

Returns

10.238.4.19 `std::string gazebo::common::SubMesh::GetName () const`

Get the name of this mesh.

Returns

the name

10.238.4.20 `NodeAssignment gazebo::common::SubMesh::GetNodeAssignment (unsigned int i) const`

Get a vertex - skeleton node assignment.

Parameters

<code>in</code>	<code><i>i</i></code>	the index of the assignment
-----------------	-----------------------	-----------------------------

10.238.4.21 `unsigned int gazebo::common::SubMesh::GetNodeAssignmentsCount () const`

Return the number of vertex - skeleton node assignments.

10.238.4.22 `math::Vector3 gazebo::common::SubMesh::GetNormal (unsigned int i) const`

Get a normal.

Parameters

<code>in</code>	<code><i>i</i></code>	the normal index
-----------------	-----------------------	------------------

Returns

the orientation of the normal, or throws an exception

10.238.4.23 `unsigned int gazebo::common::SubMesh::GetNormalCount () const`

Return the number of normals.

10.238.4.24 `PrimitiveType gazebo::common::SubMesh::GetPrimitiveType () const`

Get the primitive type.

Returns

the primitive type

10.238.4.25 `math::Vector2d gazebo::common::SubMesh::GetTexCoord (unsigned int i) const`

Get a tex coord.

Parameters

<code>in</code>	<code><i>i</i></code>	the texture index
-----------------	-----------------------	-------------------

Returns

the texture coordinates

10.238.4.26 `unsigned int gazebo::common::SubMesh::GetTexCoordCount () const`

Return the number of texture coordinates.

10.238.4.27 `math::Vector3 gazebo::common::SubMesh::GetVertex (unsigned int i) const`

Get a vertex.

Parameters

<code>in</code>	<code><i>i</i></code>	the vertex index
-----------------	-----------------------	------------------

Returns

the position or throws an exception

10.238.4.28 `unsigned int gazebo::common::SubMesh::GetVertexCount () const`

Return the number of vertices.

10.238.4.29 `unsigned int gazebo::common::SubMesh::GetVertexIndex (const math::Vector3 & v) const`

Get the index of the vertex.

Parameters

<code>in</code>	<code><i>v</i></code>	
-----------------	-----------------------	--

10.238.4.30 `bool gazebo::common::SubMesh::HasVertex (const math::Vector3 & v) const`

Return true if this submesh has the vertex.

Parameters

in	<code>_v</code>	
----	-----------------	--

10.238.4.31 `void gazebo::common::SubMesh::RecalculateNormals ()`

Recalculate all the normals.

10.238.4.32 `void gazebo::common::SubMesh::Scale (double _factor)`

Scale all vertices by `_factor`.

Parameters

in	<code>_factor</code>	Scaling factor
----	----------------------	----------------

10.238.4.33 `void gazebo::common::SubMesh::SetIndexCount (unsigned int _count)`

Resize the index array.

Parameters

in	<code>_count</code>	the new size of the array
----	---------------------	---------------------------

10.238.4.34 `void gazebo::common::SubMesh::SetMaterialIndex (unsigned int _index)`

Set the material index.

Relates to the parent mesh material list

Parameters

in	<code>_index</code>	
----	---------------------	--

10.238.4.35 `void gazebo::common::SubMesh::SetName (const std::string & _n)`

Set the name of this mesh.

Parameters

in	<code>_n</code>	the name to set
----	-----------------	-----------------

10.238.4.36 `void gazebo::common::SubMesh::SetNormal (unsigned int _i, const math::Vector3 & _n)`

Set a normal.

Parameters

in	<i>_i</i>	the normal index
in	<i>_n</i>	the normal direction

10.238.4.37 void gazebo::common::SubMesh::SetNormalCount (unsigned int *_count*)

Resize the normal array.

Parameters

in	<i>_count</i>	the new size of the array
----	---------------	---------------------------

10.238.4.38 void gazebo::common::SubMesh::SetPrimitiveType (PrimitiveType *_type*)

Set the primitive type.

Parameters

in	<i>_type</i>	the type
----	--------------	----------

10.238.4.39 void gazebo::common::SubMesh::SetScale (const math::Vector3 & *_factor*)

Scale all vertices by the *_factor* vector.

Parameters

in	<i>_factor</i>	Scaling vector
----	----------------	----------------

10.238.4.40 void gazebo::common::SubMesh::SetSubMeshCenter (math::Vector3 *_center*)

Reset mesh center to geometric center.

Parameters

in	<i>_center</i>	
----	----------------	--

10.238.4.41 void gazebo::common::SubMesh::SetTexCoord (unsigned int *_i*, const math::Vector2d & *_t*)

Set a tex coord.

Parameters

in	<i>_i</i>	
in	<i>_t</i>	

10.238.4.42 void gazebo::common::SubMesh::SetTexCoordCount (unsigned int *_count*)

Resize the texture coordinate array.

Parameters

in	<i>_count</i>	
----	---------------	--

10.238.4.43 void gazebo::common::SubMesh::SetVertex (unsigned int *_i*, const math::Vector3 & *_v*)

Set a vertex.

Parameters

in	<i>_i</i>	the index
in	<i>_v</i>	the position

10.238.4.44 void gazebo::common::SubMesh::SetVertexCount (unsigned int *_count*)

Resize the vertex array.

Parameters

in	<i>_count</i>	the new size of the array
----	---------------	---------------------------

10.238.4.45 void gazebo::common::SubMesh::Translate (const math::Vector3 & *_vec*)

Move all vertices by *_vec*.

Parameters

in	<i>_vec</i>	Amount to translate vertices.
----	-------------	-------------------------------

The documentation for this class was generated from the following file:

- **Mesh.hh**

10.239 gazebo::transport::SubscribeOptions Class Reference

Options for a subscription.

```
#include <transport/transport.hh>
```

Public Member Functions

- **SubscribeOptions** ()
Constructor.
- bool **GetLatching** () const

- Are we latching?*

 - `std::string GetMsgType () const`
Get the type of the topic we're subscribed to.
 - `NodePtr GetNode () const`
Get the node we're subscribed to.
 - `std::string GetTopic () const`
Get the topic we're subscribed to.
 - `template<class M >`
`void Init (const std::string &_topic, NodePtr _node, bool _latching)`
Initialize the options.
 - `void Init (const std::string &_topic, NodePtr _node, bool _latching)`
Initialize the options.

10.239.1 Detailed Description

Options for a subscription.

10.239.2 Constructor & Destructor Documentation

10.239.2.1 `gazebo::transport::SubscribeOptions::SubscribeOptions () [inline]`

Constructor.

10.239.3 Member Function Documentation

10.239.3.1 `bool gazebo::transport::SubscribeOptions::GetLatching () const [inline]`

Are we latching?

Returns

true if we're latching the latest message, false otherwise

10.239.3.2 `std::string gazebo::transport::SubscribeOptions::GetMsgType () const [inline]`

Get the type of the topic we're subscribed to.

Returns

The type of the topic we're subscribed to

10.239.3.3 `NodePtr gazebo::transport::SubscribeOptions::GetNode () const [inline]`

Get the node we're subscribed to.

Returns

The associated node

10.239.3.4 `std::string gazebo::transport::SubscribeOptions::GetTopic () const` `[inline]`

Get the topic we're subscribed to.

Returns

The topic we're subscribed to

10.239.3.5 `template<class M> void gazebo::transport::SubscribeOptions::Init (const std::string & _topic, NodePtr _node, bool _latching)` `[inline]`

Initialize the options.

Parameters

<code>in</code>	<code>_topic</code>	Topic we're subscribing to
<code>in, out</code>	<code>_node</code>	The associated node
<code>in</code>	<code>_latching</code>	If true, latch the latest message; if false, don't latch

References `gzthrow`, and `NULL`.

Referenced by `gazebo::transport::Node::Subscribe()`.

10.239.3.6 `void gazebo::transport::SubscribeOptions::Init (const std::string & _topic, NodePtr _node, bool _latching)` `[inline]`

Initialize the options.

This version of `init` is only used when creating subscribers of raw data.

Parameters

<code>in</code>	<code>_topic</code>	Topic we're subscribing to
<code>in, out</code>	<code>_node</code>	The associated node
<code>in</code>	<code>_latching</code>	If true, latch the latest message; if false, don't latch

The documentation for this class was generated from the following file:

- **SubscribeOptions.hh**

10.240 gazebo::transport::Subscriber Class Reference

A subscriber to a topic.

```
#include <transport/transport.hh>
```

Public Member Functions

- **Subscriber** (const std::string &_topic, NodePtr _node)
Constructor.
- virtual **~Subscriber** ()

Destructor.

- unsigned int **GetCallbackId** () const
- std::string **GetTopic** () const

Get the topic name.

- void **SetCallbackId** (unsigned int *_id*)
- void **Unsubscribe** () const

Unsubscribe from the topic.

10.240.1 Detailed Description

A subscriber to a topic.

10.240.2 Constructor & Destructor Documentation

10.240.2.1 gazebo::transport::Subscriber::Subscriber (const std::string & *_topic*, NodePtr *_node*)

Constructor.

Parameters

in	<i>_topic</i>	The topic we're subscribing to
in	<i>_node</i>	The associated node

10.240.2.2 virtual gazebo::transport::Subscriber::~Subscriber () [virtual]

Destructor.

10.240.3 Member Function Documentation

10.240.3.1 unsigned int gazebo::transport::Subscriber::GetCallbackId () const

10.240.3.2 std::string gazebo::transport::Subscriber::GetTopic () const

Get the topic name.

Returns

The topic name

10.240.3.3 void gazebo::transport::Subscriber::SetCallbackId (unsigned int *_id*)

10.240.3.4 void gazebo::transport::Subscriber::Unsubscribe () const

Unsubscribe from the topic.

The documentation for this class was generated from the following file:

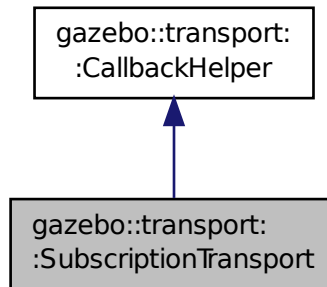
- **Subscriber.hh**

10.241 gazebo::transport::SubscriptionTransport Class Reference

transport/transport.hh

```
#include <SubscriptionTransport.hh>
```

Inheritance diagram for gazebo::transport::SubscriptionTransport:



Public Member Functions

- **SubscriptionTransport** ()
Constructor.
- virtual **~SubscriptionTransport** ()
Destructor.
- const **ConnectionPtr** & **GetConnection** () const
Get the connection we're using.
- virtual bool **HandleData** (const std::string &_newdata, boost::function< void(uint32_t)> _cb, uint32_t _id)
Output a message to a connection.
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)
Process new incoming message.
- void **Init** (**ConnectionPtr** _conn, bool _latching)
Initialize the publication link.
- virtual bool **IsLocal** () const
Is the callback local?

Additional Inherited Members

10.241.1 Detailed Description

transport/transport.hh

Handles sending data over the wire to remote subscribers

10.241.2 Constructor & Destructor Documentation

10.241.2.1 gazebo::transport::SubscriptionTransport::SubscriptionTransport ()

Constructor.

10.241.2.2 virtual gazebo::transport::SubscriptionTransport::~~SubscriptionTransport () [virtual]

Destructor.

10.241.3 Member Function Documentation

10.241.3.1 const ConnectionPtr& gazebo::transport::SubscriptionTransport::GetConnection () const

Get the connection we're using.

Returns

Pointer to the connection we're using

10.241.3.2 virtual bool gazebo::transport::SubscriptionTransport::HandleData (const std::string & *_newdata*, boost::function< void(uint32_t)> *_cb*, uint32_t *_id*) [virtual]

Output a message to a connection.

Parameters

in	<i>_newdata</i>	The message to be handled
----	-----------------	---------------------------

Returns

true if the message was handled successfully, false otherwise

Parameters

in	<i>_cb</i>	If non-null, callback to be invoked after transmission is complete.
in	<i>_id</i>	ID associated with the message data.

Implements [gazebo::transport::CallbackHelper](#) (p. 194).

10.241.3.3 virtual bool gazebo::transport::SubscriptionTransport::HandleMessage (MessagePtr *_newMsg*) [virtual]

Process new incoming message.

Parameters

in	<i>_newMsg</i>	Incoming message to be processed
----	----------------	----------------------------------

Returns

true if successfully processed; false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 194).

10.241.3.4 void **gazebo::transport::SubscriptionTransport::Init** (**ConnectionPtr** *_conn*, bool *_latching*)

Initialize the publication link.

Parameters

in	<i>_conn</i>	The connection to use
in	<i>_latching</i>	If true, latch the latest message; if false, don't latch

10.241.3.5 virtual bool **gazebo::transport::SubscriptionTransport::IsLocal** () const [virtual]

Is the callback local?

Returns

true if the callback is local, false if the callback is tied to a remote connection

Implements **gazebo::transport::CallbackHelper** (p. 194).

The documentation for this class was generated from the following file:

- **SubscriptionTransport.hh**

10.242 gazebo::physics::SurfaceParams Class Reference

SurfaceParams (p. 1090) defines various Surface contact parameters.

```
#include <physics/physics.hh>
```

Public Member Functions

- **SurfaceParams** ()
Constructor.
- virtual ~**SurfaceParams** ()
Destructor.
- virtual void **FillMsg** (msgs::Surface &_msg)
Fill in a surface message.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the contact params.
- virtual void **ProcessMsg** (const msgs::Surface &_msg)
Process a surface message.

Public Attributes

- bool **collideWithoutContact**
Allow collision checking without generating a contact joint.
- unsigned int **collideWithoutContactBitmask**
Custom collision filtering used when collideWithoutContact is true.

10.242.1 Detailed Description

SurfaceParams (p. 1090) defines various Surface contact parameters.

These parameters defines the properties of a **physics::Contact** (p. 279) constraint.

10.242.2 Constructor & Destructor Documentation

10.242.2.1 gazebo::physics::SurfaceParams::SurfaceParams ()

Constructor.

10.242.2.2 virtual gazebo::physics::SurfaceParams::~~SurfaceParams () [virtual]

Destructor.

10.242.3 Member Function Documentation

10.242.3.1 virtual void gazebo::physics::SurfaceParams::FillMsg (msgs::Surface & *msg*) [virtual]

Fill in a surface message.

Parameters

in	<i>_msg</i>	Message to fill with this object's values.
----	-------------	--

10.242.3.2 virtual void gazebo::physics::SurfaceParams::Load (sdf::ElementPtr *sdf*) [virtual]

Load the contact params.

Parameters

in	<i>_sdf</i>	SDF values to load from.
----	-------------	--------------------------

10.242.3.3 virtual void gazebo::physics::SurfaceParams::ProcessMsg (const msgs::Surface & *msg*) [virtual]

Process a surface message.

Parameters

in	<i>_msg</i>	Message to read values from.
----	-------------	------------------------------

10.242.4 Member Data Documentation

10.242.4.1 bool gazebo::physics::SurfaceParams::collideWithoutContact

Allow collision checking without generating a contact joint.

10.242.4.2 unsigned int gazebo::physics::SurfaceParams::collideWithoutContactBitmask

Custom collision filtering used when collideWithoutContact is true.

The documentation for this class was generated from the following file:

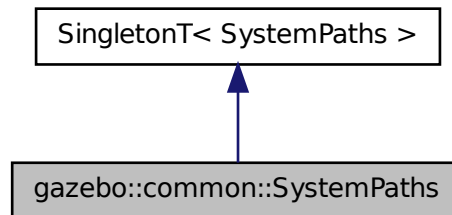
- **SurfaceParams.hh**

10.243 gazebo::common::SystemPaths Class Reference

Functions to handle getting system paths, keeps track of:

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::common::SystemPaths:



Public Member Functions

- void **AddGazeboPaths** (const std::string &_path)
Add colon delimited paths to Gazebo install.
- void **AddModelPaths** (const std::string &_path)
Add colon delimited paths to modelPaths.
- void **AddOgrePaths** (const std::string &_path)
Add colon delimited paths to ogre install.
- void **AddPluginPaths** (const std::string &_path)
Add colon delimited paths to plugins.
- void **AddSearchPathSuffix** (const std::string &_suffix)
add _suffix to the list of path search suffixes
- void **ClearGazeboPaths** ()

- clear out SystemPaths::gazeboPaths*
- void **ClearModelPaths** ()
 - clear out SystemPaths::modelPaths*
- void **ClearOgrePaths** ()
 - clear out SystemPaths::ogrePaths*
- void **ClearPluginPaths** ()
 - clear out SystemPaths::pluginPaths*
- std::string **FindFile** (const std::string &_filename, bool _searchLocalPath=true)
 - Find a file in the gazebo paths.*
- std::string **FindFileURI** (const std::string &_uri)
 - Find a file or path using a URI.*
- std::string **GetDefaultTestPath** ()
 - Returns the default temporary test path.*
- const std::list< std::string > & **GetGazeboPaths** ()
 - Get the gazebo install paths.*
- std::string **GetLogPath** () const
 - Get the log path.*
- const std::list< std::string > & **GetModelPaths** ()
 - Get the model paths.*
- const std::list< std::string > & **GetOgrePaths** ()
 - Get the ogre install paths.*
- const std::list< std::string > & **GetPluginPaths** ()
 - Get the plugin paths.*
- std::string **GetTmpInstancePath** ()
 - Returns a unique temporary file for this instance of SystemPath.*
- std::string **GetTmpPath** ()
 - Returns the default path suitable for temporary files.*
- std::string **GetWorldPathExtension** ()
 - Returns the world path extension.*

Public Attributes

- bool **gazeboPathsFromEnv**
 - if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 1096)*
- bool **modelPathsFromEnv**
 - if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 1096)*
- bool **ogrePathsFromEnv**
 - if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 1096)*
- bool **pluginPathsFromEnv**
 - if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 1096)*

Additional Inherited Members

10.243.1 Detailed Description

Functions to handle getting system paths, keeps track of:

- SystemPaths::gazeboPaths - media paths containing worlds, models, sdf descriptions, material scripts, textures.
- SystemPaths::ogrePaths - ogre library paths. Should point to **Ogre** (p. 137) RenderSystem_GL.so et. al.
- SystemPaths::pluginPaths - plugin library paths for common::WorldPlugin

10.243.2 Member Function Documentation

10.243.2.1 void gazebo::common::SystemPaths::AddGazeboPaths (const std::string & *_path*)

Add colon delimited paths to Gazebo install.

Parameters

in	<i>_path</i>	the directory to add
----	--------------	----------------------

10.243.2.2 void gazebo::common::SystemPaths::AddModelPaths (const std::string & *_path*)

Add colon delimited paths to modelPaths.

Parameters

in	<i>_path</i>	the directory to add
----	--------------	----------------------

10.243.2.3 void gazebo::common::SystemPaths::AddOgrePaths (const std::string & *_path*)

Add colon delimited paths to ogre install.

Parameters

in	<i>_path</i>	the directory to add
----	--------------	----------------------

10.243.2.4 void gazebo::common::SystemPaths::AddPluginPaths (const std::string & *_path*)

Add colon delimited paths to plugins.

Parameters

in	<i>_path</i>	the directory to add
----	--------------	----------------------

10.243.2.5 void gazebo::common::SystemPaths::AddSearchPathSuffix (const std::string & *_suffix*)

add *_suffix* to the list of path search suffixes

Parameters

in	<i>_suffix</i>	The suffix to add
----	----------------	-------------------

10.243.2.6 void gazebo::common::SystemPaths::ClearGazeboPaths ()

clear out SystemPaths::gazeboPaths

10.243.2.7 void gazebo::common::SystemPaths::ClearModelPaths ()

clear out SystemPaths::modelPaths

10.243.2.8 void gazebo::common::SystemPaths::ClearOgrePaths ()

clear out SystemPaths::ogrePaths

10.243.2.9 void gazebo::common::SystemPaths::ClearPluginPaths ()

clear out SystemPaths::pluginPaths

10.243.2.10 std::string gazebo::common::SystemPaths::FindFile (const std::string & *_filename*, bool *_searchLocalPath* = true)

Find a file in the gazebo paths.

Parameters

in	<i>_filename</i>	Name of the file to find.
in	<i>_searchLocalPath</i>	True to search in the current working directory.

Returns

Returns full path name to file

10.243.2.11 std::string gazebo::common::SystemPaths::FindFileURI (const std::string & *_uri*)

Find a file or path using a URI.

Parameters

in	<i>_uri</i>	the uniform resource identifier
----	-------------	---------------------------------

Returns

Returns full path name to file

10.243.2.12 `std::string gazebo::common::SystemPaths::GetDefaultTestPath ()`

Returns the default temporary test path.

Returns

a full path name to directory. E.g.: /tmp/gazebo_test (Linux).

10.243.2.13 `const std::list<std::string>& gazebo::common::SystemPaths::GetGazeboPaths ()`

Get the gazebo install paths.

Returns

a list of paths

10.243.2.14 `std::string gazebo::common::SystemPaths::GetLogPath () const`

Get the log path.

Returns

the path

10.243.2.15 `const std::list<std::string>& gazebo::common::SystemPaths::GetModelPaths ()`

Get the model paths.

Returns

a list of paths

10.243.2.16 `const std::list<std::string>& gazebo::common::SystemPaths::GetOgrePaths ()`

Get the ogre install paths.

Returns

a list of paths

10.243.2.17 `const std::list<std::string>& gazebo::common::SystemPaths::GetPluginPaths ()`

Get the plugin paths.

Returns

a list of paths

10.243.2.18 `std::string gazebo::common::SystemPaths::GetTmpInstancePath ()`

Returns a unique temporary file for this instance of SystemPath.

Returns

a full path name to directory. E.g.: /tmp/gazebo_234123 (Linux).

10.243.2.19 `std::string gazebo::common::SystemPaths::GetTmpPath ()`

Returns the default path suitable for temporary files.

Returns

a full path name to directory. E.g.: /tmp (Linux).

10.243.2.20 `std::string gazebo::common::SystemPaths::GetWorldPathExtension ()`

Returns the world path extension.

Returns

Right now, it just returns "/worlds"

10.243.3 Member Data Documentation

10.243.3.1 `bool gazebo::common::SystemPaths::gazeboPathsFromEnv`

if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 1096)

10.243.3.2 `bool gazebo::common::SystemPaths::modelPathsFromEnv`

if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 1096)

10.243.3.3 `bool gazebo::common::SystemPaths::ogrePathsFromEnv`

if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 1096)

10.243.3.4 `bool gazebo::common::SystemPaths::pluginPathsFromEnv`

if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 1096)

The documentation for this class was generated from the following file:

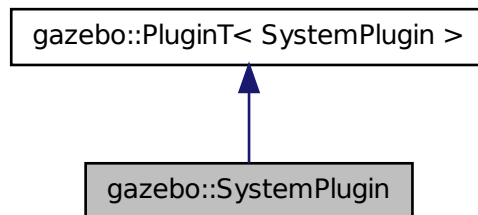
- **SystemPaths.hh**

10.244 gazebo::SystemPlugin Class Reference

A plugin loaded within the gzserver on startup.

```
#include <Plugin.hh>
```

Inheritance diagram for gazebo::SystemPlugin:



Public Member Functions

- **SystemPlugin** ()
Constructor.
- virtual **~SystemPlugin** ()
Destructor.
- virtual void **Init** ()
Initialize the plugin.
- virtual void **Load** (int _argc=0, char **_argv=NULL)=0
Load function.
- virtual void **Reset** ()
Override this method for custom plugin reset behavior.

Additional Inherited Members

10.244.1 Detailed Description

A plugin loaded within the gzserver on startup.

See [reference](#).

Todo how to make doxygen reference to the file gazebo.cc::g_plugins?

10.244.2 Constructor & Destructor Documentation

10.244.2.1 gazebo::SystemPlugin::SystemPlugin () [inline]

Constructor.

References gazebo::SYSTEM_PLUGIN.

10.244.2.2 virtual gazebo::SystemPlugin::~~SystemPlugin () [inline],[virtual]

Destructor.

10.244.3 Member Function Documentation

10.244.3.1 virtual void gazebo::SystemPlugin::Init () [inline],[virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.244.3.2 virtual void gazebo::SystemPlugin::Load (int *_argc* = 0, char ** *_argv* = NULL) [pure virtual]

Load function.

Called before Gazebo is loaded. Must not block.

Parameters

<i>_argc</i>	Number of command line arguments.
<i>_argv</i>	Array of command line arguments.

10.244.3.3 virtual void gazebo::SystemPlugin::Reset () [inline],[virtual]

Override this method for custom plugin reset behavior.

The documentation for this class was generated from the following file:

- **Plugin.hh**

10.245 gazebo::common::Time Class Reference

A **Time** (p. 1099) class, can be used to hold wall- or sim-time.

```
#include <common/common.hh>
```

Public Member Functions

- **Time** ()
Constructors.
- **Time** (const **Time** &*_time*)
Copy constructor.
- **Time** (const struct timeval &*_tv*)
Constructor.
- **Time** (const struct timespec &*_tv*)
Constructor.

- **Time** (int32_t _sec, int32_t _nsec)
Constructor.
- **Time** (double _time)
Constructor.
- virtual \sim **Time** ()
Destructor.
- double **Double** () const
Get the time as a double.
- float **Float** () const
Get the time as a float.
- bool **operator!=** (const struct timeval &_tv) const
Equal to operator.
- bool **operator!=** (const struct timespec &_tv) const
Equal to operator.
- bool **operator!=** (const **Time** &_time) const
Equal to operator.
- bool **operator!=** (double _time) const
Equal to operator.
- **Time operator*** (const struct timeval &_tv) const
Multiplication operator.
- **Time operator*** (const struct timespec &_tv) const
Multiplication operator.
- **Time operator*** (const **Time** &_time) const
Multiplication operators.
- const **Time & operator*=** (const struct timeval &_tv)
Multiplication assignment operator.
- const **Time & operator*=** (const struct timespec &_tv)
Multiplication assignment operator.
- const **Time & operator*=** (const **Time** &_time)
Multiplication operators.
- **Time operator+** (const struct timeval &_tv) const
Addition operators.
- **Time operator+** (const struct timespec &_tv) const
Addition operators.
- **Time operator+** (const **Time** &_time) const
Addition operators.
- const **Time & operator+=** (const struct timeval &_tv)
Addition assignment operator.
- const **Time & operator+=** (const struct timespec &_tv)
Addition assignment operator.
- const **Time & operator+=** (const **Time** &_time)
Addition assignment operator.
- **Time operator-** (const struct timeval &_tv) const
Subtraction operator.
- **Time operator-** (const struct timespec &_tv) const
Subtraction operator.
- **Time operator-** (const **Time** &_time) const

- Subtraction operator.*

 - const **Time & operator-** (const struct timeval &_tv)
- Subtraction assignment operator.*

 - const **Time & operator-=** (const struct timespec &_tv)
- Subtraction assignment operator.*

 - const **Time & operator-=** (const **Time** &_time)
- Subtraction assignment operator.*

 - **Time operator/** (const struct timeval &_tv) const
- Division operator.*

 - **Time operator/** (const struct timespec &_tv) const
- Division operator.*

 - **Time operator/** (const **Time** &_time) const
- Division operator.*

 - const **Time & operator/=** (const struct timeval &_tv)
- Division assignment operator.*

 - const **Time & operator/=** (const struct timespec &_tv)
- Division assignment operator.*

 - const **Time & operator/=** (const **Time** &time)
- Division assignment operator.*

 - bool **operator<** (const struct timeval &_tv) const
- Less than operator.*

 - bool **operator<** (const struct timespec &_tv) const
- Less than operator.*

 - bool **operator<** (const **Time** &_time) const
- Less than operator.*

 - bool **operator<** (double _time) const
- Less than operator.*

 - bool **operator<=** (const struct timeval &_tv) const
- Less than or equal to operator.*

 - bool **operator<=** (const struct timespec &_tv) const
- Less than or equal to operator.*

 - bool **operator<=** (const **Time** &_time) const
- Less than or equal to operator.*

 - bool **operator<=** (double _time) const
- Less than or equal to operator.*

 - **Time & operator=** (const struct timeval &_tv)
- Assignment operator.*

 - **Time & operator=** (const struct timespec &_tv)
- Assignment operator.*

 - **Time & operator=** (const **Time** &_time)
- Assignment operator.*

 - bool **operator==** (const struct timeval &_tv) const
- Equal to operator.*

 - bool **operator==** (const struct timespec &_tv) const
- Equal to operator.*

 - bool **operator==** (const **Time** &_time) const
- Equal to operator.*

 - bool **operator==** (const **Time** &_time) const

- bool **operator==** (double _time) const
Equal to operator.
- bool **operator>** (const struct timeval &_tv) const
Greater than operator.
- bool **operator>** (const struct timespec &_tv) const
Greater than operator.
- bool **operator>** (const **Time** &_time) const
Greater than operator.
- bool **operator>** (double _time) const
Greater than operator.
- bool **operator>=** (const struct timeval &_tv) const
Greater than or equal operator.
- bool **operator>=** (const struct timespec &_tv) const
Greater than or equal operator.
- bool **operator>=** (const **Time** &_time) const
Greater than or equal operator.
- bool **operator>=** (double _time) const
Greater than or equal operator.
- void **Set** (int32_t _sec, int32_t _nsec)
Set to sec and nsec.
- void **Set** (double _seconds)
Set to seconds.
- void **SetToWallTime** ()
Set the time to the wall time.

Static Public Member Functions

- static const **Time** & **GetWallTime** ()
Get the wall time.
- static const std::string & **GetWallTimeAsISOString** ()
Get the wall time as an ISO string: YYYY-MM-DDTHH:MM:SS.
- static double **MicToNano** (double _ms)
Convert microseconds to nanoseconds.
- static double **MilToNano** (double _ms)
Convert milliseconds to nanoseconds.
- static **Time** **MSleep** (unsigned int _ms)
Millisecond sleep.
- static **Time** **NSleep** (unsigned int _ns)
Nano sleep.
- static double **SecToNano** (double _sec)
Convert seconds to nanoseconds.
- static **Time** **Sleep** (const **common::Time** &_time)
Sleep for the specified time.

Public Attributes

- `int32_t nsec`
Nanoseconds.
- `int32_t sec`
Seconds.

Static Public Attributes

- static const **Time Zero**
A static zero time variable set to `common::Time(0, 0)`.

Friends

- `std::ostream & operator<<` (`std::ostream &_out`, const **gazebo::common::Time** &_time)
Stream insertion operator.
- `std::istream & operator>>` (`std::istream &_in`, **gazebo::common::Time** &_time)
Stream extraction operator.

10.245.1 Detailed Description

A **Time** (p. 1099) class, can be used to hold wall- or sim-time. stored as sec and nano-sec.

10.245.2 Constructor & Destructor Documentation

10.245.2.1 gazebo::common::Time::Time ()

Constructors.

10.245.2.2 gazebo::common::Time::Time (const Time & _time)

Copy constructor.

Parameters

<code>in</code>	<code>time</code>	Time (p. 1099) to copy
-----------------	-------------------	-------------------------------

10.245.2.3 gazebo::common::Time::Time (const struct timeval & _tv)

Constructor.

Parameters

<code>in</code>	<code>_tv</code>	Time (p. 1099) to initialize to
-----------------	------------------	--

10.245.2.4 `gazebo::common::Time::Time (const struct timespec & _tv)`

Constructor.

Parameters

in	_tv	Time (p. 1099) to initialize to
----	-----	--

10.245.2.5 `gazebo::common::Time::Time (int32_t _sec, int32_t _nsec)`

Constructor.

Parameters

in	_sec	Seconds
in	_nsec	Nanoseconds

10.245.2.6 `gazebo::common::Time::Time (double _time)`

Constructor.

Parameters

in	_time	Time (p. 1099) in double format sec.nsec
----	-------	---

10.245.2.7 `virtual gazebo::common::Time::~~Time () [virtual]`

Destructor.

10.245.3 Member Function Documentation

10.245.3.1 `double gazebo::common::Time::Double () const`

Get the time as a double.

Returns

Time (p. 1099) as a double in seconds

10.245.3.2 `float gazebo::common::Time::Float () const`

Get the time as a float.

Returns

Time (p. 1099) as a float in seconds

10.245.3.3 `static const Time& gazebo::common::Time::GetWallTime () [static]`

Get the wall time.

Returns

the current time

10.245.3.4 `static const std::string& gazebo::common::Time::GetWallTimeAsISOString () [static]`

Get the wall time as an ISO string: YYYY-MM-DDTHH:MM:SS.

Returns

The current wall time as an ISO string.

10.245.3.5 `static double gazebo::common::Time::MicToNano (double _ms) [inline],[static]`

Convert microseconds to nanoseconds.

Parameters

<i>_ms</i>	microseconds
------------	--------------

Returns

nanoseconds

10.245.3.6 `static double gazebo::common::Time::MilToNano (double _ms) [inline],[static]`

Convert milliseconds to nanoseconds.

Parameters

<i>in</i>	<i>_ms</i>	milliseconds
-----------	------------	--------------

Returns

nanoseconds

10.245.3.7 `static Time gazebo::common::Time::MSleep (unsigned int _ms) [static]`

Millisecond sleep.

Parameters

<i>in</i>	<i>_ms</i>	milliseconds
-----------	------------	--------------

Returns

Time (p. 1099) actually slept

10.245.3.8 `static Time gazebo::common::Time::NSleep (unsigned int _ns) [static]`

Nano sleep.

Parameters

<code>in</code>	<code>_ns</code>	nanoseconds
-----------------	------------------	-------------

Returns

Time (p. 1099) actually slept

10.245.3.9 `bool gazebo::common::Time::operator!= (const struct timeval & _tv) const`

Equal to operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare to
-----------------	------------------	------------------------

Returns

true if values are the same, false otherwise

10.245.3.10 `bool gazebo::common::Time::operator!= (const struct timespec & _tv) const`

Equal to operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare to
-----------------	------------------	------------------------

Returns

true if values are the same, false otherwise

10.245.3.11 `bool gazebo::common::Time::operator!= (const Time & _time) const`

Equal to operator.

Parameters

<code>in</code>	<code>_time</code>	the time to compare to
-----------------	--------------------	------------------------

Returns

true if values are the same, false otherwise

10.245.3.12 `bool gazebo::common::Time::operator!=(double _time) const`

Equal to operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare to
-----------------	---------------------------	------------------------

Returns

true if values are the same, false otherwise

10.245.3.13 `Time gazebo::common::Time::operator* (const struct timeval & _tv) const`

Multiplication operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	The scaling duration
-----------------	-------------------------	----------------------

Returns

Time (p. 1099) instance

10.245.3.14 `Time gazebo::common::Time::operator* (const struct timespec & _tv) const`

Multiplication operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the scaling duration
-----------------	-------------------------	----------------------

Returns

Time (p. 1099) instance

10.245.3.15 `Time gazebo::common::Time::operator* (const Time & _time) const`

Multiplication operators.

Parameters

<code>in</code>	<code><i>_time</i></code>	the scaling factor
-----------------	---------------------------	--------------------

Returns

a scaled **Time** (p. 1099) instance

10.245.3.16 `const Time& gazebo::common::Time::operator*=(const struct timeval & _tv)`

Multiplication assignment operator.

Parameters

<code>in</code>	<code>_tv</code>	the scaling duration
-----------------	------------------	----------------------

Returns

a reference to this instance

10.245.3.17 `const Time& gazebo::common::Time::operator*=(const struct timespec & _tv)`

Multiplication assignment operator.

Parameters

<code>in</code>	<code>_tv</code>	the scaling duration
-----------------	------------------	----------------------

Returns

a reference to this instance

10.245.3.18 `const Time& gazebo::common::Time::operator*=(const Time & _time)`

Multiplication operators.

Parameters

<code>in</code>	<code>_time</code>	scale factor
-----------------	--------------------	--------------

Returns

a scaled **Time** (p. 1099) instance

10.245.3.19 `Time gazebo::common::Time::operator+(const struct timeval & _tv) const`

Addition operators.

Parameters

<code>in</code>	<code>_tv</code>	the time to add
-----------------	------------------	-----------------

Returns

a **Time** (p. 1099) instance

10.245.3.20 Time gazebo::common::Time::operator+ (const struct timespec & *_tv*) const

Addition operators.

Parameters

<i>in</i>	<i>_tv</i>	the time to add
-----------	------------	-----------------

Returns

a **Time** (p. 1099) instance

10.245.3.21 Time gazebo::common::Time::operator+ (const Time & *_time*) const

Addition operators.

Parameters

<i>in</i>	<i>_time</i>	The time to add
-----------	--------------	-----------------

Returns

a **Time** (p. 1099) instance

10.245.3.22 const Time& gazebo::common::Time::operator+= (const struct timeval & *_tv*)

Addition assignment operator.

Parameters

<i>in</i>	<i>_tv</i>	the time to add
-----------	------------	-----------------

Returns

a reference to this instance

10.245.3.23 const Time& gazebo::common::Time::operator+= (const struct timespec & *_tv*)

Addition assignment operator.

Parameters

<i>in</i>	<i>_tv</i>	the time to add
-----------	------------	-----------------

Returns

a reference to this instance

10.245.3.24 `const Time& gazebo::common::Time::operator+=(const Time & _time)`

Addition assignemtn operator.

Parameters

<code>in</code>	<code>_time</code>	The time to add
-----------------	--------------------	-----------------

Returns

a **Time** (p. 1099) instance

10.245.3.25 `Time gazebo::common::Time::operator- (const struct timeval & _tv) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_tv</code>	The time to subtract
-----------------	------------------	----------------------

Returns

a **Time** (p. 1099) instance

10.245.3.26 `Time gazebo::common::Time::operator- (const struct timespec & _tv) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_tv</code>	The time to subtract
-----------------	------------------	----------------------

Returns

a **Time** (p. 1099) instance

10.245.3.27 `Time gazebo::common::Time::operator- (const Time & _time) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_time</code>	The time to subtract
-----------------	--------------------	----------------------

Returns

a **Time** (p. 1099) instance

10.245.3.28 `const Time& gazebo::common::Time::operator-= (const struct timeval & _tv)`

Subtraction assignment operator.

Parameters

<code>in</code>	<code>_tv</code>	The time to subtract
-----------------	------------------	----------------------

Returns

a **Time** (p. 1099) instance

10.245.3.29 `const Time& gazebo::common::Time::operator-= (const struct timespec & _tv)`

Subtraction assignment operator.

Parameters

<code>in</code>	<code>_tv</code>	The time to subtract
-----------------	------------------	----------------------

Returns

a **Time** (p. 1099) instance

10.245.3.30 `const Time& gazebo::common::Time::operator-= (const Time & _time)`

Subtraction assignment operator.

Parameters

<code>in</code>	<code>_time</code>	The time to subtract
-----------------	--------------------	----------------------

Returns

a reference to this instance

10.245.3.31 `Time gazebo::common::Time::operator/ (const struct timeval & _tv) const`

Division operator.

Parameters

<code>in</code>	<code>_tv</code>	a timeval divisor
-----------------	------------------	-------------------

Returns

a **Time** (p. 1099) instance

10.245.3.32 **Time** gazebo::common::Time::operator/ (const struct timespec & *_tv*) const

Division operator.

Parameters

<i>in</i>	<i>_tv</i>	a timespec divisor
-----------	------------	--------------------

Returns

a **Time** (p. 1099) instance

10.245.3.33 **Time** gazebo::common::Time::operator/ (const Time & *_time*) const

Division operator.

Parameters

<i>in</i>	<i>_time</i>	the divisor
-----------	--------------	-------------

Returns

a **Time** (p. 1099) instance

10.245.3.34 **const Time&** gazebo::common::Time::operator/= (const struct timeval & *_tv*)

Division assignment operator.

Parameters

<i>in</i>	<i>_tv</i>	a divisor
-----------	------------	-----------

Returns

a **Time** (p. 1099) instance

10.245.3.35 **const Time&** gazebo::common::Time::operator/= (const struct timespec & *_tv*)

Division assignment operator.

Parameters

<i>in</i>	<i>_tv</i>	a divisor
-----------	------------	-----------

Returns

a **Time** (p. 1099) instance

10.245.3.36 `const Time& gazebo::common::Time::operator/= (const Time & time)`

Division assignment operator.

Parameters

<code>in</code>	<code><i>time</i></code>	the divisor
-----------------	--------------------------	-------------

Returns

a **Time** (p. 1099) instance

10.245.3.37 `bool gazebo::common::Time::operator< (const struct timeval & _tv) const`

Less than operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if tv is shorter than this, false otherwise

10.245.3.38 `bool gazebo::common::Time::operator< (const struct timespec & _tv) const`

Less than operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if tv is shorter than this, false otherwise

10.245.3.39 `bool gazebo::common::Time::operator< (const Time & _time) const`

Less than operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is shorter than this, false otherwise

10.245.3.40 `bool gazebo::common::Time::operator< (double _time) const`

Less than operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is shorter than this, false otherwise

10.245.3.41 `bool gazebo::common::Time::operator<= (const struct timeval & _tv) const`

Less than or equal to operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if tv is shorter than or equal to this, false otherwise

10.245.3.42 `bool gazebo::common::Time::operator<= (const struct timespec & _tv) const`

Less than or equal to operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if tv is shorter than or equal to this, false otherwise

10.245.3.43 `bool gazebo::common::Time::operator<= (const Time & _time) const`

Less than or equal to operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is shorter than or equal to this, false otherwise

10.245.3.44 `bool gazebo::common::Time::operator<=(double _time) const`

Less than or equal to operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is shorter than or equal to this, false otherwise

10.245.3.45 `Time& gazebo::common::Time::operator=(const struct timeval & _tv)`

Assignment operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the new time
-----------------	-------------------------	--------------

Returns

a reference to this instance

10.245.3.46 `Time& gazebo::common::Time::operator=(const struct timespec & _tv)`

Assignment operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the new time
-----------------	-------------------------	--------------

Returns

a reference to this instance

10.245.3.47 `Time& gazebo::common::Time::operator=(const Time & _time)`

Assignment operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the new time
-----------------	---------------------------	--------------

Returns

a reference to this instance

10.245.3.48 `bool gazebo::common::Time::operator==(const struct timeval & _tv) const`

Equal to operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare to
-----------------	------------------	------------------------

Returns

true if values are the same, false otherwise

10.245.3.49 `bool gazebo::common::Time::operator==(const struct timespec & _tv) const`

Equal to operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare to
-----------------	------------------	------------------------

Returns

true if values are the same, false otherwise

10.245.3.50 `bool gazebo::common::Time::operator==(const Time & _time) const`

Equal to operator.

Parameters

<code>in</code>	<code>_time</code>	the time to compare to
-----------------	--------------------	------------------------

Returns

true if values are the same, false otherwise

10.245.3.51 `bool gazebo::common::Time::operator==(double _time) const`

Equal to operator.

Parameters

<code>in</code>	<code>_time</code>	the time to compare to
-----------------	--------------------	------------------------

Returns

true if values are the same, false otherwise

10.245.3.52 `bool gazebo::common::Time::operator> (const struct timeval & _tv) const`

Greater than operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare with
-----------------	------------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.245.3.53 `bool gazebo::common::Time::operator> (const struct timespec & _tv) const`

Greater than operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare with
-----------------	------------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.245.3.54 `bool gazebo::common::Time::operator> (const Time & _time) const`

Greater than operator.

Parameters

<code>in</code>	<code>_time</code>	the time to compare with
-----------------	--------------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.245.3.55 `bool gazebo::common::Time::operator> (double _time) const`

Greater than operator.

Parameters

<code>in</code>	<code>_time</code>	the time to compare with
-----------------	--------------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.245.3.56 `bool gazebo::common::Time::operator>= (const struct timeval & _tv) const`

Greater than or equal operator.

Parameters

in	_tv	the time to compare with
----	-----	--------------------------

Returns

true if tv is greater than or equal to this, false otherwise

10.245.3.57 `bool gazebo::common::Time::operator>= (const struct timespec & _tv) const`

Greater than or equal operator.

Parameters

in	_tv	the time to compare with
----	-----	--------------------------

Returns

true if tv is greater than or equal to this, false otherwise

10.245.3.58 `bool gazebo::common::Time::operator>= (const Time & _time) const`

Greater than or equal operator.

Parameters

in	_time	the time to compare with
----	-------	--------------------------

Returns

true if time is greater than or equal to this, false otherwise

10.245.3.59 `bool gazebo::common::Time::operator>= (double _time) const`

Greater than or equal operator.

Parameters

in	_time	the time to compare with
----	-------	--------------------------

Returns

true if time is greater than or equal to this, false otherwise

10.245.3.60 `static double gazebo::common::Time::SecToNano (double _sec)` `[inline],[static]`

Convert seconds to nanoseconds.

Parameters

<code>in</code>	<code>_sec</code>	duration in seconds
-----------------	-------------------	---------------------

Returns

nanoseconds

10.245.3.61 `void gazebo::common::Time::Set (int32_t _sec, int32_t _nsec)`

Set to sec and nsec.

Parameters

<code>in</code>	<code>_sec</code>	Seconds
<code>in</code>	<code>_nsec</code>	Nanoseconds

10.245.3.62 `void gazebo::common::Time::Set (double _seconds)`

Set to seconds.

Parameters

<code>in</code>	<code>_seconds</code>	Number of seconds
-----------------	-----------------------	-------------------

10.245.3.63 `void gazebo::common::Time::SetToWallTime ()`

Set the time to the wall time.

10.245.3.64 `static Time gazebo::common::Time::Sleep (const common::Time & _time)` `[static]`

Sleep for the specified time.

Parameters

<code>in</code>	<code>_time</code>	Sleep time
-----------------	--------------------	------------

Returns

Time (p. 1099) actually slept

10.245.4 Friends And Related Function Documentation

10.245.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::common::Time & _time)` [*friend*]

Stream insertion operator.

Parameters

<i>in</i>	<i>_out</i>	the output stream
<i>in</i>	<i>_time</i>	time to write to the stream

Returns

the output stream

10.245.4.2 `std::istream& operator>> (std::istream & _in, gazebo::common::Time & _time)` [*friend*]

Stream extraction operator.

Parameters

<i>in</i>	<i>_in</i>	the input stream
<i>in</i>	<i>_time</i>	time to read from to the stream

Returns

the input stream

10.245.5 Member Data Documentation

10.245.5.1 `int32_t gazebo::common::Time::nsec`

Nanoseconds.

10.245.5.2 `int32_t gazebo::common::Time::sec`

Seconds.

10.245.5.3 `const Time gazebo::common::Time::Zero` [*static*]

A static zero time variable set to `common::Time(0, 0)`.

The documentation for this class was generated from the following file:

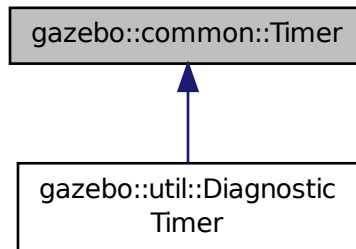
- **Time.hh**

10.246 gazebo::common::Timer Class Reference

A timer class, used to time things in real world walltime.

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::common::Timer:



Public Member Functions

- **Timer** ()
Constructor.
- virtual **~Timer** ()
Destructor.
- **Time GetElapsed** () const
Get the elapsed time.
- bool **GetRunning** () const
Returns true if the timer is running.
- virtual void **Start** ()
Start the timer.
- virtual void **Stop** ()
Stop the timer.

Friends

- std::ostream & **operator**<< (std::ostream &out, const **gazebo::common::Timer** &t)
Stream operator friendly.

10.246.1 Detailed Description

A timer class, used to time things in real world walltime.

10.246.2 Constructor & Destructor Documentation

10.246.2.1 gazebo::common::Timer::Timer ()

Constructor.

10.246.2.2 `virtual gazebo::common::Timer::~~Timer () [virtual]`

Destructor.

10.246.3 Member Function Documentation

10.246.3.1 `Time gazebo::common::Timer::GetElapsed () const`

Get the elapsed time.

Returns

The time

10.246.3.2 `bool gazebo::common::Timer::GetRunning () const`

Returns true if the timer is running.

Returns

True if the timer has been started and not stopped.

10.246.3.3 `virtual void gazebo::common::Timer::Start () [virtual]`

Start the timer.

Reimplemented in `gazebo::util::DiagnosticTimer` (p. 395).

10.246.3.4 `virtual void gazebo::common::Timer::Stop () [virtual]`

Stop the timer.

Reimplemented in `gazebo::util::DiagnosticTimer` (p. 395).

10.246.4 Friends And Related Function Documentation

10.246.4.1 `std::ostream& operator<< (std::ostream & out, const gazebo::common::Timer & t) [friend]`

Stream operator friendly.

The documentation for this class was generated from the following file:

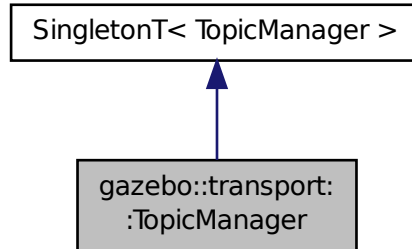
- `Timer.hh`

10.247 gazebo::transport::TopicManager Class Reference

Manages topics and their subscriptions.

```
#include <transport/transport.hh>
```

Inheritance diagram for gazebo::transport::TopicManager:



Public Types

- typedef std::map< std::string, std::list< **NodePtr** > > **SubNodeMap**
A map of string->list of **Node** (p. 731) pointers.

Public Member Functions

- void **AddNode** (**NodePtr** _node)
Add a node to the manager.
- void **AddNodeToProcess** (**NodePtr** _ptr)
Add a node to the list of nodes that requires processing.
- template<typename M > **PublisherPtr Advertise** (const std::string &_topic, unsigned int _queueLimit, double _hzRate)
Advertise on a topic.
- void **ClearBuffers** ()
Clear all buffers.
- void **ConnectPubToSub** (const std::string &_topic, const **SubscriptionTransportPtr** _sublink)
Connection (p. 264) a local **Publisher** (p. 820) to a remote **Subscriber** (p. 1086).
- void **ConnectSubscribers** (const std::string &_topic)
Connect all subscribers on a topic to known publishers.
- void **ConnectSubToPub** (const msgs::Publish &_pub)
Connect a local **Subscriber** (p. 1086) to a remote **Publisher** (p. 820).
- void **DisconnectPubFromSub** (const std::string &_topic, const std::string &_host, unsigned int _port)
Disconnect a local publisher from a remote subscriber.
- void **DisconnectSubFromPub** (const std::string &_topic, const std::string &_host, unsigned int _port)
Disconnect all local subscribers from a remote publisher.
- **PublicationPtr FindPublication** (const std::string &_topic)
Find a publication object by topic.
- void **Fini** ()

- Finalize the manager.*

 - void **GetTopicNamespaces** (std::list< std::string > &_namespaces)
Get all the topic namespaces.
- void **Init** ()
Initialize the manager.
- bool **IsAdvertised** (const std::string &_topic)
Has the topic been advertised?
- void **PauseIncoming** (bool _pause)
Pause or unpaue processing of incoming messages.
- void **ProcessNodes** (bool _onlyOut=false)
Process all nodes under management.
- void **Publish** (const std::string &_topic, **MessagePtr** _message, boost::function< void(uint32_t)> _cb, uint32_t _id)
Send a message.
- void **RegisterTopicNamespace** (const std::string &_name)
Register a new topic namespace.
- void **RemoveNode** (unsigned int _id)
Remove a node by its id.
- **SubscriberPtr** **Subscribe** (const **SubscribeOptions** &_options)
Subscribe to a topic.
- void **Unadvertise** (const std::string &_topic)
Unadvertise a topic.
- void **Unadvertise** (**PublisherPtr** _pub)
Unadvertise a publisher.
- void **Unsubscribe** (const std::string &_topic, const **NodePtr** &_sub)
Unsubscribe from a topic.
- **PublicationPtr** **UpdatePublications** (const std::string &_topic, const std::string &_msgType)
Update our list of advertised topics.

Additional Inherited Members

10.247.1 Detailed Description

Manages topics and their subscriptions.

10.247.2 Member Typedef Documentation

10.247.2.1 typedef std::map<std::string, std::list<NodePtr> > gazebo::transport::TopicManager::SubNodeMap

A map of string->list of **Node** (p. 731) pointers.

10.247.3 Member Function Documentation

10.247.3.1 void gazebo::transport::TopicManager::AddNode (NodePtr _node)

Add a node to the manager.

Parameters

in, out	<i>_node</i>	The node to be added
---------	--------------	----------------------

10.247.3.2 void gazebo::transport::TopicManager::AddNodeToProcess (**NodePtr** *_ptr*)

Add a node to the list of nodes that requires processing.

Parameters

in	<i>_ptr</i>	Node (p. 731) to process.
----	-------------	----------------------------------

10.247.3.3 template<typename M > **PublisherPtr** gazebo::transport::TopicManager::Advertise (const std::string & *_topic*, unsigned int *_queueLimit*, double *_hzRate*) [inline]

Advertise on a topic.

Parameters

in	<i>_topic</i>	The name of the topic
in	<i>_queueLimit</i>	The maximum number of outgoing messages to queue
in	<i>_hz</i>	Update rate for the publisher. Units are 1.0/seconds.

Returns

Pointer to the newly created **Publisher** (p. 820)

References GZ_ASSERT, gzthrow, SingletonT< T >::Instance(), and NULL.

10.247.3.4 void gazebo::transport::TopicManager::ClearBuffers ()

Clear all buffers.

10.247.3.5 void gazebo::transport::TopicManager::ConnectPubToSub (const std::string & *_topic*, const **SubscriptionTransportPtr** *_sublink*)

Connection (p. 264) a local **Publisher** (p. 820) to a remote **Subscriber** (p. 1086).

Parameters

in	<i>_topic</i>	The topic to use
in	<i>_sublink</i>	The subscription transport object to use

10.247.3.6 void gazebo::transport::TopicManager::ConnectSubscribers (const std::string & *_topic*)

Connect all subscribers on a topic to known publishers.

Parameters

<i>in</i>	<i>_topic</i>	The topic to be connected
-----------	---------------	---------------------------

10.247.3.7 `void gazebo::transport::TopicManager::ConnectSubToPub (const msgs::Publish & _pub)`

Connect a local **Subscriber** (p. 1086) to a remote **Publisher** (p. 820).

Parameters

<i>in</i>	<i>_pub</i>	The publish object to use
-----------	-------------	---------------------------

10.247.3.8 `void gazebo::transport::TopicManager::DisconnectPubFromSub (const std::string & _topic, const std::string & _host, unsigned int _port)`

Disconnect a local publisher from a remote subscriber.

Parameters

<i>in</i>	<i>_topic</i>	The topic to be disconnected
<i>in</i>	<i>_host</i>	The host to be disconnected
<i>in</i>	<i>_port</i>	The port to be disconnected

10.247.3.9 `void gazebo::transport::TopicManager::DisconnectSubFromPub (const std::string & _topic, const std::string & _host, unsigned int _port)`

Disconnect all local subscribers from a remote publisher.

Parameters

<i>in</i>	<i>_topic</i>	The topic to be disconnected
<i>in</i>	<i>_host</i>	The host to be disconnected
<i>in</i>	<i>_port</i>	The port to be disconnected

10.247.3.10 `PublicationPtr gazebo::transport::TopicManager::FindPublication (const std::string & _topic)`

Find a publication object by topic.

Parameters

<i>in</i>	<i>_topic</i>	The topic to search for
-----------	---------------	-------------------------

Returns

Pointer to the publication object, if found (can be null)

10.247.3.11 `void gazebo::transport::TopicManager::Fini ()`

Finalize the manager.

10.247.3.12 `void gazebo::transport::TopicManager::GetTopicNamespaces (std::list< std::string > & _namespaces)`

Get all the topic namespaces.

Parameters

out	<code>_namespaces</code>	The list of namespaces will be written here
-----	--------------------------	---

10.247.3.13 `void gazebo::transport::TopicManager::Init ()`

Initialize the manager.

10.247.3.14 `bool gazebo::transport::TopicManager::IsAdvertised (const std::string & _topic)`

Has the topic been advertised?

Parameters

in	<code>_topic</code>	The name of the topic to check
----	---------------------	--------------------------------

Returns

true if the topic has been advertised, false otherwise

10.247.3.15 `void gazebo::transport::TopicManager::PauseIncoming (bool _pause)`

Pause or unpaue processing of incoming messages.

Parameters

in	<code>_pause</code>	If true pause processing; otherwise unpaue
----	---------------------	--

10.247.3.16 `void gazebo::transport::TopicManager::ProcessNodes (bool _onlyOut = false)`

Process all nodes under management.

Parameters

in	<code>_onlyOut</code>	True means only outbound messages on nodes will be sent. False means nodes process both outbound and inbound messages
----	-----------------------	---

10.247.3.17 `void gazebo::transport::TopicManager::Publish (const std::string & _topic, MessagePtr _message, boost::function< void(uint32_t)> _cb, uint32_t _id)`

Send a message.

Use a **Publisher** (p. 820) instead of calling this function directly.

Parameters

in	<i>_topic</i>	Name of the topic
in	<i>_message</i>	The message to send.
in	<i>_cb</i>	Callback, used when the publish is completed.
in	<i>_id</i>	ID associated with the message.

10.247.3.18 void gazebo::transport::TopicManager::RegisterTopicNamespace (const std::string & *_name*)

Register a new topic namespace.

Parameters

in	<i>_name</i>	The name of the new namespace
----	--------------	-------------------------------

10.247.3.19 void gazebo::transport::TopicManager::RemoveNode (unsigned int *_id*)

Remove a node by its id.

Parameters

in	<i>_id</i>	The ID of the node to be removed
----	------------	----------------------------------

10.247.3.20 SubscriberPtr gazebo::transport::TopicManager::Subscribe (const SubscribeOptions & *_options*)

Subscribe to a topic.

Parameters

in	<i>_options</i>	The options to use for the subscription
----	-----------------	---

Returns

Pointer to the newly created subscriber

10.247.3.21 void gazebo::transport::TopicManager::Unadvertise (const std::string & *_topic*)

Unadvertise a topic.

Parameters

in	<i>_topic</i>	The topic to be unadvertised
----	---------------	------------------------------

10.247.3.22 void gazebo::transport::TopicManager::Unadvertise (PublisherPtr *_pub*)

Unadvertise a publisher.

Parameters

in	<code>_pub</code>	Publisher (p. 820) to unadvertise.
----	-------------------	---

10.247.3.23 `void gazebo::transport::TopicManager::Unsubscribe (const std::string & .topic, const NodePtr & .sub)`

Unsubscribe from a topic.

Use a **Subscriber** (p. 1086) rather than calling this function directly

Parameters

in	<code>_topic</code>	The topic to unsubscribe from
in	<code>_sub</code>	The node to unsubscribe

10.247.3.24 `PublicationPtr gazebo::transport::TopicManager::UpdatePublications (const std::string & .topic, const std::string & .msgType)`

Update our list of advertised topics.

Parameters

in	<code>_topic</code>	The topic to be updated
in	<code>_msgType</code>	The type of the topic to be updated

Returns

True if the provided params define a new publisher, false otherwise

The documentation for this class was generated from the following file:

- **TopicManager.hh**

10.248 gazebo::physics::TrajectoryInfo Class Reference

Information about a trajectory for an **Actor** (p. 139).

```
#include <Actor.hh>
```

Public Member Functions

- **TrajectoryInfo ()**
Constructor.

Public Attributes

- double **duration**
Duration of the trajectory.
- double **endTime**

End time of the trajectory.

- unsigned int **id**

ID of the trajectory.

- double **startTime**

Start time of the trajectory.

- bool **translated**

True if the trajectory is translated.

- std::string **type**

Type of trajectory.

10.248.1 Detailed Description

Information about a trajectory for an **Actor** (p. 139).

10.248.2 Constructor & Destructor Documentation

10.248.2.1 gazebo::physics::TrajectoryInfo::TrajectoryInfo ()

Constructor.

10.248.3 Member Data Documentation

10.248.3.1 double gazebo::physics::TrajectoryInfo::duration

Duration of the trajectory.

10.248.3.2 double gazebo::physics::TrajectoryInfo::endTime

End time of the trajectory.

10.248.3.3 unsigned int gazebo::physics::TrajectoryInfo::id

ID of the trajectory.

10.248.3.4 double gazebo::physics::TrajectoryInfo::startTime

Start time of the trajectory.

10.248.3.5 bool gazebo::physics::TrajectoryInfo::translated

True if the trajectory is translated.

10.248.3.6 `std::string gazebo::physics::TrajectoryInfo::type`

Type of trajectory.

The documentation for this class was generated from the following file:

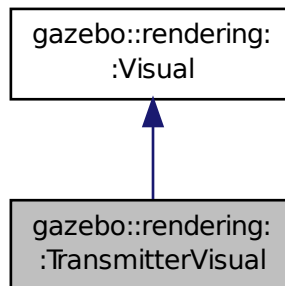
- **Actor.hh**

10.249 gazebo::rendering::TransmitterVisual Class Reference

Visualization for the wireless propagation data.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::TransmitterVisual:



Public Member Functions

- **TransmitterVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)
Constructor.
- virtual **~TransmitterVisual** ()
Destructor.
- virtual void **Load** ()
Documentation inherited from parent.
- virtual void **Update** ()
Function that runs on the OGRE thread to refresh the UI.

Additional Inherited Members

10.249.1 Detailed Description

Visualization for the wireless propagation data.

10.249.2 Constructor & Destructor Documentation

10.249.2.1 `gazebo::rendering::TransmitterVisual::TransmitterVisual (const std::string & _name, VisualPtr _vis, const std::string & _topicName)`

Constructor.

Parameters

<code>in</code>	<code><i>_name</i></code>	Name of the visual.
<code>in</code>	<code><i>_vis</i></code>	Pointer to the parent Visual (p. 1196).
<code>in</code>	<code><i>_topicName</i></code>	Name of the topic that has laser data.

10.249.2.2 `virtual gazebo::rendering::TransmitterVisual::~~TransmitterVisual () [virtual]`

Destructor.

10.249.3 Member Function Documentation

10.249.3.1 `virtual void gazebo::rendering::TransmitterVisual::Load () [virtual]`

Documentation inherited from parent.

Reimplemented from **gazebo::rendering::Visual** (p. 1210).

10.249.3.2 `virtual void gazebo::rendering::TransmitterVisual::Update () [virtual]`

Function that runs on the OGRE thread to refresh the UI.

Reimplemented from **gazebo::rendering::Visual** (p. 1217).

The documentation for this class was generated from the following file:

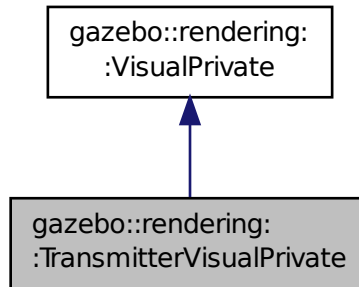
- **TransmitterVisual.hh**

10.250 gazebo::rendering::TransmitterVisualPrivate Class Reference

Private data for the Transmitter **Visual** (p. 1196) class.

```
#include <TransmitterVisualPrivate.hh>
```


Inheritance diagram for gazebo::rendering::TransmitterVisualPrivate:



Public Attributes

- `std::vector< event::ConnectionPtr > connections`
All the event connections.
- `boost::shared_ptr< msgs::PropagationGrid const > gridMsg`
The current contact message.
- `bool isFirst`
Use for allocate the visuals for the grid only the first time you receive the grid.
- `boost::mutex mutex`
Mutex to protect the contact message.
- `transport::NodePtr node`
Pointer to a node that handles communication.
- `DynamicLines * points`
Renders the points representing the signal strength.
- `bool receivedMsg`
True if we have received a message.
- `transport::SubscriberPtr signalPropagationSub`
Subscription to the propagation data.
- `std::vector< rendering::VisualPtr > vectorLink`
Store the list of visuals.

Additional Inherited Members

10.250.1 Detailed Description

Private data for the Transmitter **Visual** (p. 1196) class.

10.250.2 Member Data Documentation

10.250.2.1 `std::vector<event::ConnectionPtr>` gazebo::rendering::TransmitterVisualPrivate::connections

All the event connections.

10.250.2.2 `boost::shared_ptr<msgs::PropagationGrid const>` gazebo::rendering::TransmitterVisualPrivate::gridMsg

The current contact message.

10.250.2.3 `bool` gazebo::rendering::TransmitterVisualPrivate::isFirst

Use for allocate the visuals for the grid only the first time you receive the grid.

The next times there are just updates.

10.250.2.4 `boost::mutex` gazebo::rendering::TransmitterVisualPrivate::mutex

Mutex to protect the contact message.

10.250.2.5 `transport::NodePtr` gazebo::rendering::TransmitterVisualPrivate::node

Pointer to a node that handles communication.

10.250.2.6 `DynamicLines*` gazebo::rendering::TransmitterVisualPrivate::points

Renders the points representing the signal strength.

10.250.2.7 `bool` gazebo::rendering::TransmitterVisualPrivate::receivedMsg

True if we have received a message.

10.250.2.8 `transport::SubscriberPtr` gazebo::rendering::TransmitterVisualPrivate::signalPropagationSub

Subscription to the propagation data.

10.250.2.9 `std::vector<rendering::VisualPtr>` gazebo::rendering::TransmitterVisualPrivate::vectorLink

Store the list of visuals.

The documentation for this class was generated from the following file:

- **TransmitterVisualPrivate.hh**

10.251 gazebo::physics::UniversalJoint< T > Class Template Reference

A universal joint.

```
#include <physics/physics.hh>
```

Public Types

- enum **AxisIndex** { **AXIS_PARENT** = 0, **AXIS_CHILD** = 1 }
Map joint axes to corresponding link.

Public Member Functions

- **UniversalJoint** (**BasePtr** _parent)
Constructor.
- virtual **~UniversalJoint** ()
Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (sdf::ElementPtr _sdf)
*Load a **UniversalJoint** (p. 1135).*

Protected Member Functions

- virtual void **Init** ()
Initialize joint.

10.251.1 Detailed Description

```
template<class T>class gazebo::physics::UniversalJoint< T >
```

A universal joint.

Axis1 and axis2 are body-fixed, with axis1 attached to parent body and axis2 attached to child body.

10.251.2 Member Enumeration Documentation

10.251.2.1 `template<class T> enum gazebo::physics::UniversalJoint::AxisIndex`

Map joint axes to corresponding link.

Enumerator:

AXIS_PARENT

AXIS_CHILD

10.251.3 Constructor & Destructor Documentation

10.251.3.1 `template<class T> gazebo::physics::UniversalJoint< T >::UniversalJoint (BasePtr _parent)`
`[inline], [explicit]`

Constructor.

Parameters

in	_parent	Parent link of the univernal joint.
----	---------	-------------------------------------

10.251.3.2 `template<class T> virtual gazebo::physics::UniversalJoint< T >::~~UniversalJoint ()` `[inline],`
`[virtual]`

Destuctor.

10.251.4 Member Function Documentation

10.251.4.1 `template<class T> virtual unsigned int gazebo::physics::UniversalJoint< T >::GetAngleCount () const`
`[inline], [virtual]`

10.251.4.2 `template<class T> virtual void gazebo::physics::UniversalJoint< T >::Init ()` `[inline],`
`[protected], [virtual]`

Initialize joint.

Reimplemented in `gazebo::physics::DARTUniversalJoint` (p. 381).

10.251.4.3 `template<class T> virtual void gazebo::physics::UniversalJoint< T >::Load (sdf::ElementPtr _sdf)`
`[inline], [virtual]`

Load a `UniversalJoint` (p. 1135).

Parameters

in	_sdf	SDF values to load from.
----	------	--------------------------

Reimplemented in `gazebo::physics::SimbodyUniversalJoint` (p. 1019), and `gazebo::physics::DARTUniversalJoint` (p. 381).

The documentation for this class was generated from the following file:

- `UniversalJoint.hh`

10.252 gazebo::common::UpdateInfo Class Reference

Information for use in an update event.

```
#include <common/common.hh>
```

Public Attributes

- **common::Time realTime**

Current real time.

- **common::Time simTime**

Current simulation time.

- **std::string worldName**

Name of the world.

10.252.1 Detailed Description

Information for use in an update event.

10.252.2 Member Data Documentation

10.252.2.1 **common::Time gazebo::common::UpdateInfo::realTime**

Current real time.

10.252.2.2 **common::Time gazebo::common::UpdateInfo::simTime**

Current simulation time.

10.252.2.3 **std::string gazebo::common::UpdateInfo::worldName**

Name of the world.

The documentation for this class was generated from the following file:

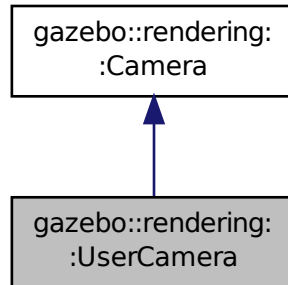
- **UpdateInfo.hh**

10.253 gazebo::rendering::UserCamera Class Reference

A camera used for user visualization of a scene.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::UserCamera:



Public Member Functions

- **UserCamera** (const std::string &_name, **ScenePtr** _scene)
Constructor.
- virtual ~**UserCamera** ()
Destructor.
- void **EnableViewController** (bool _value) const
Set whether the view controller is enabled.
- void **Fini** ()
Finalize.
- float **GetAvgFPS** () const
Get the average frames per second.
- **GUIOverlay** * **GetGUIOverlay** ()
Get the GUI overlay.
- virtual unsigned int **GetImageHeight** () const
Get the height of the image.
- virtual unsigned int **GetImageWidth** () const
Get the width of the image.
- unsigned int **GetTriangleCount** () const
Get the triangle count.
- std::string **GetViewControllerTypeString** ()
Get current view controller type.
- **VisualPtr** **GetVisual** (const **math::Vector2i** &_mousePos, std::string &_mod)
Get an entity at a pixel location using a camera.
- **VisualPtr** **GetVisual** (const **math::Vector2i** &_mousePos) const
Get a visual at a mouse position.
- void **HandleKeyPressEvent** (const std::string &_key)
Handle a key press.
- void **HandleKeyReleaseEvent** (const std::string &_key)

- Handle a key release.*

 - void **HandleMouseEvent** (const **common::MouseEvent** &_evt)
- Handle a mouse event.*

 - void **Init** ()
- Initialize.*

 - bool **IsCameraSetInWorldFile** ()

brief Show if the user camera pose has changed in the world file.
- void **Load** (sdf::ElementPtr _sdf)

Load the user camera.
- void **Load** ()

Generic load function.
- virtual bool **MoveToPosition** (const **math::Pose** &_pose, double _time)

Move the camera to a position (this is an animated motion).
- void **MoveToVisual** (**VisualPtr** _visual)

Move the camera to focus on a visual.
- void **MoveToVisual** (const std::string &_visualName)

Move the camera to focus on a visual.
- virtual void **PostRender** ()

Post render.
- void **Resize** (unsigned int _w, unsigned int _h)

Resize the camera.
- void **SetFocalPoint** (const **math::Vector3** &_pt)

Set the point the camera should orbit around.
- virtual void **SetRenderTarget** (Ogre::RenderTarget *_target)

Set to true to enable rendering.
- void **SetUseSDFPose** (bool _value)

brief Set if the user camera pose has changed in the world file.
- void **SetViewController** (const std::string &_type)

Set view controller.
- void **SetViewController** (const std::string &_type, const **math::Vector3** &_pos)

Set view controller.
- void **SetViewportDimensions** (float _x, float _y, float _w, float _h)

Set the dimensions of the viewport.
- virtual void **SetWorldPose** (const **math::Pose** &_pose)

Set the pose in the world coordinate frame.
- virtual void **Update** ()

Render the camera.

Protected Member Functions

- virtual void **AnimationComplete** ()

Internal function used to indicate that an animation has completed.
- virtual bool **AttachToVisualImpl** (**VisualPtr** _visual, bool _inheritOrientation, double _minDist=0, double _maxDist=0)

Set the camera to be attached to a visual.
- virtual bool **TrackVisualImpl** (**VisualPtr** _visual)

Set the camera to track a scene node.

Additional Inherited Members

10.253.1 Detailed Description

A camera used for user visualization of a scene.

10.253.2 Constructor & Destructor Documentation

10.253.2.1 gazebo::rendering::UserCamera::UserCamera (const std::string & *_name*, ScenePtr *_scene*)

Constructor.

Parameters

in	<i>_name</i>	Name of the camera.
in	<i>_scene</i>	Scene (p. 879) to put the camera in.

10.253.2.2 virtual gazebo::rendering::UserCamera::~~UserCamera () [virtual]

Destructor.

10.253.3 Member Function Documentation

10.253.3.1 virtual void gazebo::rendering::UserCamera::AnimationComplete () [protected],[virtual]

Internal function used to indicate that an animation has completed.

Reimplemented from **gazebo::rendering::Camera** (p. 204).

10.253.3.2 virtual bool gazebo::rendering::UserCamera::AttachToVisualImpl (VisualPtr *_visual*, bool *_inheritOrientation*, double *_minDist* = 0, double *_maxDist* = 0) [protected],[virtual]

Set the camera to be attached to a visual.

This causes the camera to move in relation to the specified visual.

Parameters

in	<i>_visual</i>	The visual to attach to.
in	<i>_inheritOrientation</i>	True if the camera should also rotate when the visual rotates.
in	<i>_minDist</i>	Minimum distance the camera can get to the visual.
in	<i>_maxDist</i>	Maximum distance the camera can get from the visual.

Returns

True if successfully attach to the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 206).

10.253.3.3 void gazebo::rendering::UserCamera::EnableViewController (bool *_value*) const

Set whether the view controller is enabled.

The view controller is used to handle user camera movements.

Parameters

in	<i>_value</i>	True to enable viewcontroller, False to disable.
----	---------------	--

10.253.3.4 void gazebo::rendering::UserCamera::Fini () [virtual]

Finalize.

Reimplemented from **gazebo::rendering::Camera** (p. 207).

10.253.3.5 float gazebo::rendering::UserCamera::GetAvgFPS () const [virtual]

Get the average frames per second.

Returns

The average rendering frames per second

Reimplemented from **gazebo::rendering::Camera** (p. 207).

10.253.3.6 **GUIOverlay*** gazebo::rendering::UserCamera::GetGUIOverlay ()

Get the GUI overlay.

An overlay allows you to draw 2D elements on the viewport.

Returns

Pointer to the **GUIOverlay** (p. 494).

10.253.3.7 virtual unsigned int gazebo::rendering::UserCamera::GetImageHeight () const [virtual]

Get the height of the image.

Returns

Image height

Reimplemented from **gazebo::rendering::Camera** (p. 209).

10.253.3.8 virtual unsigned int gazebo::rendering::UserCamera::GetImageWidth () const [virtual]

Get the width of the image.

Returns

Image width

Reimplemented from **gazebo::rendering::Camera** (p. 210).

10.253.3.9 `unsigned int gazebo::rendering::UserCamera::GetTriangleCount () const` [virtual]

Get the triangle count.

Returns

The number of triangles currently being rendered.

Reimplemented from `gazebo::rendering::Camera` (p. 212).

10.253.3.10 `std::string gazebo::rendering::UserCamera::GetViewControllerTypeString ()`

Get current view controller type.

Returns

Type of the current view controller: "orbit", "fps"

10.253.3.11 `VisualPtr gazebo::rendering::UserCamera::GetVisual (const math::Vector2i & _mousePos, std::string & _mod)`

Get an entity at a pixel location using a camera.

Used for mouse picking.

Parameters

<code>in</code>	<code>_mousePos</code>	The position of the mouse in screen coordinates
<code>out</code>	<code>_mod</code>	Used for object manipulation

Returns

The selected entity, or NULL

10.253.3.12 `VisualPtr gazebo::rendering::UserCamera::GetVisual (const math::Vector2i & _mousePos) const`

Get a visual at a mouse position.

Parameters

<code>in</code>	<code>_mousePos</code>	2D position of the mouse in pixels.
-----------------	------------------------	-------------------------------------

10.253.3.13 `void gazebo::rendering::UserCamera::HandleKeyPressEvent (const std::string & _key)`

Handle a key press.

Parameters

<code>in</code>	<code>_key</code>	The key pressed.
-----------------	-------------------	------------------

10.253.3.14 void gazebo::rendering::UserCamera::HandleKeyReleaseEvent (const std::string & *_key*)

Handle a key release.

Parameters

in	<i>_key</i>	The key released.
----	-------------	-------------------

10.253.3.15 void gazebo::rendering::UserCamera::HandleMouseEvent (const common::MouseEvent & *_evt*)

Handle a mouse event.

Parameters

in	<i>_evt</i>	The mouse event.
----	-------------	------------------

10.253.3.16 void gazebo::rendering::UserCamera::Init () [virtual]

Initialize.

Reimplemented from **gazebo::rendering::Camera** (p. 214).

10.253.3.17 bool gazebo::rendering::UserCamera::IsCameraSetInWorldFile ()

brief Show if the user camera pose has changed in the world file.

return true if the camera pose changed in the world file.

10.253.3.18 void gazebo::rendering::UserCamera::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the user camera.

Parameters

in	<i>_sdf</i>	Parameters for the camera.
----	-------------	----------------------------

Reimplemented from **gazebo::rendering::Camera** (p. 215).

10.253.3.19 void gazebo::rendering::UserCamera::Load () [virtual]

Generic load function.

Reimplemented from **gazebo::rendering::Camera** (p. 215).

10.253.3.20 virtual bool gazebo::rendering::UserCamera::MoveToPosition (const math::Pose & *_pose*, double *_time*)
[virtual]

Move the camera to a position (this is an animated motion).

See Also

Camera::MoveToPositions (p. 216)

Parameters

in	<code>_pose</code>	End position of the camera
in	<code>_time</code>	Duration of the camera's movement

Reimplemented from **gazebo::rendering::Camera** (p. 215).

10.253.3.21 `void gazebo::rendering::UserCamera::MoveToVisual (VisualPtr _visual)`

Move the camera to focus on a visual.

Parameters

in	<code>_visual</code>	Visual (p. 1196) to move the camera to.
----	----------------------	--

10.253.3.22 `void gazebo::rendering::UserCamera::MoveToVisual (const std::string & _visualName)`

Move the camera to focus on a visual.

Parameters

in	<code>_visualName</code>	Name of the visual to move the camera to.
----	--------------------------	---

10.253.3.23 `virtual void gazebo::rendering::UserCamera::PostRender () [virtual]`

Post render.

Reimplemented from **gazebo::rendering::Camera** (p. 216).

10.253.3.24 `void gazebo::rendering::UserCamera::Resize (unsigned int _w, unsigned int _h)`

Resize the camera.

Parameters

in	<code>_w</code>	Width of the camera image.
in	<code>_h</code>	Height of the camera image.

10.253.3.25 `void gazebo::rendering::UserCamera::SetFocalPoint (const math::Vector3 & _pt)`

Set the point the camera should orbit around.

Parameters

in	<code>_pt</code>	The focal point
----	------------------	-----------------

10.253.3.26 `virtual void gazebo::rendering::UserCamera::SetRenderTarget (Ogre::RenderTarget * _target) [virtual]`

Set to true to enable rendering.

Use this only if you really know what you're doing.

Parameters

<i>in</i>	<i>_target</i>	The new rendering target.
-----------	----------------	---------------------------

Reimplemented from `gazebo::rendering::Camera` (p. 219).

10.253.3.27 `void gazebo::rendering::UserCamera::SetUseSDFPose (bool _value)`

brief Set if the user camera pose has changed in the world file.

Parameters

<i>in</i>	<i>_value</i>	True if the camera pose changed in the world file.
-----------	---------------	--

10.253.3.28 `void gazebo::rendering::UserCamera::SetViewController (const std::string & _type)`

Set view controller.

Parameters

<i>in</i>	<i>_type</i>	The type of view controller: "orbit", "fps"
-----------	--------------	---

10.253.3.29 `void gazebo::rendering::UserCamera::SetViewController (const std::string & _type, const math::Vector3 & _pos)`

Set view controller.

Parameters

<i>in</i>	<i>_type</i>	The type of view controller: "orbit", "fps"
<i>in</i>	<i>_pos</i>	The initial pose of the camera.

10.253.3.30 `void gazebo::rendering::UserCamera::SetViewportDimensions (float _x, float _y, float _w, float _h)`

Set the dimensions of the viewport.

Parameters

<i>in</i>	<i>_x</i>	X position of the viewport.
<i>in</i>	<i>_y</i>	Y position of the viewport.
<i>in</i>	<i>_w</i>	Width of the viewport.
<i>in</i>	<i>_h</i>	Height of the viewport.

10.253.3.31 `virtual void gazebo::rendering::UserCamera::SetWorldPose (const math::Pose & _pose) [virtual]`

Set the pose in the world coordinate frame.

Parameters

in	_pose	New pose of the camera.
----	-------	-------------------------

Reimplemented from `gazebo::rendering::Camera` (p. 220).

10.253.3.32 `virtual bool gazebo::rendering::UserCamera::TrackVisualImpl (VisualPtr _visual) [protected], [virtual]`

Set the camera to track a scene node.

Tracking just causes the camera to rotate to follow the visual.

Parameters

in	_visual	Visual (p. 1196) to track.
----	---------	-----------------------------------

Returns

True if the camera is now tracking the visual.

Reimplemented from `gazebo::rendering::Camera` (p. 221).

10.253.3.33 `virtual void gazebo::rendering::UserCamera::Update () [virtual]`

Render the camera.

Reimplemented from `gazebo::rendering::Camera` (p. 222).

The documentation for this class was generated from the following file:

- **UserCamera.hh**

10.254 gazebo::rendering::UserCameraPrivate Class Reference

Private data for the `UserCamera` (p. 1137) class.

```
#include <UserCameraPrivate.hh>
```

Public Attributes

- **FPSViewController * fpsViewController**
A FPS view controller.
- **GUIOverlay * gui**
The GUI overlay.
- **bool isCameraSetInWorldFile**
Flag to detect if the user changed the camera pose in the world file.

- **OrbitViewController** * **orbitViewController**

An orbit view controller.

- **SelectionBuffer** * **selectionBuffer**

Draws a 3D axis in the viewport.

- **ViewController** * **viewController**

The currently active view controller.

10.254.1 Detailed Description

Private data for the **UserCamera** (p. 1137) class.

10.254.2 Member Data Documentation

10.254.2.1 FPSViewController* gazebo::rendering::UserCameraPrivate::fpsViewController

A FPS view controller.

10.254.2.2 GUIOverlay* gazebo::rendering::UserCameraPrivate::gui

The GUI overlay.

10.254.2.3 bool gazebo::rendering::UserCameraPrivate::isCameraSetInWorldFile

Flag to detect if the user changed the camera pose in the world file.

10.254.2.4 OrbitViewController* gazebo::rendering::UserCameraPrivate::orbitViewController

An orbit view controller.

10.254.2.5 SelectionBuffer* gazebo::rendering::UserCameraPrivate::selectionBuffer

Draws a 3D axis in the viewport.

Used to select objects from mouse clicks.

10.254.2.6 ViewController* gazebo::rendering::UserCameraPrivate::viewController

The currently active view controller.

The documentation for this class was generated from the following file:

- **UserCameraPrivate.hh**

10.255 gazebo::math::Vector2d Class Reference

Generic double x, y vector.

```
#include <Vector2d.hh>
```

Public Member Functions

- **Vector2d** ()
Constructor.
- **Vector2d** (const double &_x, const double &_y)
Constructor.
- **Vector2d** (const **Vector2d** &_v)
Copy constructor.
- virtual ~**Vector2d** ()
Destructor.
- **Vector2d Cross** (const **Vector2d** &_v) const
Return the cross product of this vector and _v.
- double **Distance** (const **Vector2d** &_pt) const
Calc distance to the given point.
- bool **IsFinite** () const
See if a point is finite (e.g., not nan)
- void **Normalize** ()
Normalize the vector length.
- bool **operator!=** (const **Vector2d** &_v) const
Not equal to operator.
- const **Vector2d operator*** (const **Vector2d** &_v) const
Multiplication operators.
- const **Vector2d operator*** (double _v) const
Multiplication operators.
- const **Vector2d & operator*=** (const **Vector2d** &_v)
Multiplication assignment operator.
- const **Vector2d & operator*=** (double _v)
Multiplication assignment operator.
- **Vector2d operator+** (const **Vector2d** &_v) const
Addition operator.
- const **Vector2d & operator+=** (const **Vector2d** &_v)
Addition assignment operator.
- **Vector2d operator-** (const **Vector2d** &_v) const
Subtraction operator.
- const **Vector2d & operator-=** (const **Vector2d** &_v)
Subtraction assignment operator.
- const **Vector2d operator/** (const **Vector2d** &_v) const
Division operator.
- const **Vector2d operator/** (double _v) const
Division operator.
- const **Vector2d & operator/=** (const **Vector2d** &_v)
Division operator.
- const **Vector2d & operator/=** (double _v)
Division operator.
- **Vector2d & operator=** (const **Vector2d** &_v)
Assignment operator.
- const **Vector2d & operator=** (double _v)

Assignment operator.

- bool **operator==** (const **Vector2d** &_v) const

Equal to operator.

- double **operator[]** (unsigned int _index) const

Array subscript operator.

- void **Set** (double _x, double _y)

Set the contents of the vector.

Public Attributes

- double **x**

x data

- double **y**

y data

Friends

- std::ostream & **operator<<** (std::ostream &_out, const **gazebo::math::Vector2d** &_pt)

Stream extraction operator.

- std::istream & **operator>>** (std::istream &_in, **gazebo::math::Vector2d** &_pt)

Stream extraction operator.

10.255.1 Detailed Description

Generic double x, y vector.

10.255.2 Constructor & Destructor Documentation

10.255.2.1 gazebo::math::Vector2d::Vector2d ()

Constructor.

10.255.2.2 gazebo::math::Vector2d::Vector2d (const double & _x, const double & _y)

Constructor.

Parameters

in	_x	value along x
in	_y	value along y

10.255.2.3 gazebo::math::Vector2d::Vector2d (const Vector2d & _v)

Copy constructor.

Parameters

in	<code>_v</code>	the value
----	-----------------	-----------

10.255.2.4 `virtual gazebo::math::Vector2d::~~Vector2d () [virtual]`

Destructor.

10.255.3 Member Function Documentation

10.255.3.1 `Vector2d gazebo::math::Vector2d::Cross (const Vector2d & _v) const`

Return the cross product of this vector and `_v`.

Parameters

in	<code>_v</code>	the vector
----	-----------------	------------

Returns

the cross product

10.255.3.2 `double gazebo::math::Vector2d::Distance (const Vector2d & _pt) const`

Calc distance to the given point.

Parameters

in	<code>_pt</code>	The point to measure to
----	------------------	-------------------------

Returns

the distance

10.255.3.3 `bool gazebo::math::Vector2d::IsFinite () const`

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.255.3.4 `void gazebo::math::Vector2d::Normalize ()`

Normalize the vector length.

10.255.3.5 `bool gazebo::math::Vector2d::operator!= (const Vector2d & _v) const`

Not equal to operator.

Returns

true if elements are of different values (tolerance 1e-6)

10.255.3.6 `const Vector2d gazebo::math::Vector2d::operator*(const Vector2d & _v) const`

Multiplication operators.

Parameters

<code>in</code>	<code>_v</code>	the vector
-----------------	-----------------	------------

Returns

the result

10.255.3.7 `const Vector2d gazebo::math::Vector2d::operator*(double _v) const`

Multiplication operators.

Parameters

<code>in</code>	<code>_v</code>	the scaling factor
-----------------	-----------------	--------------------

Returns

a scaled vector

10.255.3.8 `const Vector2d& gazebo::math::Vector2d::operator*=(const Vector2d & _v)`

Multiplication assignment operator.

Remarks

this is an element wise multiplication

Parameters

<code>in</code>	<code>_v</code>	the vector
-----------------	-----------------	------------

Returns

this

10.255.3.9 `const Vector2d& gazebo::math::Vector2d::operator*=(double _v)`

Multiplication assignment operator.

Parameters

in	_v	the scaling factor
----	----	--------------------

Returns

a scaled vector

10.255.3.10 **Vector2d** gazebo::math::Vector2d::operator+ (const Vector2d & _v) const

Addition operator.

Parameters

in	_v	vector to add
----	----	---------------

Returns

sum vector

10.255.3.11 **const Vector2d&** gazebo::math::Vector2d::operator+= (const Vector2d & _v)

Addition assignment operator.

Parameters

in	_v	the vector to add
----	----	-------------------

10.255.3.12 **Vector2d** gazebo::math::Vector2d::operator- (const Vector2d & _v) const

Subtraction operator.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

the subtracted vector

10.255.3.13 **const Vector2d&** gazebo::math::Vector2d::operator-= (const Vector2d & _v)

Subtraction assignment operator.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

this

10.255.3.14 `const Vector2d gazebo::math::Vector2d::operator/ (const Vector2d & _v) const`

Division operator.

Remarks

this is an element wise division

Parameters

<code>in</code>	<code>_v</code>	a vector
-----------------	-----------------	----------

Returns

a result

10.255.3.15 `const Vector2d gazebo::math::Vector2d::operator/ (double _v) const`

Division operator.

Parameters

<code>in</code>	<code>_v</code>	the value
-----------------	-----------------	-----------

Returns

a vector

10.255.3.16 `const Vector2d& gazebo::math::Vector2d::operator/= (const Vector2d & _v)`

Division operator.

Remarks

this is an element wise division

Parameters

<code>in</code>	<code>_v</code>	a vector
-----------------	-----------------	----------

Returns

this

10.255.3.17 `const Vector2d& gazebo::math::Vector2d::operator/= (double _v)`

Division operator.

Parameters

<code>in</code>	<code>_v</code>	the divisor
-----------------	-----------------	-------------

Returns

a vector

10.255.3.18 `Vector2d& gazebo::math::Vector2d::operator= (const Vector2d & _v)`

Assignment operator.

Parameters

<code>in</code>	<code>_v</code>	a value for x and y element
-----------------	-----------------	-----------------------------

Returns

this

10.255.3.19 `const Vector2d& gazebo::math::Vector2d::operator= (double _v)`

Assignment operator.

Parameters

<code>in</code>	<code>_v</code>	the value for x and y element
-----------------	-----------------	-------------------------------

Returns

this

10.255.3.20 `bool gazebo::math::Vector2d::operator==(const Vector2d & _v) const`

Equal to operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to compare to
-----------------	-----------------	--------------------------

Returns

true if the elements of the 2 vectors are equal within a tolerance (1e-6)

10.255.3.21 `double gazebo::math::Vector2d::operator[] (unsigned int _index) const`

Array subscript operator.

Parameters

<code>in</code>	<code><i>_index</i></code>	the index
-----------------	----------------------------	-----------

Returns

the value, or 0 if `_index` is out of bounds

10.255.3.22 `void gazebo::math::Vector2d::Set (double _x, double _y)`

Set the contents of the vector.

Parameters

<code>in</code>	<code><i>_x</i></code>	value along x
<code>in</code>	<code><i>_y</i></code>	value along y

10.255.4 Friends And Related Function Documentation

10.255.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector2d & _pt) [friend]`

Stream extraction operator.

Parameters

<code>in</code>	<code><i>_out</i></code>	output stream
<code>in</code>	<code><i>_pt</i></code>	Vector2d (p. 1147) to output

Returns

The stream

10.255.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector2d & _pt) [friend]`

Stream extraction operator.

Parameters

<code>in</code>	<code><i>_in</i></code>	input stream
<code>in</code>	<code><i>_pt</i></code>	Vector3 (p. 1165) to read values into

Returns

The stream

10.255.5 Member Data Documentation

10.255.5.1 double gazebo::math::Vector2d::x

x data

10.255.5.2 double gazebo::math::Vector2d::y

y data

The documentation for this class was generated from the following file:

- **Vector2d.hh**

10.256 gazebo::math::Vector2i Class Reference

Generic integer x, y vector.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Vector2i** ()
Constructor.
- **Vector2i** (const int &_x, const int &_y)
Constructor.
- **Vector2i** (const **Vector2i** &_pt)
Copy onstructor.
- virtual ~**Vector2i** ()
Destructor.
- **Vector2i Cross** (const **Vector2i** &_pt) const
Return the cross product of this vector and _pt.
- int **Distance** (const **Vector2i** &_pt) const
Calc distance to the given point.
- bool **IsFinite** () const
See if a point is finite (e.g., not nan)
- void **Normalize** ()
Normalize the vector length.
- bool **operator!=** (const **Vector2i** &_v) const
Equality operators.
- const **Vector2i operator*** (const **Vector2i** &_v) const
Multiplication operator.
- const **Vector2i operator*** (int _v) const
Multiplication operator.
- const **Vector2i & operator*=** (const **Vector2i** &_v)
Multiplication operators.
- const **Vector2i & operator*=** (int _v)
Multiplication operator.
- **Vector2i operator+** (const **Vector2i** &_v) const

Addition operator.

- const **Vector2i** & **operator+=** (const **Vector2i** &_v)

Addition assignment operator.

- **Vector2i** **operator-** (const **Vector2i** &_v) const

Subtraction operator.

- const **Vector2i** & **operator-=** (const **Vector2i** &_v)

Subtraction operators.

- const **Vector2i** **operator/** (const **Vector2i** &_v) const

Division operator.

- const **Vector2i** **operator/** (int _v) const

Division operator.

- const **Vector2i** & **operator/=** (const **Vector2i** &_v)

Division operator.

- const **Vector2i** & **operator/=** (int _v)

Division operator.

- **Vector2i** & **operator=** (const **Vector2i** &_v)

Assignment operator.

- const **Vector2i** & **operator=** (int _value)

Assignment operator.

- bool **operator==** (const **Vector2i** &_v) const

Equality operator.

- int **operator[]** (unsigned int _index) const

Array subscript operator.

- void **Set** (int _x, int _y)

Set the contents of the vector.

Public Attributes

- int **x**
x data
- int **y**
y data

Friends

- std::ostream & **operator<<** (std::ostream &_out, const gazebo::math::Vector2i &_pt)
Stream insertion operator.
- std::istream & **operator>>** (std::istream &_in, gazebo::math::Vector2i &_pt)
Stream extraction operator.

10.256.1 Detailed Description

Generic integer x, y vector.

10.256.2 Constructor & Destructor Documentation

10.256.2.1 gazebo::math::Vector2i::Vector2i ()

Constructor.

10.256.2.2 gazebo::math::Vector2i::Vector2i (const int & _x, const int & _y)

Constructor.

Parameters

in	<code>_x</code>	value along x
in	<code>_y</code>	value along y

10.256.2.3 gazebo::math::Vector2i::Vector2i (const Vector2i & _pt)

Copy onstructor.

Parameters

in	<code>_pt</code>	a point
----	------------------	---------

10.256.2.4 virtual gazebo::math::Vector2i::~~Vector2i () [virtual]

Destructor.

10.256.3 Member Function Documentation

10.256.3.1 Vector2i gazebo::math::Vector2i::Cross (const Vector2i & _pt) const

Return the cross product of this vector and `_pt`.

Parameters

in	<code>_pt</code>	the other vector
----	------------------	------------------

Returns

the product

10.256.3.2 int gazebo::math::Vector2i::Distance (const Vector2i & _pt) const

Calc distance to the given point.

Parameters

in	<code>_pt</code>	a point
----	------------------	---------

Returns

the distance

10.256.3.3 `bool gazebo::math::Vector2i::IsFinite () const`

See if a point is finite (e.g., not nan)

Returns

the result

10.256.3.4 `void gazebo::math::Vector2i::Normalize ()`

Normalize the vector length.

10.256.3.5 `bool gazebo::math::Vector2i::operator!=(const Vector2i & _v) const`

Equality operators.

Parameters

<code>_v</code>	the vector to compare with
-----------------	----------------------------

Returns

true if component have different values, false otherwise

10.256.3.6 `const Vector2i gazebo::math::Vector2i::operator*(const Vector2i & _v) const`

Multiplication operator.

Remarks

this is an element wise multiplication

Parameters

<code>in</code>	<code>_v</code>	the vector
-----------------	-----------------	------------

Returns

the result

10.256.3.7 `const Vector2i gazebo::math::Vector2i::operator*(int _v) const`

Multiplication operator.

Parameters

<code>in</code>	<code>_v</code>	the scaling factor
-----------------	-----------------	--------------------

Returns

the result

10.256.3.8 `const Vector2i& gazebo::math::Vector2i::operator*=(const Vector2i & _v)`

Multiplication operators.

Remarks

this is an element wise multiplication

Parameters

<code>in</code>	<code>_v</code>	the vector
-----------------	-----------------	------------

Returns

this

10.256.3.9 `const Vector2i& gazebo::math::Vector2i::operator*=(int _v)`

Multiplication operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

this

10.256.3.10 `Vector2i gazebo::math::Vector2i::operator+(const Vector2i & _v) const`

Addition operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to add
-----------------	-----------------	-------------------

Returns

the sum vector

10.256.3.11 `const Vector2i& gazebo::math::Vector2i::operator+=(const Vector2i & _v)`

Addition assignment operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to add
-----------------	-----------------	-------------------

Returns

this

10.256.3.12 `Vector2i gazebo::math::Vector2i::operator-(const Vector2i & _v) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

the result vector

10.256.3.13 `const Vector2i& gazebo::math::Vector2i::operator-=(const Vector2i & _v)`

Subtraction operators.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

this

10.256.3.14 `const Vector2i gazebo::math::Vector2i::operator/(const Vector2i & _v) const`

Division operator.

Remarks

this is an element wise division.

Parameters

<code>in</code>	<code>_v</code>	the vector to divide
-----------------	-----------------	----------------------

Returns

the result

10.256.3.15 `const Vector2i gazebo::math::Vector2i::operator/ (int _v) const`

Division operator.

Remarks

this is an element wise division.

Parameters

<code>in</code>	<code>_v</code>	the vector to divide
-----------------	-----------------	----------------------

Returns

the result

10.256.3.16 `const Vector2i& gazebo::math::Vector2i::operator/= (const Vector2i & _v)`

Division operator.

Remarks

this is an element wise division.

Parameters

<code>in</code>	<code>_v</code>	the vector to divide
-----------------	-----------------	----------------------

Returns

this

10.256.3.17 `const Vector2i& gazebo::math::Vector2i::operator/= (int _v)`

Division operator.

Remarks

this is an element wise division.

Parameters

<code>in</code>	<code>_v</code>	the vector to divide
-----------------	-----------------	----------------------

Returns

this

10.256.3.18 `Vector2i& gazebo::math::Vector2i::operator=(const Vector2i & _v)`

Assignment operator.

Parameters

<code>in</code>	<code>_v</code>	the value
-----------------	-----------------	-----------

Returns

this

10.256.3.19 `const Vector2i& gazebo::math::Vector2i::operator=(int _value)`

Assignment operator.

Parameters

<code>in</code>	<code>_value</code>	the value for x and y
-----------------	---------------------	-----------------------

Returns

this

10.256.3.20 `bool gazebo::math::Vector2i::operator==(const Vector2i & _v) const`

Equality operator.

Parameters

<code>_v</code>	the vector to compare with
-----------------	----------------------------

Returns

true if component have the same values, false otherwise

10.256.3.21 `int gazebo::math::Vector2i::operator[](unsigned int _index) const`

Array subscript operator.

Parameters

<code>in</code>	<code>_index</code>	the array index
-----------------	---------------------	-----------------

10.256.3.22 `void gazebo::math::Vector2i::Set (int _x, int _y)`

Set the contents of the vector.

Parameters

<code>in</code>	<code>_x</code>	value along x
<code>in</code>	<code>_y</code>	value along y

10.256.4 Friends And Related Function Documentation

10.256.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector2i & _pt)` [`friend`]

Stream insertion operator.

Parameters

<code>in</code>	<code>_out</code>	output stream
<code>in</code>	<code>pt</code>	Vector2i (p. 1156) to output

Returns

the stream

10.256.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector2i & _pt)` [`friend`]

Stream extraction operator.

Parameters

<code>in</code>	<code>_in</code>	input stream
<code>in</code>	<code>_pt</code>	Vector3 (p. 1165) to read values into

Returns

The stream

10.256.5 Member Data Documentation

10.256.5.1 `int gazebo::math::Vector2i::x`

x data

10.256.5.2 `int gazebo::math::Vector2i::y`

y data

The documentation for this class was generated from the following file:

- **Vector2i.hh**

10.257 gazebo::math::Vector3 Class Reference

The **Vector3** (p. 1165) class represents the generic vector containing 3 elements.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Vector3** ()
Constructor.
- **Vector3** (const double &_x, const double &_y, const double &_z)
Constructor.
- **Vector3** (const **Vector3** &_v)
Copy constructor.
- virtual ~**Vector3** ()
Destructor.
- void **Correct** ()
Corrects any nan values.
- **Vector3 Cross** (const **Vector3** &_pt) const
Return the cross product of this vector and pt.
- double **Distance** (const **Vector3** &_pt) const
Calc distance to the given point.
- double **Distance** (double _x, double _y, double _z) const
Calc distance to the given point.
- double **Dot** (const **Vector3** &_pt) const
Return the dot product of this vector and pt.
- bool **Equal** (const **Vector3** &_v) const
Equality test.
- **Vector3 GetAbs** () const
Get the absolute value of the vector.
- double **GetDistToLine** (const **Vector3** &_pt1, const **Vector3** &_pt2)
Get distance to a line.
- double **GetLength** () const
Returns the length (magnitude) of the vector \ return the length.
- double **GetMax** () const
Get the maximum value in the vector.
- double **GetMin** () const
Get the minimum value in the vector.
- **Vector3 GetPerpendicular** () const
Return a vector that is perpendicular to this one.
- **Vector3 GetRounded** () const
Get a rounded version of this vector.
- double **GetSquaredLength** () const
Return the square of the length (magnitude) of the vector.
- double **GetSum** () const
Return the sum of the values.
- bool **IsFinite** () const

- See if a point is finite (e.g., not nan)*
- **Vector3 Normalize ()**
Normalize the vector length.
 - **bool operator!= (const Vector3 &_v) const**
Not equal to operator.
 - **Vector3 operator* (const Vector3 &_p) const**
Multiplication operator.
 - **Vector3 operator* (double _v) const**
Multiplication operators.
 - **const Vector3 & operator*= (const Vector3 &_v)**
Multiplication operators.
 - **const Vector3 & operator*= (double _v)**
Multiplication operator.
 - **Vector3 operator+ (const Vector3 &_v) const**
Addition operator.
 - **const Vector3 & operator+= (const Vector3 &_v)**
Addition assignment operator.
 - **Vector3 operator- () const**
Negation operator.
 - **Vector3 operator- (const Vector3 &_pt) const**
Subtraction operators.
 - **const Vector3 & operator-= (const Vector3 &_pt)**
Subtraction operators.
 - **const Vector3 operator/ (const Vector3 &_pt) const**
Division operator.
 - **const Vector3 operator/ (double _v) const**
Division operator.
 - **const Vector3 & operator/= (const Vector3 &_pt)**
Division assignment operator.
 - **const Vector3 & operator/= (double _v)**
Division operator.
 - **Vector3 & operator= (const Vector3 &_v)**
Assignment operator.
 - **Vector3 & operator= (double _value)**
Assignment operator.
 - **bool operator== (const Vector3 &_pt) const**
Equal to operator.
 - **double operator[] (unsigned int index) const**
[] operator
 - **Vector3 Round ()**
Round to near whole number, return the result.
 - **void Round (int _precision)**
Round all values to _precision decimal places.
 - **void Set (double _x=0, double _y=0, double _z=0)**
Set the contents of the vector.
 - **void SetToMax (const Vector3 &_v)**
Set this vector's components to the maximum of itself and the passed in vector.
 - **void SetToMin (const Vector3 &_v)**
Set this vector's components to the minimum of itself and the passed in vector.

Static Public Member Functions

- static **Vector3 GetNormal** (const **Vector3** &_v1, const **Vector3** &_v2, const **Vector3** &_v3)
Get a normal vector to a triangle.

Public Attributes

- double **x**
X location.
- double **y**
Y location.
- double **z**
Z location.

Static Public Attributes

- static const **Vector3 One**
math::Vector3(1, 1, 1)
- static const **Vector3 UnitX**
math::Vector3(1, 0, 0)
- static const **Vector3 UnitY**
math::Vector3(0, 1, 0)
- static const **Vector3 UnitZ**
math::Vector3(0, 0, 1)
- static const **Vector3 Zero**
math::Vector3(0, 0, 0)

Friends

- **Vector3 operator*** (double _s, const **Vector3** &_v)
Multiplication operators.
- std::ostream & **operator<<** (std::ostream &_out, const **gazebo::math::Vector3** &_pt)
Stream insertion operator.
- std::istream & **operator>>** (std::istream &_in, **gazebo::math::Vector3** &_pt)
Stream extraction operator.

10.257.1 Detailed Description

The **Vector3** (p. 1165) class represents the generic vector containing 3 elements.

Since it's commonly used to keep coordinate system related information, its elements are labeled by x, y, z.

10.257.2 Constructor & Destructor Documentation

10.257.2.1 gazebo::math::Vector3::Vector3 ()

Constructor.

10.257.2.2 `gazebo::math::Vector3::Vector3 (const double & _x, const double & _y, const double & _z)`

Constructor.

Parameters

in	<code>_x</code>	value along x
in	<code>_y</code>	value along y
in	<code>_z</code>	value along z

10.257.2.3 `gazebo::math::Vector3::Vector3 (const Vector3 & _v)`

Copy constructor.

Parameters

in	<code>_v</code>	a vector
----	-----------------	----------

10.257.2.4 `virtual gazebo::math::Vector3::~~Vector3 () [virtual]`

Destructor.

10.257.3 Member Function Documentation

10.257.3.1 `void gazebo::math::Vector3::Correct () [inline]`

Corrects any nan values.

10.257.3.2 `Vector3 gazebo::math::Vector3::Cross (const Vector3 & _pt) const`

Return the cross product of this vector and pt.

Returns

the product

10.257.3.3 `double gazebo::math::Vector3::Distance (const Vector3 & _pt) const`

Calc distance to the given point.

Parameters

in	<code>_pt</code>	the point
----	------------------	-----------

Returns

the distance

10.257.3.4 `double gazebo::math::Vector3::Distance (double _x, double _y, double _z) const`

Calc distance to the given point.

Parameters

<code>in</code>	<code>_x</code>	value along x
<code>in</code>	<code>_y</code>	value along y
<code>in</code>	<code>_z</code>	value along z

Returns

the distance

10.257.3.5 `double gazebo::math::Vector3::Dot (const Vector3 & _pt) const`

Return the dot product of this vector and pt.

Returns

the product

10.257.3.6 `bool gazebo::math::Vector3::Equal (const Vector3 & _v) const`

Equality test.

Remarks

This is equivalent to the `==` operator

Parameters

<code>in</code>	<code>_v</code>	the other vector
-----------------	-----------------	------------------

Returns

true if the 2 vectors have the same values, false otherwise

10.257.3.7 `Vector3 gazebo::math::Vector3::GetAbs () const`

Get the absolute value of the vector.

Returns

a vector with positive elements

10.257.3.8 `double gazebo::math::Vector3::GetDistToLine (const Vector3 & _pt1, const Vector3 & _pt2)`

Get distance to a line.

Parameters

in	<code>_pt1</code>	first point on the line
in	<code>_pt2</code>	second point on the line

Returns

the minimum distance from this point to the line

10.257.3.9 double gazebo::math::Vector3::GetLength () const

Returns the length (magnitude) of the vector \ return the length.

10.257.3.10 double gazebo::math::Vector3::GetMax () const

Get the maximum value in the vector.

Returns

the maximum element

10.257.3.11 double gazebo::math::Vector3::GetMin () const

Get the minimum value in the vector.

Returns

the minimum element

10.257.3.12 static Vector3 gazebo::math::Vector3::GetNormal (const Vector3 & _v1, const Vector3 & _v2, const Vector3 & _v3) [static]

Get a normal vector to a triangle.

Parameters

in	<code>_v1</code>	first vertex of the triangle
in	<code>_v2</code>	second vertex
in	<code>_v3</code>	third vertex

Returns

the normal

10.257.3.13 Vector3 gazebo::math::Vector3::GetPerpendicular () const

Return a vector that is perpendicular to this one.

Returns

an orthogonal vector

10.257.3.14 `Vector3 gazebo::math::Vector3::GetRounded () const`

Get a rounded version of this vector.

Returns

a rounded vector

10.257.3.15 `double gazebo::math::Vector3::GetSquaredLength () const`

Return the square of the length (magnitude) of the vector.

Returns

the squared length

10.257.3.16 `double gazebo::math::Vector3::GetSum () const`

Return the sum of the values.

Returns

the sum

10.257.3.17 `bool gazebo::math::Vector3::IsFinite () const`

See if a point is finite (e.g., not nan)

10.257.3.18 `Vector3 gazebo::math::Vector3::Normalize ()`

Normalize the vector length.

Returns

unit length vector

10.257.3.19 `bool gazebo::math::Vector3::operator!=(const Vector3 & _v) const`

Not equal to operator.

Parameters

<code>in</code>	<code>_v</code>	The vector to compare against
-----------------	-----------------	-------------------------------

Returns

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.257.3.20 **Vector3** gazebo::math::Vector3::operator* (const Vector3 & _p) const

Multiplication operator.

Remarks

this is an element wise multiplication, not a cross product

Parameters

in	_v	
----	----	--

10.257.3.21 **Vector3** gazebo::math::Vector3::operator* (double _v) const

Multiplication operators.

Parameters

in	_v	the scaling factor
----	----	--------------------

Returns

a scaled vector

10.257.3.22 **const Vector3&** gazebo::math::Vector3::operator*= (const Vector3 & _v)

Multiplication operators.

Remarks

this is an element wise multiplication, not a cross product

Parameters

in	_v	a vector
----	----	----------

Returns

this

10.257.3.23 **const Vector3&** gazebo::math::Vector3::operator*= (double _v)

Multiplication operator.

Parameters

in	<code>_v</code>	scaling factor
----	-----------------	----------------

Returns

this

10.257.3.24 `Vector3 gazebo::math::Vector3::operator+ (const Vector3 & _v) const`

Addition operator.

Parameters

in	<code>_v</code>	vector to add
----	-----------------	---------------

Returns

the sum vector

10.257.3.25 `const Vector3& gazebo::math::Vector3::operator+= (const Vector3 & _v)`

Addition assignment operator.

Parameters

in	<code>_v</code>	vector to add
----	-----------------	---------------

10.257.3.26 `Vector3 gazebo::math::Vector3::operator- () const` `[inline]`

Negation operator.

Returns

negative of this vector

10.257.3.27 `Vector3 gazebo::math::Vector3::operator- (const Vector3 & _pt) const` `[inline]`

Subtraction operators.

Parameters

in	<code>_pt</code>	a vector to subtract
----	------------------	----------------------

Returns

a vector

References x, y, and z.

10.257.3.28 `const Vector3& gazebo::math::Vector3::operator-= (const Vector3 & _pt)`

Subtraction operators.

Parameters

in	_pt	subtrahend
----	-----	------------

10.257.3.29 `const Vector3 gazebo::math::Vector3::operator/ (const Vector3 & _pt) const`

Division operator.

[in] _pt the vector divisor

Remarks

this is an element wise division

Returns

a vector

10.257.3.30 `const Vector3 gazebo::math::Vector3::operator/ (double _v) const`

Division operator.

Remarks

this is an element wise division

Returns

a vector

10.257.3.31 `const Vector3& gazebo::math::Vector3::operator/= (const Vector3 & _pt)`

Division assignment operator.

[in] _pt the vector divisor

Remarks

this is an element wise division

Returns

a vector

10.257.3.32 `const Vector3& gazebo::math::Vector3::operator/= (double _v)`

Division operator.

Remarks

this is an element wise division

Returns

this

10.257.3.33 `Vector3& gazebo::math::Vector3::operator= (const Vector3 & _v)`

Assignment operator.

Parameters

<code>in</code>	<code>_v</code>	a new value
-----------------	-----------------	-------------

Returns

this

10.257.3.34 `Vector3& gazebo::math::Vector3::operator= (double _value)`

Assignment operator.

Parameters

<code>in</code>	<code>_value</code>	assigned to all elements
-----------------	---------------------	--------------------------

Returns

this

10.257.3.35 `bool gazebo::math::Vector3::operator==(const Vector3 & _pt) const`

Equal to operator.

Parameters

<code>in</code>	<code>_pt</code>	The vector to compare against
-----------------	------------------	-------------------------------

Returns

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.257.3.36 `double gazebo::math::Vector3::operator[] (unsigned int index) const`

[] operator

10.257.3.37 `Vector3 gazebo::math::Vector3::Round ()`

Round to near whole number, return the result.

Returns

the result

10.257.3.38 `void gazebo::math::Vector3::Round (int _precision)`

Round all values to `_precision` decimal places.

Parameters

<code>in</code>	<code>_precision</code>	the decimal places
-----------------	-------------------------	--------------------

10.257.3.39 `void gazebo::math::Vector3::Set (double _x = 0, double _y = 0, double _z = 0) [inline]`

Set the contents of the vector.

Parameters

<code>in</code>	<code>_x</code>	value along x
<code>in</code>	<code>_y</code>	value along y
<code>in</code>	<code>_z</code>	value along z

10.257.3.40 `void gazebo::math::Vector3::SetToMax (const Vector3 & _v)`

Set this vector's components to the maximum of itself and the passed in vector.

Parameters

<code>in</code>	<code>_v</code>	the maximum clamping vector
-----------------	-----------------	-----------------------------

10.257.3.41 `void gazebo::math::Vector3::SetToMin (const Vector3 & _v)`

Set this vector's components to the minimum of itself and the passed in vector.

Parameters

<code>in</code>	<code>_v</code>	the minimum clamping vector
-----------------	-----------------	-----------------------------

10.257.4 Friends And Related Function Documentation

10.257.4.1 Vector3 operator*(double _s, const Vector3 & _v) [friend]

Multiplication operators.

Parameters

<i>in</i>	<i>_s</i>	the scaling factor
<i>in</i>	<i>_v</i>	input vector

Returns

a scaled vector

10.257.4.2 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector3 & _pt) [friend]

Stream insertion operator.

Parameters

<i>_out</i>	output stream
<i>_pt</i>	Vector3 (p. 1165) to output

Returns

the stream

10.257.4.3 std::istream& operator>> (std::istream & _in, gazebo::math::Vector3 & _pt) [friend]

Stream extraction operator.

Parameters

<i>_in</i>	input stream
<i>_pt</i>	vector3 to read values into

Returns

the stream

10.257.5 Member Data Documentation

10.257.5.1 const Vector3 gazebo::math::Vector3::One [static]

math::Vector3(1, 1, 1)

10.257.5.2 const Vector3 gazebo::math::Vector3::UnitX [static]

math::Vector3(1, 0, 0)

10.257.5.3 `const Vector3 gazebo::math::Vector3::UnitY` `[static]`

`math::Vector3(0, 1, 0)`

10.257.5.4 `const Vector3 gazebo::math::Vector3::UnitZ` `[static]`

`math::Vector3(0, 0, 1)`

10.257.5.5 `double gazebo::math::Vector3::x`

X location.

Referenced by `gazebo::physics::DARTTypes::ConvVec3()`, `gazebo::math::Pose::CoordPositionSub()`, `gazebo::math::Matrix3::operator*()`, `operator-`, `gazebo::math::Quaternion::RotateVector()`, `gazebo::physics::SimbodyBoxShape::SetSize()`, and `gazebo::physics::DARTBoxShape::SetSize()`.

10.257.5.6 `double gazebo::math::Vector3::y`

Y location.

Referenced by `gazebo::physics::DARTTypes::ConvVec3()`, `gazebo::math::Pose::CoordPositionSub()`, `gazebo::math::Matrix3::operator*()`, `operator-`, `gazebo::math::Quaternion::RotateVector()`, `gazebo::physics::SimbodyBoxShape::SetSize()`, and `gazebo::physics::DARTBoxShape::SetSize()`.

10.257.5.7 `double gazebo::math::Vector3::z`

Z location.

Referenced by `gazebo::physics::DARTTypes::ConvVec3()`, `gazebo::math::Pose::CoordPositionSub()`, `gazebo::math::Matrix3::operator*()`, `operator-`, `gazebo::math::Quaternion::RotateVector()`, `gazebo::physics::SimbodyBoxShape::SetSize()`, and `gazebo::physics::DARTBoxShape::SetSize()`.

10.257.5.8 `const Vector3 gazebo::math::Vector3::Zero` `[static]`

`math::Vector3(0, 0, 0)`

Referenced by `gazebo::physics::Link::GetWorldLinearVel()`.

The documentation for this class was generated from the following file:

- **Vector3.hh**

10.258 gazebo::math::Vector4 Class Reference

double Generic x, y, z, w vector

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Vector4** ()

Constructor.

- **Vector4** (const double &_x, const double &_y, const double &_z, const double &_w)

Constructor with component values.

- **Vector4** (const **Vector4** &_v)

Copy constructor.

- virtual ~**Vector4** ()

Destructor.

- double **Distance** (const **Vector4** &_pt) const

Calc distance to the given point.

- double **GetLength** () const

Returns the length (magnitude) of the vector.

- double **GetSquaredLength** () const

Return the square of the length (magnitude) of the vector.

- bool **IsFinite** () const

See if a point is finite (e.g., not nan)

- void **Normalize** ()

Normalize the vector length.

- bool **operator!=** (const **Vector4** &_pt) const

Not equal to operator.

- const **Vector4 operator*** (const **Vector4** &_pt) const

Multiplication operator.

- const **Vector4 operator*** (const **Matrix4** &_m) const

Matrix multiplication operator.

- const **Vector4 operator*** (double _v) const

Multiplication operators.

- const **Vector4 & operator*=** (const **Vector4** &_pt)

Multiplication assignment operator.

- const **Vector4 & operator*=** (double _v)

Multiplication assignment operator.

- **Vector4 operator+** (const **Vector4** &_v) const

Addition operator.

- const **Vector4 & operator+=** (const **Vector4** &_v)

Addition operator.

- **Vector4 operator-** (const **Vector4** &_v) const

Subtraction operator.

- const **Vector4 & operator-=** (const **Vector4** &_v)

Subtraction assignment operators.

- const **Vector4 operator/** (const **Vector4** &_v) const

Division assignment operator.

- const **Vector4 operator/** (double _v) const

Division assignment operator.

- const **Vector4 & operator/=** (const **Vector4** &_v)

Division assignment operator.

- const **Vector4 & operator/=** (double _v)

Division operator.

- **Vector4 & operator=** (const **Vector4** &_v)

Assignment operator.

- **Vector4** & **operator=** (double _value)
Assignment operator.
- bool **operator==** (const **Vector4** &_pt) const
Equal to operator.
- double **operator[]** (unsigned int _index) const
Array subscript operator.
- void **Set** (double _x=0, double _y=0, double _z=0, double _w=0)
Set the contents of the vector.

Public Attributes

- double **w**
W value.
- double **x**
X value.
- double **y**
Y value.
- double **z**
Z value.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector4** &_pt)
Stream insertion operator.
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector4** &_pt)
Stream extraction operator.

10.258.1 Detailed Description

double Generic x, y, z, w vector

10.258.2 Constructor & Destructor Documentation

10.258.2.1 gazebo::math::Vector4::Vector4 ()

Constructor.

10.258.2.2 gazebo::math::Vector4::Vector4 (const double & _x, const double & _y, const double & _z, const double & _w)

Constructor with component values.

Parameters

in	_x	value along x axis
in	_y	value along y axis
in	_z	value along z axis
in	_w	value along w axis

10.258.2.3 gazebo::math::Vector4::Vector4 (const Vector4 & _v)

Copy constructor.

Parameters

in	_v	vector
----	----	--------

10.258.2.4 virtual gazebo::math::Vector4::~~Vector4 () [virtual]

Destructor.

10.258.3 Member Function Documentation

10.258.3.1 double gazebo::math::Vector4::Distance (const Vector4 & _pt) const

Calc distance to the given point.

Parameters

in	_pt	the point
----	-----	-----------

Returns

the distance

10.258.3.2 double gazebo::math::Vector4::GetLength () const

Returns the length (magnitude) of the vector.

10.258.3.3 double gazebo::math::Vector4::GetSquaredLength () const

Return the square of the length (magnitude) of the vector.

Returns

the length

10.258.3.4 bool gazebo::math::Vector4::IsFinite () const

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.258.3.5 void gazebo::math::Vector4::Normalize ()

Normalize the vector length.

10.258.3.6 `bool gazebo::math::Vector4::operator!=(const Vector4 & _pt) const`

Not equal to operator.

Parameters

<code>in</code>	<code>_pt</code>	the other vector
-----------------	------------------	------------------

Returns

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.258.3.7 `const Vector4 gazebo::math::Vector4::operator*(const Vector4 & _pt) const`

Multiplication operator.

Remarks

Performs element wise multiplication, which has limited use.

Parameters

<code>in</code>	<code>_pt</code>	another vector
-----------------	------------------	----------------

Returns

result vector

10.258.3.8 `const Vector4 gazebo::math::Vector4::operator*(const Matrix4 & _m) const`

Matrix multiplication operator.

Parameters

<code>in</code>	<code>_m</code>	matrix
-----------------	-----------------	--------

Returns

the vector multiplied by `_m`

10.258.3.9 `const Vector4 gazebo::math::Vector4::operator*(double _v) const`

Multiplication operators.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

a scaled vector

10.258.3.10 `const Vector4& gazebo::math::Vector4::operator*=(const Vector4 & _pt)`

Multiplication assignment operator.

Remarks

Performs element wise multiplication, which has limited use.

Parameters

in	_pt	a vector
----	-----	----------

Returns

this

10.258.3.11 `const Vector4& gazebo::math::Vector4::operator*=(double _v)`

Multiplication assignment operator.

Parameters

in	_v	scaling factor
----	----	----------------

Returns

this

10.258.3.12 `Vector4 gazebo::math::Vector4::operator+(const Vector4 & _v) const`

Addition operator.

Parameters

in	_v	the vector to add
----	----	-------------------

Returns

a sum vector

10.258.3.13 `const Vector4& gazebo::math::Vector4::operator+=(const Vector4 & _v)`

Addition operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to add
-----------------	-----------------	-------------------

Returns

this vector

10.258.3.14 `Vector4 gazebo::math::Vector4::operator- (const Vector4 & _v) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

a vector

10.258.3.15 `const Vector4& gazebo::math::Vector4::operator-= (const Vector4 & _v)`

Subtraction assignment operators.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

this vector

10.258.3.16 `const Vector4 gazebo::math::Vector4::operator/ (const Vector4 & _v) const`

Division assignment operator.

Remarks

Performs element wise division, which has limited use.

Parameters

<code>in</code>	<code>_v</code>	the vector to perform element wise division with
-----------------	-----------------	--

Returns

a result vector

10.258.3.17 `const Vector4 gazebo::math::Vector4::operator/(double _v) const`

Division assignment operator.

Remarks

Performs element wise division, which has limited use.

Parameters

<code>in</code>	<code>_pt</code>	another vector
-----------------	------------------	----------------

Returns

a result vector

10.258.3.18 `const Vector4& gazebo::math::Vector4::operator/= (const Vector4 & _v)`

Division assignment operator.

Remarks

Performs element wise division, which has limited use.

Parameters

<code>in</code>	<code>_v</code>	the vector to perform element wise division with
-----------------	-----------------	--

Returns

this

10.258.3.19 `const Vector4& gazebo::math::Vector4::operator/= (double _v)`

Division operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

a vector

10.258.3.20 `Vector4& gazebo::math::Vector4::operator= (const Vector4 & _v)`

Assignment operator.

Parameters

in	<code>_v</code>	the vector
----	-----------------	------------

Returns

a reference to this vector

10.258.3.21 `Vector4& gazebo::math::Vector4::operator=(double _value)`

Assignment operator.

Parameters

in	<code>_value</code>	
----	---------------------	--

10.258.3.22 `bool gazebo::math::Vector4::operator==(const Vector4 & _pt) const`

Equal to operator.

Parameters

in	<code>_pt</code>	the other vector
----	------------------	------------------

Returns

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.258.3.23 `double gazebo::math::Vector4::operator[](unsigned int _index) const`

Array subscript operator.

Parameters

in	<code>_index</code>	
----	---------------------	--

10.258.3.24 `void gazebo::math::Vector4::Set (double _x = 0, double _y = 0, double _z = 0, double _w = 0)`

Set the contents of the vector.

Parameters

in	<code>_x</code>	value along x axis
in	<code>_y</code>	value along y axis
in	<code>_z</code>	value along z axis
in	<code>_w</code>	value along w axis

10.258.4 Friends And Related Function Documentation

10.258.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector4 & _pt)` [*friend*]

Stream insertion operator.

Parameters

<i>in</i>	<i>_out</i>	output stream
<i>in</i>	<i>_pt</i>	Vector4 (p. 1178) to output

Returns

The stream

10.258.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector4 & _pt)` [*friend*]

Stream extraction operator.

Parameters

<i>in</i>	<i>_in</i>	input stream
<i>in</i>	<i>_pt</i>	Vector4 (p. 1178) to read values into

Returns

the stream

10.258.5 Member Data Documentation

10.258.5.1 `double gazebo::math::Vector4::w`

W value.

10.258.5.2 `double gazebo::math::Vector4::x`

X value.

10.258.5.3 `double gazebo::math::Vector4::y`

Y value.

10.258.5.4 `double gazebo::math::Vector4::z`

Z value.

The documentation for this class was generated from the following file:

- **Vector4.hh**

10.259 gazebo::common::Video Class Reference

Handle video encoding and decoding using libavcodec.

```
#include <common/common.hh>
```

Public Member Functions

- **Video** ()
Constructor.
- virtual **~Video** ()
Destructor.
- int **GetHeight** () const
Get the height of the video in pixels.
- bool **GetNextFrame** (unsigned char **_buffer)
Get the next frame of the video.
- int **GetWidth** () const
Get the width of the video in pixels.
- bool **Load** (const std::string &_filename)
Load a video file.

10.259.1 Detailed Description

Handle video encoding and decoding using libavcodec.

10.259.2 Constructor & Destructor Documentation

10.259.2.1 gazebo::common::Video::Video ()

Constructor.

10.259.2.2 virtual gazebo::common::Video::~~Video () [virtual]

Destructor.

10.259.3 Member Function Documentation

10.259.3.1 int gazebo::common::Video::GetHeight () const

Get the height of the video in pixels.

Returns

the height

10.259.3.2 `bool gazebo::common::Video::GetNextFrame (unsigned char ** _buffer)`

Get the next frame of the video.

Parameters

<code>out</code>	<code><i>_img</i></code>	Image (p. 515) in which the frame is stored
------------------	--------------------------	--

Returns

false if HAVE_FFmpeg is not defined, true otherwise

10.259.3.3 `int gazebo::common::Video::GetWidth () const`

Get the width of the video in pixels.

Returns

the width

10.259.3.4 `bool gazebo::common::Video::Load (const std::string & _filename)`

Load a video file.

Parameters

<code>in</code>	<code><i>_filename</i></code>	Full path of the video file
-----------------	-------------------------------	-----------------------------

Returns

false if HAVE_FFmpeg is not defined or if a video stream can't be found

The documentation for this class was generated from the following file:

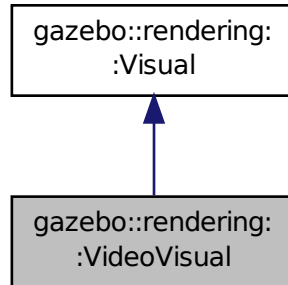
- **Video.hh**

10.260 gazebo::rendering::VideoVisual Class Reference

A visual element that displays a video as a texture.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::VideoVisual:



Public Member Functions

- **VideoVisual** (const std::string &_name, **VisualPtr** _parent)
Constructor.
- virtual ~**VideoVisual** ()
Destructor.

Additional Inherited Members

10.260.1 Detailed Description

A visual element that displays a video as a texture.

10.260.2 Constructor & Destructor Documentation

10.260.2.1 gazebo::rendering::VideoVisual::VideoVisual (const std::string & _name, VisualPtr _parent)

Constructor.

Parameters

in	<i>_name</i>	Name of the video visual.
in	<i>_parent</i>	Parent of the video visual.

10.260.2.2 virtual gazebo::rendering::VideoVisual::~~VideoVisual () [virtual]

Destructor.

The documentation for this class was generated from the following file:

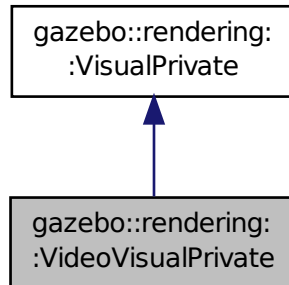
- **VideoVisual.hh**

10.261 gazebo::rendering::VideoVisualPrivate Class Reference

Private data for the Video **Visual** (p. 1196) class.

```
#include <VideoVisualPrivate.hh>
```

Inheritance diagram for gazebo::rendering::VideoVisualPrivate:



Public Attributes

- `std::vector< event::ConnectionPtr > connections`
All the event connections.
- `int height`
Height of the video.
- `unsigned char * imageBuffer`
One frame of the video.
- `Ogre::TexturePtr texture`
Texture to draw the video onto.
- `common::Video * video`
Load a video.
- `int width`
Width of the video.

Additional Inherited Members

10.261.1 Detailed Description

Private data for the Video **Visual** (p. 1196) class.

10.261.2 Member Data Documentation

10.261.2.1 `std::vector<event::ConnectionPtr>` gazebo::rendering::VideoVisualPrivate::connections

All the event connections.

10.261.2.2 `int` gazebo::rendering::VideoVisualPrivate::height

Height of the video.

10.261.2.3 `unsigned char*` gazebo::rendering::VideoVisualPrivate::imageBuffer

One frame of the video.

10.261.2.4 `Ogre::TexturePtr` gazebo::rendering::VideoVisualPrivate::texture

Texture to draw the video onto.

10.261.2.5 `common::Video*` gazebo::rendering::VideoVisualPrivate::video

Load a video.

10.261.2.6 `int` gazebo::rendering::VideoVisualPrivate::width

Width of the video.

The documentation for this class was generated from the following file:

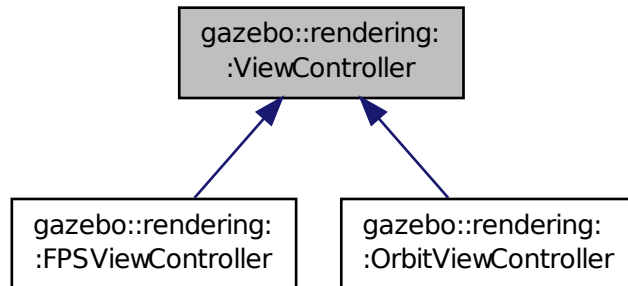
- **VideoVisualPrivate.hh**

10.262 gazebo::rendering::ViewController Class Reference

Base class for view controllers.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::ViewController:



Public Member Functions

- **ViewController** (**UserCameraPtr** _camera)
Constructor.
- virtual **~ViewController** ()
Destructor.
- std::string **GetTypeString** () const
Get the type of view controller.
- virtual void **HandleKeyPressEvent** (const std::string &_key)=0
Handle a key press event.
- virtual void **HandleKeyReleaseEvent** (const std::string &_key)=0
Handle a key release event.
- virtual void **HandleMouseEvent** (const **common::MouseEvent** &_event)=0
Handle a mouse event.
- virtual void **Init** ()=0
Initialize the view controller.
- virtual void **Init** (const **math::Vector3** &_focalPoint)
Initialize with a focus point.
- void **SetEnabled** (bool _value)
Set whether the controller is enabled.
- virtual void **Update** ()=0
*Update the controller, which should update the position of the **Camera** (p. 197).*

Protected Attributes

- **UserCameraPtr** camera
Pointer to the camera to control.
- bool **enabled**
True if enabled.
- std::string **typeString**
Type of view controller.

10.262.1 Detailed Description

Base class for view controllers.

10.262.2 Constructor & Destructor Documentation

10.262.2.1 gazebo::rendering::ViewController::ViewController (UserCameraPtr *_camera*)

Constructor.

Parameters

in	<i>_camera</i>	The user camera to controll.
----	----------------	------------------------------

10.262.2.2 virtual gazebo::rendering::ViewController::~~ViewController () [virtual]

Destructor.

10.262.3 Member Function Documentation

10.262.3.1 std::string gazebo::rendering::ViewController::GetTypeString () const

Get the type of view controller.

Returns

The view controller type string.

10.262.3.2 virtual void gazebo::rendering::ViewController::HandleKeyPressEvent (const std::string & *_key*) [pure virtual]

Handle a key press event.

Parameters

in	<i>_key</i>	The key that was pressed.
----	-------------	---------------------------

Implemented in **gazebo::rendering::OrbitViewController** (p. 765), and **gazebo::rendering::FPSViewController** (p. 454).

10.262.3.3 virtual void gazebo::rendering::ViewController::HandleKeyReleaseEvent (const std::string & *_key*) [pure virtual]

Handle a key release event.

Parameters

in	<i>_key</i>	The key that was released.
----	-------------	----------------------------

Implemented in **gazebo::rendering::OrbitViewController** (p. 765), and **gazebo::rendering::FPSViewController** (p. 455).

10.262.3.4 `virtual void gazebo::rendering::ViewController::HandleMouseEvent (const common::MouseEvent & _event)` [pure virtual]

Handle a mouse event.

Parameters

in	<code>_event</code>	The mouse position.
----	---------------------	---------------------

Implemented in **gazebo::rendering::OrbitViewController** (p. 765), and **gazebo::rendering::FPSViewController** (p. 455).

10.262.3.5 `virtual void gazebo::rendering::ViewController::Init ()` [pure virtual]

Initialize the view controller.

Implemented in **gazebo::rendering::OrbitViewController** (p. 765), and **gazebo::rendering::FPSViewController** (p. 455).

10.262.3.6 `virtual void gazebo::rendering::ViewController::Init (const math::Vector3 & _focalPoint)` [virtual]

Initialize with a focus point.

Parameters

in	<code>_focalPoint</code>	The point to look at.
----	--------------------------	-----------------------

Reimplemented in **gazebo::rendering::OrbitViewController** (p. 765).

10.262.3.7 `void gazebo::rendering::ViewController::SetEnabled (bool _value)`

Set whether the controller is enabled.

Parameters

in	<code>_value</code>	True if the controller is enabled.
----	---------------------	------------------------------------

10.262.3.8 `virtual void gazebo::rendering::ViewController::Update ()` [pure virtual]

Update the controller, which should update the position of the **Camera** (p. 197).

Implemented in **gazebo::rendering::OrbitViewController** (p. 766), and **gazebo::rendering::FPSViewController** (p. 455).

10.262.4 Member Data Documentation

10.262.4.1 `UserCameraPtr gazebo::rendering::ViewController::camera` [protected]

Pointer to the camera to control.

10.262.4.2 `bool gazebo::rendering::ViewController::enabled` [protected]

True if enabled.

10.262.4.3 `std::string gazebo::rendering::ViewController::typeString` [protected]

Type of view controller.

The documentation for this class was generated from the following file:

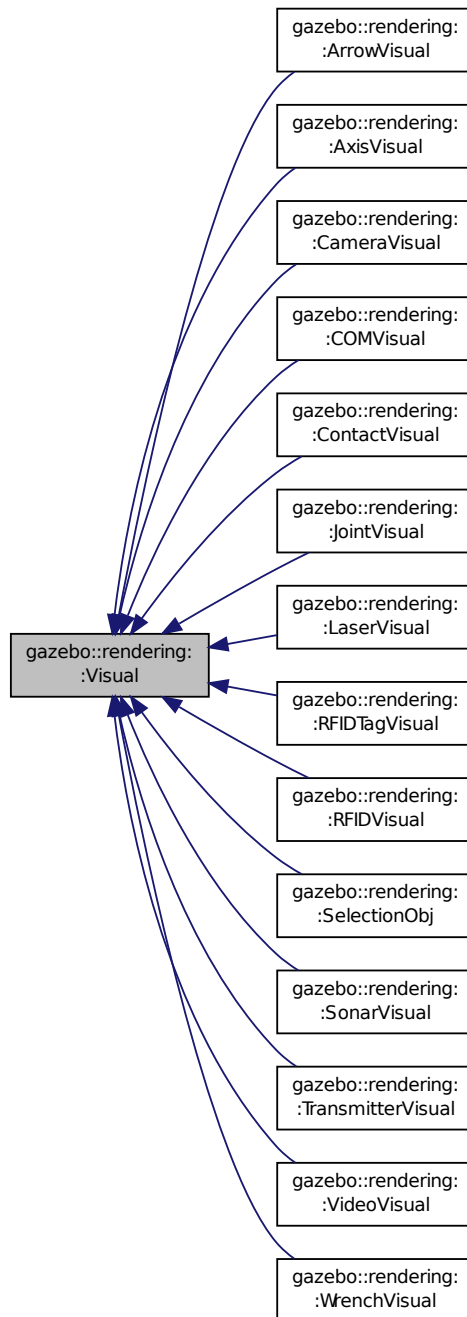
- `ViewController.hh`

10.263 gazebo::rendering::Visual Class Reference

A renderable object.

```
#include <rendering/rendering.hh>
```


Inheritance diagram for gazebo::rendering::Visual:



Public Member Functions

- **Visual** (const std::string &_name, **VisualPtr** _parent, bool _useRTShader=true)

Constructor.

- **Visual** (const std::string &_name, **ScenePtr** _scene, bool _useRTShader=true)

Constructor.

- virtual ~**Visual** ()

Destructor.

- void **AttachAxes** ()

Attach visualization axes.

- void **AttachLineVertex** (**DynamicLines** *_line, unsigned int _index)

Attach a vertex of a line to the position of the visual.

- Ogre::MovableObject * **AttachMesh** (const std::string &_meshName, const std::string &_subMesh="", bool _centerSubmesh=false, const std::string &_objName="")

Attach a mesh to this visual by name.

- void **AttachObject** (Ogre::MovableObject *_obj)

Attach a reusable object to the visual.

- void **AttachVisual** (**VisualPtr** _vis)

Attach a visual to this visual.

- void **ClearParent** ()

Clear parents.

- **VisualPtr** **Clone** (const std::string &_name, **VisualPtr** _newParent)

Clone the visual with a new name.

- **DynamicLines** * **CreateDynamicLine** (**RenderOpType** _type=RENDERING_LINE_STRIP)

Add a line to the visual.

- void **DeleteDynamicLine** (**DynamicLines** *_line)

Delete a dynamic line.

- void **DetachObjects** ()

Detach all objects.

- void **DetachVisual** (**VisualPtr** _vis)

Detach a visual.

- void **DetachVisual** (const std::string &_name)

Detach a visual.

- void **DisableTrackVisual** ()

Disable tracking of a visual.

- void **EnableTrackVisual** (**VisualPtr** _vis)

Set one visual to track/follow another.

- void **Fini** ()

Helper for the destructor.

- unsigned int **GetAttachedObjectCount** () const

Return the number of attached movable objects.

- **math::Box** **GetBoundingBox** () const

Get the bounding box for the visual.

- **VisualPtr** **GetChild** (unsigned int _index)

Get an attached visual based on an index.

- unsigned int **GetChildCount** ()

Get the number of attached visuals.

- bool **GetHighlighted** () const

Get whether or not the visual is visually highlighted.

- uint32_t **GetId** () const

- Get the id associated with this visual.*

 - `std::string GetMaterialName ()` const

Get the name of the material.
- `std::string GetMeshName ()` const

The name of the mesh set in the visual's SDF.
- `std::string GetName ()` const

Get the name of the visual.
- `std::string GetNormalMap ()` const

Get the normal map.
- `VisualPtr GetParent ()` const

Get the parent visual, if one exists.
- `math::Pose GetPose ()` const

Get the pose of the visual.
- `math::Vector3 GetPosition ()` const

Get the position of the visual.
- `VisualPtr GetRootVisual ()`

Get the root visual.
- `math::Quaternion GetRotation ()` const

Get the rotation of the visual.
- `math::Vector3 GetScale ()`

Get the scale.
- `ScenePtr GetScene ()` const

Get current.
- `Ogre::SceneNode * GetSceneNode ()` const

Return the scene Node of this visual entity.
- `std::string GetShaderType ()` const

Get the shader type.
- `std::string GetSubMeshName ()` const

Get the name of the sub mesh set in the visual's SDF.
- `float GetTransparency ()`

Get the transparency.
- `uint32_t GetVisibilityFlags ()`

Get visibility flags for this visual and all children.
- `bool GetVisible ()` const

Get whether the visual is visible.
- `math::Pose GetWorldPose ()` const

Get the global pose of the node.
- `bool HasAttachedObject (const std::string &_name)`

Returns true if an object with _name is attached.
- `void Init ()`

Helper for the constructor.
- `void InsertMesh (const std::string &_meshName, const std::string &_subMesh="", bool _centerSubmesh=false)`

*Insert a mesh into **Ogre** (p. 137).*
- `bool IsPlane ()` const

Return true if the visual is a plane.
- `bool IsStatic ()` const

Return true if the visual is a static geometry.

- void **Load** (sdf::ElementPtr _sdf)
Load the visual with a set of parameters.
- virtual void **Load** ()
Load the visual with default parameters.
- void **LoadFromMsg** (ConstVisualPtr &_msg)
Load from a message.
- void **LoadPlugin** (const std::string &_filename, const std::string &_name, sdf::ElementPtr _sdf)
Load a plugin.
- void **MakeStatic** ()
Make the visual objects static renderables.
- void **MoveToPosition** (const **math::Pose** &_pose, double _time)
Move to a pose and over a given time.
- void **MoveToPositions** (const std::vector< **math::Pose** > &_pts, double _time, boost::function< void()> _on-Complete=NULL)
Move to a series of pose and over a given time.
- void **RemovePlugin** (const std::string &_name)
Remove a running plugin.
- void **SetAmbient** (const **common::Color** &_color)
Set the ambient color of the visual.
- void **SetCastShadows** (bool _shadows)
Set whether the visual should cast shadows.
- void **SetDiffuse** (const **common::Color** &_color)
Set the diffuse color of the visual.
- virtual void **SetEmissive** (const **common::Color** &_color)
Set the emissive value.
- void **SetHighlighted** (bool _highlighted)
Set the visual to be visually highlighted.
- void **SetId** (uint32_t _id)
Set the id associated with this visual.
- void **SetLighting** (bool _lighting)
Set whether or not to enable or disable lighting.
- void **SetMaterial** (const std::string &_materialName, bool _unique=true)
Set the material.
- void **SetName** (const std::string &_name)
Set the name of the visual.
- void **SetNormalMap** (const std::string &_nmap)
Set the normal map.
- void **SetPose** (const **math::Pose** &_pose)
Set the pose of the visual.
- void **SetPosition** (const **math::Vector3** &_pos)
Set the position of the visual.
- void **SetRibbonTrail** (bool _value, const **common::Color** &_initialColor, const **common::Color** &_change-Color)
True on or off a ribbon trail.
- void **SetRotation** (const **math::Quaternion** &_rot)
Set the rotation of the visual.
- void **SetScale** (const **math::Vector3** &_scale)

- Set the scale.*

 - void **SetScene** (**ScenePtr** _scene)
 - Set current scene.*
 - void **SetShaderType** (const std::string &_type)
 - Set the shader type for the visual's material.*
 - void **SetSkeletonPose** (const msgs::PoseAnimation &_pose)
 - Set animation skeleton pose.*
 - void **SetSpecular** (const **common::Color** &_color)
 - Set the specular color of the visual.*
 - void **SetTransparency** (float _trans)
 - Set the transparency.*
 - void **SetVisibilityFlags** (uint32_t _flags)
 - Set visibility flags for this visual and all children.*
 - void **SetVisible** (bool _visible, bool _cascade=true)
 - Set whether the visual is visible.*
 - void **SetWireframe** (bool _show)
 - Enable or disable wireframe for this visual.*
 - void **SetWorldPose** (const **math::Pose** &_pose)
 - Set the world pose of the visual.*
 - void **SetWorldPosition** (const **math::Vector3** &_pos)
 - Set the world linear position of the visual.*
 - void **SetWorldRotation** (const **math::Quaternion** &_rot)
 - Set the world orientation of the visual.*
 - void **ShowBoundingBox** ()
 - Display the bounding box visual.*
 - void **ShowCollision** (bool _show)
 - Display the collision visuals.*
 - void **ShowCOM** (bool _show)
 - Display Center of Mass visuals.*
 - void **ShowJoints** (bool _show)
 - Display joint visuals.*
 - void **ShowSkeleton** (bool _show)
 - Display the skeleton visuals.*
 - void **ToggleVisible** ()
 - Toggle whether this visual is visible.*
 - void **Update** ()
 - Update the visual.*
 - void **UpdateFromMsg** (ConstVisualPtr &_msg)
 - Update a visual based on a message.*

Static Public Member Functions

- static void **InsertMesh** (const **common::Mesh** *_mesh, const std::string &_subMesh="", bool _center-Submesh=false)
 - Insert a mesh into **Ogre** (p. 137).*

Protected Member Functions

- **Visual** (**VisualPrivate** &_dataPtr, const std::string &_name, **VisualPtr** _parent, bool _useRTShader=true)
- **Visual** (**VisualPrivate** &_dataPtr, const std::string &_name, **ScenePtr** _scene, bool _useRTShader=true)

Protected Attributes

- **VisualPrivate** * dataPtr

10.263.1 Detailed Description

A renderable object.

10.263.2 Constructor & Destructor Documentation

10.263.2.1 gazebo::rendering::Visual::Visual (const std::string & _name, VisualPtr _parent, bool _useRTShader = true)

Constructor.

Parameters

in	<code>_name</code>	Name of the visual.
in	<code>_parent</code>	Parent of the visual.
in	<code>_useRTShader</code>	True if the visual should use the real-time shader system (RTShader).

10.263.2.2 gazebo::rendering::Visual::Visual (const std::string & _name, ScenePtr _scene, bool _useRTShader = true)

Constructor.

Parameters

in	<code>_name</code>	Name of the visual.
in	<code>_scene</code>	Scene (p. 879) containing the visual.
in	<code>_useRTShader</code>	True if the visual should use the real-time shader system (RTShader).

10.263.2.3 virtual gazebo::rendering::Visual::~Visual () [virtual]

Destructor.

10.263.2.4 gazebo::rendering::Visual::Visual (VisualPrivate & _dataPtr, const std::string & _name, VisualPtr _parent, bool _useRTShader = true) [protected]

10.263.2.5 gazebo::rendering::Visual::Visual (VisualPrivate & _dataPtr, const std::string & _name, ScenePtr _scene, bool _useRTShader = true) [protected]

10.263.3 Member Function Documentation

10.263.3.1 void gazebo::rendering::Visual::AttachAxes ()

Attach visualization axes.

10.263.3.2 void gazebo::rendering::Visual::AttachLineVertex (*DynamicLines* * *_line*, unsigned int *_index*)

Attach a vertex of a line to the position of the visual.

Parameters

in	<i>_line</i>	Line to attach to this visual.
in	<i>_index</i>	Index of the line vertex to attach.

10.263.3.3 *Ogre::MovableObject** gazebo::rendering::Visual::AttachMesh (const std::string & *_meshName*, const std::string & *_subMesh* = " ", bool *_centerSubmesh* = false, const std::string & *_objName* = " ")

Attach a mesh to this visual by name.

Parameters

in	<i>_meshName</i>	Name of the mesh.
in	<i>_subMesh</i>	Name of the submesh. Empty string to use all submeshes.
in	<i>_centerSubmesh</i>	True to center a submesh.
in	<i>_objName</i>	Name of the attached Object to put the mesh onto.

10.263.3.4 void gazebo::rendering::Visual::AttachObject (*Ogre::MovableObject* * *_obj*)

Attach a renewable object to the visual.

Parameters

in	<i>_obj</i>	A movable object to attach to the visual.
----	-------------	---

10.263.3.5 void gazebo::rendering::Visual::AttachVisual (*VisualPtr* *_vis*)

Attach a visual to this visual.

Parameters

in	<i>_vis</i>	Visual (p. 1196) to attach.
----	-------------	------------------------------------

10.263.3.6 void gazebo::rendering::Visual::ClearParent ()

Clear parents.

10.263.3.7 *VisualPtr* gazebo::rendering::Visual::Clone (const std::string & *_name*, *VisualPtr* *_newParent*)

Clone the visual with a new name.

Parameters

in	<code>_name</code>	Name of the cloned Visual (p. 1196).
in	<code>_newParent</code>	Parent of the cloned Visual (p. 1196).

Returns

The visual.

10.263.3.8 `DynamicLines* gazebo::rendering::Visual::CreateDynamicLine (RenderOpType _type = RENDERING_LINE_STRIP)`

Add a line to the visual.

Parameters

in	<code>_type</code>	The type of line to make.
----	--------------------	---------------------------

Returns

A pointer to the new dynamic line.

10.263.3.9 `void gazebo::rendering::Visual::DeleteDynamicLine (DynamicLines * _line)`

Delete a dynamic line.

Parameters

in	<code>_line</code>	Pointer to the line to delete.
----	--------------------	--------------------------------

10.263.3.10 `void gazebo::rendering::Visual::DetachObjects ()`

Detach all objects.

10.263.3.11 `void gazebo::rendering::Visual::DetachVisual (VisualPtr _vis)`

Detach a visual.

Parameters

in	<code>_vis</code>	Visual (p. 1196) to detach.
----	-------------------	------------------------------------

10.263.3.12 `void gazebo::rendering::Visual::DetachVisual (const std::string & _name)`

Detach a visual.

Parameters

in	<code>_name</code>	Name of the visual to detach.
----	--------------------	-------------------------------

10.263.3.13 void gazebo::rendering::Visual::DisableTrackVisual ()

Disable tracking of a visual.

10.263.3.14 void gazebo::rendering::Visual::EnableTrackVisual (VisualPtr _vis)

Set one visual to track/follow another.

Parameters

in	_vis	Visual (p. 1196) to track.
----	------	----------------------------

10.263.3.15 void gazebo::rendering::Visual::Fini ()

Helper for the destructor.

10.263.3.16 unsigned int gazebo::rendering::Visual::GetAttachedObjectCount () const

Return the number of attached movable objects.

Returns

The number of attached movable objects.

10.263.3.17 math::Box gazebo::rendering::Visual::GetBoundingBox () const

Get the bounding box for the visual.

Returns

The bounding box in world coordinates.

10.263.3.18 VisualPtr gazebo::rendering::Visual::GetChild (unsigned int _index)

Get an attached visual based on an index.

Index should be between 0 and **Visual::GetChildCount** (p. 1205).

Parameters

in	_index	Index of the child to retrieve.
----	--------	---------------------------------

Returns

Pointer to the child visual, NULL if index is invalid.

10.263.3.19 unsigned int gazebo::rendering::Visual::GetChildCount ()

Get the number of attached visuals.

Returns

The number of children.

10.263.3.20 `bool gazebo::rendering::Visual::GetHighlighted () const`

Get whether or not the visual is visually highlighted.

This is most often means that an object is selected by a user via the GUI.

Returns

True if the visual is highlighted.

10.263.3.21 `uint32_t gazebo::rendering::Visual::GetId () const`

Get the id associated with this visual.

10.263.3.22 `std::string gazebo::rendering::Visual::GetMaterialName () const`

Get the name of the material.

Returns

The name of the visual applied to this visual.

10.263.3.23 `std::string gazebo::rendering::Visual::GetMeshName () const`

The name of the mesh set in the visual's SDF.

Returns

Name of the mesh.

10.263.3.24 `std::string gazebo::rendering::Visual::GetName () const`

Get the name of the visual.

Returns

The name of the visual.

10.263.3.25 `std::string gazebo::rendering::Visual::GetNormalMap () const`

Get the normal map.

Returns

The name of the normal map material.

10.263.3.26 `VisualPtr gazebo::rendering::Visual::GetParent () const`

Get the parent visual, if one exists.

Returns

Pointer to the parent visual, NULL if no parent.

10.263.3.27 `math::Pose gazebo::rendering::Visual::GetPose () const`

Get the pose of the visual.

Returns

The **Visual** (p. 1196)'s pose.

10.263.3.28 `math::Vector3 gazebo::rendering::Visual::GetPosition () const`

Get the position of the visual.

Returns

The visual's position.

10.263.3.29 `VisualPtr gazebo::rendering::Visual::GetRootVisual ()`

Get the root visual.

Returns

The root visual, which is one level below the world visual.

10.263.3.30 `math::Quaternion gazebo::rendering::Visual::GetRotation () const`

Get the rotation of the visual.

Returns

The visual's rotation.

10.263.3.31 `math::Vector3 gazebo::rendering::Visual::GetScale ()`

Get the scale.

Returns

The scaling factor.

10.263.3.32 `ScenePtr gazebo::rendering::Visual::GetScene () const`

Get current.

Returns

Pointer to the scene.

10.263.3.33 `Ogre::SceneNode* gazebo::rendering::Visual::GetSceneNode () const`

Return the scene Node of this visual entity.

Returns

The **Ogre** (p. 137) scene node.

10.263.3.34 `std::string gazebo::rendering::Visual::GetShaderType () const`

Get the shader type.

Returns

String of the shader type: "vertex", "pixel", "normal_map_object_space", "normal_map_tangent_space".

10.263.3.35 `std::string gazebo::rendering::Visual::GetSubMeshName () const`

Get the name of the sub mesh set in the visual's SDF.

Returns

Name of the submesh. Empty string if no submesh is specified.

10.263.3.36 `float gazebo::rendering::Visual::GetTransparency ()`

Get the transparency.

Returns

The transparency.

10.263.3.37 `uint32_t gazebo::rendering::Visual::GetVisibilityFlags ()`

Get visibility flags for this visual and all children.

Returns

The visibility flags.

See Also

GZ_VISIBILITY_ALL (p. 1430)

GZ_VISIBILITY_GUI (p. 1430)

GZ_VISIBILITY_SELECTABLE (p. 1430)

10.263.3.38 `bool gazebo::rendering::Visual::GetVisible () const`

Get whether the visual is visible.

Returns

True if the visual is visible.

10.263.3.39 `math::Pose gazebo::rendering::Visual::GetWorldPose () const`

Get the global pose of the node.

Returns

The pose in the world coordinate frame.

10.263.3.40 `bool gazebo::rendering::Visual::HasAttachedObject (const std::string & _name)`

Returns true if an object with `_name` is attached.

Parameters

in	<code>_name</code>	Name of an object to find.
----	--------------------	----------------------------

10.263.3.41 `void gazebo::rendering::Visual::Init ()`

Helper for the constructor.

10.263.3.42 `void gazebo::rendering::Visual::InsertMesh (const std::string & _meshName, const std::string & _subMesh = "", bool _centerSubmesh = false)`

Insert a mesh into **Ogre** (p. 137).

Parameters

in	<code>_meshName</code>	Name of the mesh to insert.
in	<code>_subMesh</code>	Name of the mesh within <code>_meshName</code> to insert.
in	<code>_centerSubmesh</code>	True to center the submesh.

10.263.3.43 `static void gazebo::rendering::Visual::InsertMesh (const common::Mesh * _mesh, const std::string & _subMesh = "", bool _centerSubmesh = false) [static]`

Insert a mesh into **Ogre** (p. 137).

Parameters

in	<code>_mesh</code>	Pointer to the mesh to insert.
in	<code>_subMesh</code>	Name of the mesh within <code>_meshName</code> to insert.
in	<code>_centerSubmesh</code>	True to center the submesh.

10.263.3.44 `bool gazebo::rendering::Visual::IsPlane () const`

Return true if the visual is a plane.

Returns

True if a plane.

10.263.3.45 `bool gazebo::rendering::Visual::IsStatic () const`

Return true if the visual is a static geometry.

Returns

True if the visual is static.

10.263.3.46 `void gazebo::rendering::Visual::Load (sdf::ElementPtr _sdf)`

Load the visual with a set of parameters.

Parameters

in	_sdf	Load from an SDF element.
----	------	---------------------------

Reimplemented in **gazebo::rendering::COMVisual** (p. 262).

10.263.3.47 `virtual void gazebo::rendering::Visual::Load () [virtual]`

Load the visual with default parameters.

Reimplemented in **gazebo::rendering::SelectionObj** (p. 902), **gazebo::rendering::SonarVisual** (p. 1052), **gazebo::rendering::TransmitterVisual** (p. 1132), **gazebo::rendering::AxisVisual** (p. 164), and **gazebo::rendering::ArrowVisual** (p. 159).

10.263.3.48 `void gazebo::rendering::Visual::LoadFromMsg (ConstVisualPtr & _msg)`

Load from a message.

Parameters

in	_msg	A visual message.
----	------	-------------------

10.263.3.49 `void gazebo::rendering::Visual::LoadPlugin (const std::string & _filename, const std::string & _name, sdf::ElementPtr _sdf)`

Load a plugin.

Parameters

<code>_filename</code>	The filename of the plugin
<code>_name</code>	A unique name for the plugin
<code>_sdf</code>	The SDF to pass into the plugin.

10.263.3.50 `void gazebo::rendering::Visual::MakeStatic ()`

Make the visual objects static renderables.

10.263.3.51 `void gazebo::rendering::Visual::MoveToPosition (const math::Pose & _pose, double _time)`

Move to a pose and over a given time.

Parameters

<code>in</code>	<code>_pose</code>	Pose the visual will end at.
<code>in</code>	<code>_time</code>	Time it takes the visual to move to the pose.

10.263.3.52 `void gazebo::rendering::Visual::MoveToPositions (const std::vector< math::Pose > & _pts, double _time, boost::function< void()> _onComplete = NULL)`

Move to a series of pose and over a given time.

Parameters

<code>in</code>	<code>_poses</code>	Series of poses the visual will move to.
<code>in</code>	<code>_time</code>	Time it takes the visual to move to the pose.
<code>in</code>	<code>_onComplete</code>	Callback used when the move is complete.

10.263.3.53 `void gazebo::rendering::Visual::RemovePlugin (const std::string & _name)`

Remove a running plugin.

Parameters

<code>_name</code>	The unique name of the plugin to remove
--------------------	---

10.263.3.54 `void gazebo::rendering::Visual::SetAmbient (const common::Color & _color)`

Set the ambient color of the visual.

Parameters

<code>in</code>	<code>_color</code>	The ambient color.
-----------------	---------------------	--------------------

10.263.3.55 `void gazebo::rendering::Visual::SetCastShadows (bool _shadows)`

Set whether the visual should cast shadows.

Parameters

<code>in</code>	<code><i>_shadows</i></code>	True to enable shadows.
-----------------	------------------------------	-------------------------

10.263.3.56 `void gazebo::rendering::Visual::SetDiffuse (const common::Color & _color)`

Set the diffuse color of the visual.

Parameters

<code>in</code>	<code><i>_color</i></code>	Set the diffuse color.
-----------------	----------------------------	------------------------

10.263.3.57 `virtual void gazebo::rendering::Visual::SetEmissive (const common::Color & _color)` [virtual]

Set the emissive value.

Parameters

<code>in</code>	<code><i>_color</i></code>	The emissive color.
-----------------	----------------------------	---------------------

Reimplemented in `gazebo::rendering::LaserVisual` (p. 587).

10.263.3.58 `void gazebo::rendering::Visual::SetHighlighted (bool _highlighted)`

Set the visual to be visually highlighted.

This is most often used when an object is selected by a user via the GUI.

Parameters

<code>in</code>	<code><i>_highlighted</i></code>	True to enable the highlighting.
-----------------	----------------------------------	----------------------------------

10.263.3.59 `void gazebo::rendering::Visual::SetId (uint32_t _id)`

Set the id associated with this visual.

10.263.3.60 `void gazebo::rendering::Visual::SetLighting (bool _lighting)`

Set whether or not to enable or disable lighting.

Parameters

<code>in</code>	<code><i>_lighting</i></code>	True to enable lighting.
-----------------	-------------------------------	--------------------------

10.263.3.61 void gazebo::rendering::Visual::SetMaterial (const std::string & *_materialName*, bool *_unique* = true)

Set the material.

Parameters

in	<i>_materialName</i>	The name of the material.
in	<i>_unique</i>	True to make the material unique, which allows the material to change without changing materials that originally had the same name.

10.263.3.62 void gazebo::rendering::Visual::SetName (const std::string & *_name*)

Set the name of the visual.

Parameters

in	<i>_name</i>	Name of the visual
----	--------------	--------------------

10.263.3.63 void gazebo::rendering::Visual::SetNormalMap (const std::string & *_nmap*)

Set the normal map.

Parameters

in	<i>_nmap</i>	Name of the normal map material.
----	--------------	----------------------------------

10.263.3.64 void gazebo::rendering::Visual::SetPose (const math::Pose & *_pose*)

Set the pose of the visual.

Parameters

in	<i>_pose</i>	The new pose of the visual.
----	--------------	-----------------------------

10.263.3.65 void gazebo::rendering::Visual::SetPosition (const math::Vector3 & *_pos*)

Set the position of the visual.

Parameters

in	<i>_pos</i>	The position to set the visual to.
----	-------------	------------------------------------

10.263.3.66 void gazebo::rendering::Visual::SetRibbonTrail (bool *_value*, const common::Color & *_initialColor*, const common::Color & *_changeColor*)

True on or off a ribbon trail.

Parameters

in	<code>_value</code>	True to enable ribbon trail.
in	<code>_initialColor</code>	The initial color of the ribbon trail.
in	<code>_changeColor</code>	Color to change too as the trail grows.

10.263.3.67 `void gazebo::rendering::Visual::SetRotation (const math::Quaternion & _rot)`

Set the rotation of the visual.

Parameters

in	<code>_rot</code>	The rotation of the visual.
----	-------------------	-----------------------------

10.263.3.68 `void gazebo::rendering::Visual::SetScale (const math::Vector3 & _scale)`

Set the scale.

Parameters

in	<code>_scale</code>	The scaling factor for the visual.
----	---------------------	------------------------------------

10.263.3.69 `void gazebo::rendering::Visual::SetScene (ScenePtr _scene)`

Set current scene.

Parameters

in	<code>_scene</code>	Pointer to the scene.
----	---------------------	-----------------------

10.263.3.70 `void gazebo::rendering::Visual::SetShaderType (const std::string & _type)`

Set the shader type for the visual's material.

Parameters

in	<code>_type</code>	Shader type string: "vertex", "pixel", "normal_map_object_space", "normal_map_tangent_space".
----	--------------------	---

10.263.3.71 `void gazebo::rendering::Visual::SetSkeletonPose (const msgs::PoseAnimation & _pose)`

Set animation skeleton pose.

Parameters

in	<code>_pose</code>	Skelton message
----	--------------------	-----------------

10.263.3.72 `void gazebo::rendering::Visual::SetSpecular (const common::Color & _color)`

Set the specular color of the visual.

Parameters

<code>in</code>	<code><i>_color</i></code>	Specular color.
-----------------	----------------------------	-----------------

10.263.3.73 `void gazebo::rendering::Visual::SetTransparency (float _trans)`

Set the transparency.

Parameters

<code>in</code>	<code><i>_trans</i></code>	The transparency, between 0 and 1 where 0 is no transparency.
-----------------	----------------------------	---

10.263.3.74 `void gazebo::rendering::Visual::SetVisibilityFlags (uint32_t _flags)`

Set visibility flags for this visual and all children.

Parameters

<code>in</code>	<code><i>_flags</i></code>	The visibility flags.
-----------------	----------------------------	-----------------------

See Also

GZ_VISIBILITY_ALL (p. 1430)

GZ_VISIBILITY_GUI (p. 1430)

GZ_VISIBILITY_SELECTABLE (p. 1430)

10.263.3.75 `void gazebo::rendering::Visual::SetVisible (bool _visible, bool _cascade = true)`

Set whether the visual is visible.

Parameters

<code>in</code>	<code><i>_visible</i></code>	set this node visible.
<code>in</code>	<code><i>_cascade</i></code>	setting this parameter in children too.

10.263.3.76 `void gazebo::rendering::Visual::SetWireframe (bool _show)`

Enable or disable wireframe for this visual.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to enable wireframe for this visual.
-----------------	---------------------------	---

10.263.3.77 `void gazebo::rendering::Visual::SetWorldPose (const math::Pose & _pose)`

Set the world pose of the visual.

Parameters

<code>in</code>	<code><i>_pose</i></code>	Pose of the visual in the world coordinate frame.
-----------------	---------------------------	---

10.263.3.78 `void gazebo::rendering::Visual::SetWorldPosition (const math::Vector3 & _pos)`

Set the world linear position of the visual.

Parameters

<code>in</code>	<code><i>_pose</i></code>	Position in the world coordinate frame.
-----------------	---------------------------	---

10.263.3.79 `void gazebo::rendering::Visual::SetWorldRotation (const math::Quaternion & _rot)`

Set the world orientation of the visual.

Parameters

<code>in</code>	<code><i>_rot</i></code>	Rotation in the world coordinate frame.
-----------------	--------------------------	---

10.263.3.80 `void gazebo::rendering::Visual::ShowBoundingBox ()`

Display the bounding box visual.

10.263.3.81 `void gazebo::rendering::Visual::ShowCollision (bool _show)`

Display the collision visuals.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to show visuals labeled as collision objects.
-----------------	---------------------------	--

10.263.3.82 `void gazebo::rendering::Visual::ShowCOM (bool _show)`

Display Center of Mass visuals.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to show center of mass visualizations.
-----------------	---------------------------	---

10.263.3.83 `void gazebo::rendering::Visual::ShowJoints (bool _show)`

Display joint visuals.

Parameters

<code>in</code>	<code>_show</code>	True to show joint visualizations.
-----------------	--------------------	------------------------------------

10.263.3.84 `void gazebo::rendering::Visual::ShowSkeleton (bool _show)`

Display the skeleton visuals.

Parameters

<code>in</code>	<code>_show</code>	True to show skeleton visuals.
-----------------	--------------------	--------------------------------

10.263.3.85 `void gazebo::rendering::Visual::ToggleVisible ()`

Toggle whether this visual is visible.

10.263.3.86 `void gazebo::rendering::Visual::Update ()`

Update the visual.

Reimplemented in `gazebo::rendering::TransmitterVisual` (p. 1132).

10.263.3.87 `void gazebo::rendering::Visual::UpdateFromMsg (ConstVisualPtr & _msg)`

Update a visual based on a message.

Parameters

<code>in</code>	<code>_msg</code>	The visual message.
-----------------	-------------------	---------------------

10.263.4 Member Data Documentation

10.263.4.1 `VisualPrivate* gazebo::rendering::Visual::dataPtr` [protected]

The documentation for this class was generated from the following file:

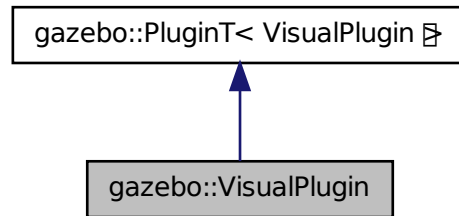
- `Visual.hh`

10.264 gazebo::VisualPlugin Class Reference

A plugin loaded within the gzserver on startup.

```
#include <Plugin.hh>
```

Inheritance diagram for gazebo::VisualPlugin:



Public Member Functions

- **VisualPlugin** ()
- virtual void **Init** ()
Initialize the plugin.
- virtual void **Load** (**rendering::VisualPtr** _visual, sdf::ElementPtr _sdf)=0
Load function.
- virtual void **Reset** ()
Override this method for custom plugin reset behavior.

Additional Inherited Members

10.264.1 Detailed Description

A plugin loaded within the gzserver on startup.

See [reference](#).

10.264.2 Constructor & Destructor Documentation

10.264.2.1 gazebo::VisualPlugin::VisualPlugin () [inline]

References [gazebo::VISUAL_PLUGIN](#).

10.264.3 Member Function Documentation

10.264.3.1 virtual void gazebo::VisualPlugin::Init () [inline], [virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.264.3.2 virtual void gazebo::VisualPlugin::Load (rendering::VisualPtr *_visual*, sdf::ElementPtr *_sdf*) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

in	<i>_visual</i>	Pointer the Visual Object.
in	<i>_sdf</i>	Pointer the the SDF element of the plugin.

10.264.3.3 virtual void gazebo::VisualPlugin::Reset () [inline],[virtual]

Override this method for custom plugin reset behavior.

The documentation for this class was generated from the following file:

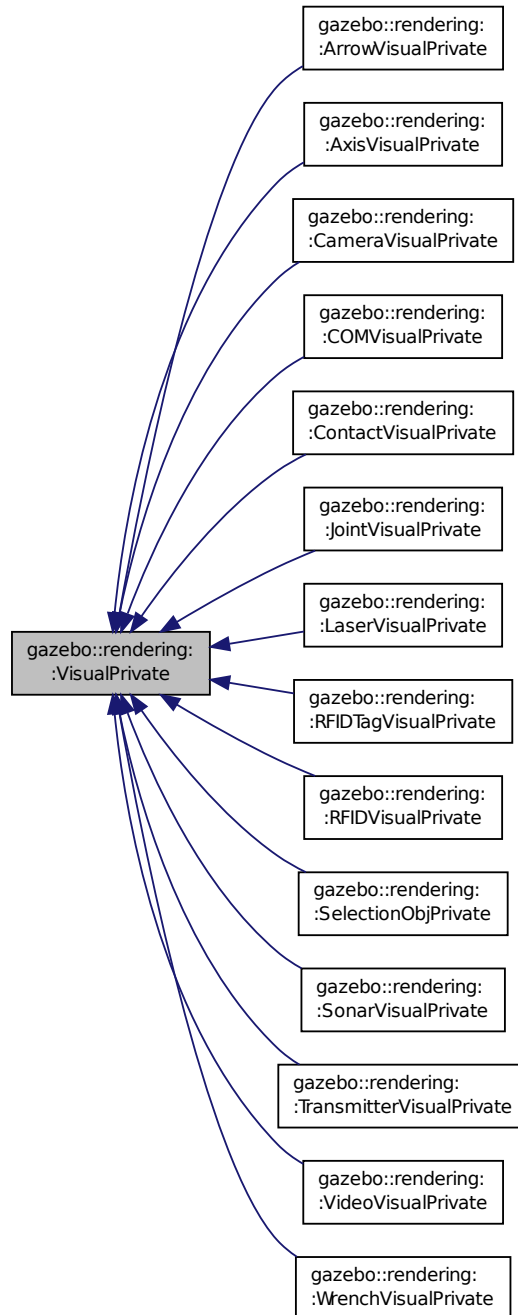
- **Plugin.hh**

10.265 gazebo::rendering::VisualPrivate Class Reference

Private data for the **Visual** (p. 1196) class.

```
#include <VisualPrivate.hh>
```

Inheritance diagram for gazebo::rendering::VisualPrivate:



Public Attributes

- `Ogre::AnimationState * animState`

Used to animate the visual.

- **WireBox * boundingBox**
A wire frame bounding box.
- `std::vector< VisualPtr > children`
Children visuals.
- `uint32_t id`
Unique id of this visual.
- `bool initialized`
True if initialized.
- `bool isStatic`
*True if the visual is static, which allows **Ogre** (p. 137) to improve performance.*
- `bool lighting`
True if lighting will be applied to this visual.
- `std::list< DynamicLines * > lines`
List of all the lines created.
- `std::list< std::pair
< DynamicLines *, unsigned int > > lineVertices`
Lines and their vertices connected to this visual.
- `std::string myMaterialName`
The unique name for the visual's material.
- `std::string name`
Name of the visual.
- `boost::function< void()> onAnimationComplete`
Callback for the animation complete event.
- `std::string origMaterialName`
The original name for the visual's material.
- **VisualPtr parent**
Parent visual.
- `std::vector< VisualPluginPtr > plugins`
A list of visual plugins.
- `event::ConnectionPtr preRenderConnection`
Connection for the pre render event.
- `common::Time prevAnimTime`
Time of the previous animation step.
- `Ogre::RibbonTrail * ribbonTrail`
The ribbon train created by the visual.
- `math::Vector3 scale`
Scale of visual.
- `ScenePtr scene`
Pointer to the visual's scene.
- `Ogre::SceneNode * sceneNode`
*Pointer to the visual's scene node in **Ogre** (p. 137).*
- `sdf::ElementPtr sdf`
The SDF element for the visual.
- `Ogre::SkeletonInstance * skeleton`
The visual's skeleton, used only for person simulation.
- `Ogre::StaticGeometry * staticGeom`

- Pointer to the static geometry.*
- float **transparency**
Transparency value.
- bool **useRTShader**
True to use RT shader system.
- bool **visible**
True if rendered.

Static Public Attributes

- static uint32_t **visualIdCount**
Counter used to create unique ids.

10.265.1 Detailed Description

Private data for the **Visual** (p. 1196) class.

10.265.2 Member Data Documentation

10.265.2.1 **Ogre::AnimationState* gazebo::rendering::VisualPrivate::animState**

Used to animate the visual.

10.265.2.2 **WireBox* gazebo::rendering::VisualPrivate::boundingBox**

A wire frame bounding box.

10.265.2.3 **std::vector<VisualPtr> gazebo::rendering::VisualPrivate::children**

Children visuals.

10.265.2.4 **uint32_t gazebo::rendering::VisualPrivate::id**

Unique id of this visual.

10.265.2.5 **bool gazebo::rendering::VisualPrivate::initialized**

True if initialized.

10.265.2.6 **bool gazebo::rendering::VisualPrivate::isStatic**

True if the visual is static, which allows **Ogre** (p. 137) to improve performance.

10.265.2.7 **bool gazebo::rendering::VisualPrivate::lighting**

True if lighting will be applied to this visual.

10.265.2.8 `std::list<DynamicLines*>` gazebo::rendering::VisualPrivate::lines

List of all the lines created.

10.265.2.9 `std::list< std::pair<DynamicLines*, unsigned int> >` gazebo::rendering::VisualPrivate::lineVertices

Lines and their vertices connected to this visual.

10.265.2.10 `std::string` gazebo::rendering::VisualPrivate::myMaterialName

The unique name for the visual's material.

10.265.2.11 `std::string` gazebo::rendering::VisualPrivate::name

Name of the visual.

10.265.2.12 `boost::function<void()>` gazebo::rendering::VisualPrivate::onAnimationComplete

Callback for the animation complete event.

10.265.2.13 `std::string` gazebo::rendering::VisualPrivate::origMaterialName

The original name for the visual's material.

10.265.2.14 `VisualPtr` gazebo::rendering::VisualPrivate::parent

Parent visual.

10.265.2.15 `std::vector<VisualPluginPtr>` gazebo::rendering::VisualPrivate::plugins

A list of visual plugins.

10.265.2.16 `event::ConnectionPtr` gazebo::rendering::VisualPrivate::preRenderConnection

Connection for the pre render event.

10.265.2.17 `common::Time` gazebo::rendering::VisualPrivate::prevAnimTime

Time of the previous animation step.

10.265.2.18 `Ogre::RibbonTrail*` gazebo::rendering::VisualPrivate::ribbonTrail

The ribbon train created by the visual.

10.265.2.19 `math::Vector3 gazebo::rendering::VisualPrivate::scale`

Scale of visual.

10.265.2.20 `ScenePtr gazebo::rendering::VisualPrivate::scene`

Pointer to the visual's scene.

10.265.2.21 `Ogre::SceneNode* gazebo::rendering::VisualPrivate::sceneNode`

Pointer to the visual's scene node in **Ogre** (p. 137).

10.265.2.22 `sdf::ElementPtr gazebo::rendering::VisualPrivate::sdf`

The SDF element for the visual.

10.265.2.23 `Ogre::SkeletonInstance* gazebo::rendering::VisualPrivate::skeleton`

The visual's skeleton, used only for person simulation.

10.265.2.24 `Ogre::StaticGeometry* gazebo::rendering::VisualPrivate::staticGeom`

Pointer to the static geometry.

10.265.2.25 `float gazebo::rendering::VisualPrivate::transparency`

Transparency value.

10.265.2.26 `bool gazebo::rendering::VisualPrivate::useRTShader`

True to use RT shader system.

10.265.2.27 `bool gazebo::rendering::VisualPrivate::visible`

True if rendered.

10.265.2.28 `uint32_t gazebo::rendering::VisualPrivate::visualIdCount` `[static]`

Counter used to create unique ids.

The documentation for this class was generated from the following file:

- **VisualPrivate.hh**

10.266 gazebo::rendering::WindowManager Class Reference

Class to manage render windows.

```
#include <rendering/rendering.hh>
```

Public Member Functions

- **WindowManager** ()
Constructor.
- virtual **~WindowManager** ()
Destructor.
- int **CreateWindow** (const std::string &_ogreHandle, uint32_t _width, uint32_t _height)
Create a window.
- void **Fini** ()
Shutdown all the windows.
- float **GetAvgFPS** (uint32_t _id)
Get the average FPS.
- uint32_t **GetTriangleCount** (uint32_t _id)
Get the triangle count.
- Ogre::RenderWindow * **GetWindow** (uint32_t _id)
Get the render window associated with the given id.
- void **Moved** (uint32_t _id)
*Tells **Ogre** (p. 137) the window has moved, and needs updating.*
- void **Resize** (uint32_t _id, int _width, int _height)
Resize a window.
- void **SetCamera** (int _windowId, **CameraPtr** _camera)
Attach a camera to a window.

10.266.1 Detailed Description

Class to manage render windows.

10.266.2 Constructor & Destructor Documentation

10.266.2.1 gazebo::rendering::WindowManager::WindowManager ()

Constructor.

10.266.2.2 virtual gazebo::rendering::WindowManager::~~WindowManager () [virtual]

Destructor.

10.266.3 Member Function Documentation

10.266.3.1 `int gazebo::rendering::WindowManager::CreateWindow (const std::string & _ogreHandle, uint32_t _width, uint32_t _height)`

Create a window.

Parameters

<code>in</code>	<code><i>_ogreHandle</i></code>	String representing the ogre window handle.
<code>in</code>	<code><i>_width</i></code>	Width of the window in pixels.
<code>in</code>	<code><i>_height</i></code>	Height of the window in pixels.

10.266.3.2 `void gazebo::rendering::WindowManager::Fini ()`

Shutdown all the windows.

10.266.3.3 `float gazebo::rendering::WindowManager::GetAvgFPS (uint32_t _id)`

Get the average FPS.

Parameters

<code>in</code>	<code><i>_id</i></code>	ID of the window.
-----------------	-------------------------	-------------------

Returns

The frames per second.

10.266.3.4 `uint32_t gazebo::rendering::WindowManager::GetTriangleCount (uint32_t _id)`

Get the triangle count.

Parameters

<code>in</code>	<code><i>_id</i></code>	ID of the window.
-----------------	-------------------------	-------------------

Returns

The triangle count.

10.266.3.5 `Ogre::RenderWindow* gazebo::rendering::WindowManager::GetWindow (uint32_t _id)`

Get the render window associated with the given id.

Parameters

<code>in</code>	<code><i>_id</i></code>	ID of the window.
-----------------	-------------------------	-------------------

Returns

Pointer to the render window, NULL if the id is invalid.

10.266.3.6 void gazebo::rendering::WindowManager::Moved (uint32_t *_id*)

Tells **Ogre** (p. 137) the window has moved, and needs updating.

Parameters

in	<i>_id</i>	ID of the window.
----	------------	-------------------

10.266.3.7 void gazebo::rendering::WindowManager::Resize (uint32_t *_id*, int *_width*, int *_height*)

Resize a window.

Parameters

in	<i>_id</i>	Id of the window to resize.
in	<i>_width</i>	New width of the window.
in	<i>_height</i>	New height of the window.

10.266.3.8 void gazebo::rendering::WindowManager::SetCamera (int *_windowId*, CameraPtr *_camera*)

Attach a camera to a window.

Parameters

in	<i>_windowId</i>	Id of the window to add the camera to.
in	<i>_camera</i>	Pointer to the camera to attach.

The documentation for this class was generated from the following file:

- **WindowManager.hh**

10.267 gazebo::rendering::WireBox Class Reference

Draws a wireframe box.

```
#include <rendering/rendering.hh>
```

Public Member Functions

- **WireBox** (VisualPtr *_parent*, const math::Box &*_box*)
Constructor.
- **~WireBox** ()
Destructor.
- **math::Box GetBox** () const

Get the wireframe box.

- bool **GetVisible** () const

Get the visibility of the box.

- void **Init** (const **math::Box** &_box)

Builds the wireframe line list.

- void **SetVisible** (bool _visible)

Set the visibility of the box.

10.267.1 Detailed Description

Draws a wireframe box.

10.267.2 Constructor & Destructor Documentation

10.267.2.1 gazebo::rendering::WireBox::WireBox (VisualPtr _parent, const math::Box & _box) [explicit]

Constructor.

Parameters

in	<code>_box</code>	Dimension of the box to draw.
in	<code>_parent</code>	Parent visual of the box.

10.267.2.2 gazebo::rendering::WireBox::~~WireBox ()

Destructor.

10.267.3 Member Function Documentation

10.267.3.1 math::Box gazebo::rendering::WireBox::GetBox () const

Get the wireframe box.

Returns

The wireframe box.

10.267.3.2 bool gazebo::rendering::WireBox::GetVisible () const

Get the visibility of the box.

Returns

True if the box is visual.

10.267.3.3 void gazebo::rendering::WireBox::Init (const math::Box & _box)

Builds the wireframe line list.

Parameters

in	_box	Box to build a wireframe from.
----	------	--------------------------------

10.267.3.4 void gazebo::rendering::WireBox::SetVisible (bool _visible)

Set the visibility of the box.

Parameters

in	_visible	True to make the box visible, False to hide.
----	----------	--

The documentation for this class was generated from the following file:

- **WireBox.hh**

10.268 gazebo::rendering::WireBoxPrivate Class Reference

Private data for the **WireBox** (p. 1227) class.

```
#include <WireBoxPrivate.hh>
```

Public Attributes

- **DynamicLines * lines**
The lines which outline the box.
- **VisualPtr parent**
The visual which this box is attached to.

10.268.1 Detailed Description

Private data for the **WireBox** (p. 1227) class.

10.268.2 Member Data Documentation

10.268.2.1 DynamicLines* gazebo::rendering::WireBoxPrivate::lines

The lines which outline the box.

10.268.2.2 VisualPtr gazebo::rendering::WireBoxPrivate::parent

The visual which this box is attached to.

The documentation for this class was generated from the following file:

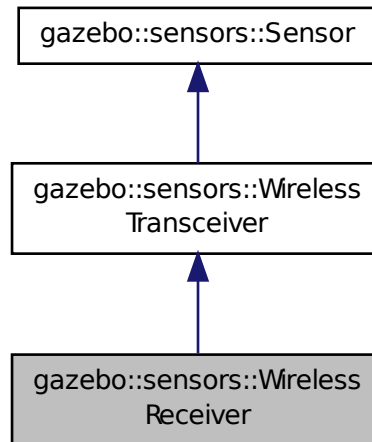
- **WireBoxPrivate.hh**

10.269 gazebo::sensors::WirelessReceiver Class Reference

Sensor (p. 907) class for receiving wireless signals.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::WirelessReceiver:



Public Member Functions

- **WirelessReceiver** ()
Constructor.
- virtual \sim **WirelessReceiver** ()
Constructor.
- virtual void **Fini** ()
Finalize the sensor.
- double **GetMaxFreqFiltered** () const
Returns the maximum frequency filtered (MHz).
- double **GetMinFreqFiltered** () const
Returns the minimum frequency filtered (MHz).
- double **GetSensitivity** () const
Returns the receiver sensitivity (dBm).
- virtual void **Init** ()
Initialize the sensor.
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.

Additional Inherited Members

10.269.1 Detailed Description

Sensor (p. 907) class for receiving wireless signals.

10.269.2 Constructor & Destructor Documentation

10.269.2.1 gazebo::sensors::WirelessReceiver::WirelessReceiver ()

Constructor.

10.269.2.2 virtual gazebo::sensors::WirelessReceiver::~WirelessReceiver () [virtual]

Constructor.

10.269.3 Member Function Documentation

10.269.3.1 virtual void gazebo::sensors::WirelessReceiver::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1234).

10.269.3.2 double gazebo::sensors::WirelessReceiver::GetMaxFreqFiltered () const

Returns the maximum frequency filtered (MHz).

Returns

Reception frequency (MHz).

10.269.3.3 double gazebo::sensors::WirelessReceiver::GetMinFreqFiltered () const

Returns the minimum frequency filtered (MHz).

Returns

Reception frequency (MHz).

10.269.3.4 double gazebo::sensors::WirelessReceiver::GetSensitivity () const

Returns the receiver sensitivity (dBm).

Returns

Receiver sensitivity (dBm).

10.269.3.5 virtual void gazebo::sensors::WirelessReceiver::Init () [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1234).

10.269.3.6 virtual void gazebo::sensors::WirelessReceiver::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	<i>_worldName</i>	Name of world to load from.
----	-------------------	-----------------------------

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1234).

The documentation for this class was generated from the following file:

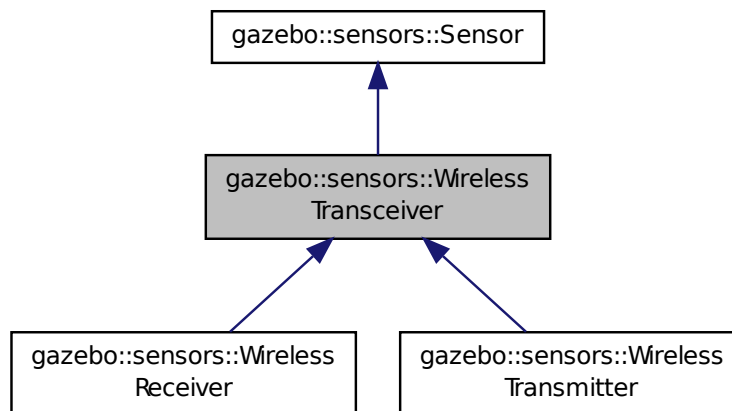
- **WirelessReceiver.hh**

10.270 gazebo::sensors::WirelessTransceiver Class Reference

Sensor (p. 907) class for receiving wireless signals.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::WirelessTransceiver:



Public Member Functions

- **WirelessTransceiver** ()
Constructor.

- **~WirelessTransceiver** ()
Constructor.
- virtual void **Fini** ()
Finalize the sensor.
- double **GetGain** () const
Returns the antenna's gain of the receiver (dBi).
- double **GetPower** () const
Returns the receiver power (dBm).
- virtual std::string **GetTopic** () const
Returns the topic name as set in SDF.
- virtual void **Init** ()
Initialize the sensor.
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.

Protected Attributes

- double **gain**
Antenna's gain of the receiver (dBi).
- boost::weak_ptr< **physics::Link** > **parentEntity**
Parent entity which the sensor is attached to.
- double **power**
Receiver's power (dBm).
- **transport::PublisherPtr** **pub**
Publisher to publish propagation model data.
- **math::Pose** **referencePose**
Sensor (p. 907) reference pose.

Additional Inherited Members

10.270.1 Detailed Description

Sensor (p. 907) class for receiving wireless signals.

10.270.2 Constructor & Destructor Documentation

10.270.2.1 gazebo::sensors::WirelessTransceiver::WirelessTransceiver ()

Constructor.

10.270.2.2 gazebo::sensors::WirelessTransceiver::~~WirelessTransceiver ()

Constructor.

10.270.3 Member Function Documentation

10.270.3.1 `virtual void gazebo::sensors::WirelessTransceiver::Fini () [virtual]`

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 912).

Reimplemented in **gazebo::sensors::WirelessReceiver** (p. 1231).

10.270.3.2 `double gazebo::sensors::WirelessTransceiver::GetGain () const`

Returns the antenna's gain of the receiver (dBi).

Returns

Antenna's gain of the receiver (dBi).

10.270.3.3 `double gazebo::sensors::WirelessTransceiver::GetPower () const`

Returns the receiver power (dBm).

Returns

Receiver power (dBm).

10.270.3.4 `virtual std::string gazebo::sensors::WirelessTransceiver::GetTopic () const [virtual]`

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 914).

10.270.3.5 `virtual void gazebo::sensors::WirelessTransceiver::Init () [virtual]`

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 915).

Reimplemented in **gazebo::sensors::WirelessTransmitter** (p. 1238), and **gazebo::sensors::WirelessReceiver** (p. 1232).

10.270.3.6 `virtual void gazebo::sensors::WirelessTransceiver::Load (const std::string & _worldName) [virtual]`

Load the sensor with default parameters.

Parameters

<code>in</code>	<code>_worldName</code>	Name of world to load from.
-----------------	-------------------------	-----------------------------

Reimplemented from `gazebo::sensors::Sensor` (p. 915).

Reimplemented in `gazebo::sensors::WirelessTransmitter` (p. 1238), and `gazebo::sensors::WirelessReceiver` (p. 1232).

10.270.4 Member Data Documentation

10.270.4.1 `double gazebo::sensors::WirelessTransceiver::gain` [protected]

Antenna's gain of the receiver (dBi).

10.270.4.2 `boost::weak_ptr<physics::Link> gazebo::sensors::WirelessTransceiver::parentEntity` [protected]

Parent entity which the sensor is attached to.

10.270.4.3 `double gazebo::sensors::WirelessTransceiver::power` [protected]

Receiver's power (dBm).

10.270.4.4 `transport::PublisherPtr gazebo::sensors::WirelessTransceiver::pub` [protected]

Publisher to publish propagation model data.

10.270.4.5 `math::Pose gazebo::sensors::WirelessTransceiver::referencePose` [protected]

Sensor (p. 907) reference pose.

The documentation for this class was generated from the following file:

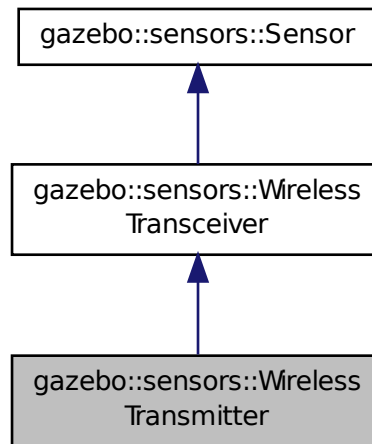
- `WirelessTransceiver.hh`

10.271 gazebo::sensors::WirelessTransmitter Class Reference

Transmitter to send wireless signals.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::WirelessTransmitter:



Public Member Functions

- **WirelessTransmitter** ()
Constructor.
- virtual \sim **WirelessTransmitter** ()
Destructor.
- std::string **GetESSID** () const
Returns the Service Set Identifier (network name).
- double **GetFreq** () const
Returns reception frequency (MHz).
- double **GetSignalStrength** (const **math::Pose** &_receiver, const double rxGain)
Returns the signal strength in a given world's point (dBm).
- virtual void **Init** ()
Initialize the sensor.
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.

Static Public Attributes

- static const double **ModelStdDesv**
Std desv of the Gaussian random variable used in the propagation model.
- static const double **NEmpty**
Constant used in the propagation model when there are no obstacles between transmitter and receiver.
- static const double **NObstacle**
Constant used in the propagation model when there are obstacles between transmitter and receiver.

Protected Member Functions

- virtual bool **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Protected Attributes

- double **freq**
Reception frequency (MHz).

10.271.1 Detailed Description

Transmitter to send wireless signals.

10.271.2 Constructor & Destructor Documentation

10.271.2.1 gazebo::sensors::WirelessTransmitter::WirelessTransmitter ()

Constructor.

10.271.2.2 virtual gazebo::sensors::WirelessTransmitter::~WirelessTransmitter () [virtual]

Destructor.

10.271.3 Member Function Documentation

10.271.3.1 std::string gazebo::sensors::WirelessTransmitter::GetESSID () const

Returns the Service Set Identifier (network name).

Returns

Service Set Identifier (network name).

10.271.3.2 double gazebo::sensors::WirelessTransmitter::GetFreq () const

Returns reception frequency (MHz).

Returns

Reception frequency (MHz).

10.271.3.3 double gazebo::sensors::WirelessTransmitter::GetSignalStrength (const math::Pose & _receiver, const double rxGain)

Returns the signal strength in a given world's point (dBm).

Returns

Signal strength in a world's point (dBm).

10.271.3.4 `virtual void gazebo::sensors::WirelessTransmitter::Init () [virtual]`

Initialize the sensor.

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1234).

10.271.3.5 `virtual void gazebo::sensors::WirelessTransmitter::Load (const std::string & _worldName) [virtual]`

Load the sensor with default parameters.

Parameters

<code>in</code>	<code>_worldName</code>	Name of world to load from.
-----------------	-------------------------	-----------------------------

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1234).

10.271.3.6 `virtual bool gazebo::sensors::WirelessTransmitter::UpdateImpl (bool) [protected],[virtual]`

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

<code>in</code>	<code>_force</code>	True if update is forced, false if not
-----------------	---------------------	--

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 917).

10.271.4 Member Data Documentation

10.271.4.1 `double gazebo::sensors::WirelessTransmitter::freq [protected]`

Reception frequency (MHz).

10.271.4.2 `const double gazebo::sensors::WirelessTransmitter::ModelStdDesv [static]`

Std desv of the Gaussian random variable used in the propagation model.

10.271.4.3 `const double gazebo::sensors::WirelessTransmitter::NEmpty` `[static]`

Constant used in the propagation model when there are no obstacles between transmitter and receiver.

10.271.4.4 `const double gazebo::sensors::WirelessTransmitter::NObstacle` `[static]`

Constant used in the propagation model when there are obstacles between transmitter and receiver.

The documentation for this class was generated from the following file:

- **WirelessTransmitter.hh**

10.272 gazebo::physics::World Class Reference

The world provides access to all other object within a simulated environment.

```
#include <physics/physics.hh>
```

Public Member Functions

- **World** (const std::string &_name="")
Constructor.
- **~World** ()
Destructor.
- void **Clear** ()
Remove all entities from the world.
- void **ClearModels** ()
Remove all entities from the world.
- void **DisableAllModels** ()
Disable all links in all the models.
- void **EnableAllModels** ()
Enable all links in all the models.
- void **EnablePhysicsEngine** (bool _enable)
enable/disable physics engine during World::Update.
- void **Fini** ()
Finalize the world.
- **BasePtr GetByName** (const std::string &_name)
Get an element by name.
- bool **GetEnablePhysicsEngine** ()
check if physics engine is enabled/disabled.
- **EntityPtr GetEntity** (const std::string &_name)
*Get a pointer to an **Entity** (p. 404) based on a name.*
- **EntityPtr GetEntityBelowPoint** (const **math::Vector3** &_pt)
Get the nearest entity below a point.
- uint32_t **GetIterations** () const
Get the total number of iterations.
- **ModelPtr GetModel** (unsigned int _index) const

- Get a model based on an index.*

 - **ModelPtr GetModel** (const std::string &_name)
Get a model by name.
 - **ModelPtr GetModelBelowPoint** (const math::Vector3 &_pt)
Get the nearest model below and not encapsulating a point.
 - unsigned int **GetModelCount** () const
Get the number of models.
 - **Model_V GetModels** () const
Get a list of all the models.
 - std::string **GetName** () const
Get the name of the world.
 - **common::Time GetPauseTime** () const
Get the amount of time simulation has been paused.
 - **PhysicsEnginePtr GetPhysicsEngine** () const
Return the physics engine.
 - **common::Time GetRealTime** () const
Get the real time (elapsed time).
 - bool **GetRunning** () const
Return the running state of the world.
 - msgs::Scene **GetSceneMsg** () const
Get the current scene in message form.
 - **EntityPtr GetSelectedEntity** () const
*Get the selected **Entity** (p. 404).*
 - boost::mutex * **GetSetWorldPoseMutex** () const
Get the set world pose mutex.
 - **common::Time GetSimTime** () const
*Get the world simulation time, note if you want the PC wall clock call **common::Time::GetWallTime** (p. 1105).*
 - **common::SphericalCoordinatesPtr GetSphericalCoordinates** () const
Return the spherical coordinates converter.
 - **common::Time GetStartTime** () const
Get the wall time simulation was started.
 - void **Init** ()
Initialize the world.
 - void **InsertModelFile** (const std::string &_sdfFilename)
Insert a model from an SDF file.
 - void **InsertModelSDF** (const sdf::SDF &_sdf)
Insert a model using SDF.
 - void **InsertModelString** (const std::string &_sdfString)
Insert a model from an SDF string.
 - bool **IsLoaded** () const
Return true if the world has been loaded.
 - bool **IsPaused** () const
Returns the state of the simulation true if paused.
 - void **Load** (sdf::ElementPtr _sdf)
Load the world using SDF parameters.
 - void **LoadPlugin** (const std::string &_filename, const std::string &_name, sdf::ElementPtr _sdf)
Load a plugin.

- void **PrintEntityTree** ()
*Print **Entity** (p. 404) tree.*
- void **PublishModelPose** (physics::ModelPtr _model)
Publish pose updates for a model.
- void **RemovePlugin** (const std::string &_name)
Remove a running plugin.
- void **Reset** ()
Reset time and model poses, configurations in simulation.
- void **ResetEntities** (Base::EntityType _type=Base::BASE)
Reset with options.
- void **ResetTime** ()
Reset simulation time back to zero.
- void **Run** (unsigned int _iterations=0)
Run the world in a thread.
- void **RunBlocking** (unsigned int _iterations=0)
Run the world. This call blocks. Run the update loop.
- void **Save** (const std::string &_filename)
Save a world to a file.
- void **SetPaused** (bool _p)
Set whether the simulation is paused.
- void **SetSimTime** (const common::Time &_t)
Set the sim time.
- void **SetState** (const WorldState &_state)
Set the current world state.
- void **Step** (unsigned int _steps)
Step the world forward in time.
- void **StepWorld** (int _steps) **GAZEBO_DEPRECATED**(3.0)
Step the world forward in time.
- void **Stop** ()
Stop the world.
- std::string **StripWorldName** (const std::string &_name) const
Return a version of the name with "<world_name>::" removed.
- void **UpdateStateSDF** ()
Update the state SDF value from the current state.

Public Attributes

- std::list< **Entity** * > **dirtyPoses**
*when physics engine makes an update and changes a link pose, this flag is set to trigger **Entity::SetWorldPose** (p. 414) on the **physics::Link** (p. 595) in **World::Update**.*

10.272.1 Detailed Description

The world provides access to all other object within a simulated environment.

The **World** (p. 1239) is the container for all models and their components (links, joints, sensors, plugins, etc), and **World-Plugin** (p. 1251) instances. Many core function are also handled in the **World** (p. 1239), including physics update, model updates, and message processing.

10.272.2 Constructor & Destructor Documentation

10.272.2.1 `gazebo::physics::World::World (const std::string & _name = " ") [explicit]`

Constructor.

Constructor for the **World** (p. 1239). Must specify a unique name.

Parameters

<code>in</code>	<code><i>_name</i></code>	Name of the world.
-----------------	---------------------------	--------------------

10.272.2.2 `gazebo::physics::World::~~World ()`

Destructor.

10.272.3 Member Function Documentation

10.272.3.1 `void gazebo::physics::World::Clear ()`

Remove all entities from the world.

This function has delayed effect. Models are cleared at the end of the current update iteration.

10.272.3.2 `void gazebo::physics::World::ClearModels ()`

Remove all entities from the world.

Implementation of **World::Clear** (p. 1242)

10.272.3.3 `void gazebo::physics::World::DisableAllModels ()`

Disable all links in all the models.

Disable is a physics concept. Disabling means that the physics engine should not update an entity.

10.272.3.4 `void gazebo::physics::World::EnableAllModels ()`

Enable all links in all the models.

Enable is a physics concept. Enabling means that the physics engine should update an entity.

10.272.3.5 `void gazebo::physics::World::EnablePhysicsEngine (bool _enable) [inline]`

enable/disable physics engine during `World::Update`.

Parameters

<code>in</code>	<code><i>_enable</i></code>	True to enable the physics engine.
-----------------	-----------------------------	------------------------------------

10.272.3.6 void gazebo::physics::World::Fini ()

Finalize the world.

Call this function to tear-down the world.

10.272.3.7 **BasePtr** gazebo::physics::World::GetByName (const std::string & *_name*)

Get an element by name.

Searches the list of entities, and return a pointer to the model with a matching *_name*.

Parameters

<i>in</i>	<i>_name</i>	The name of the Model (p. 678) to find.
-----------	--------------	--

Returns

A pointer to the entity, or NULL if no entity was found.

10.272.3.8 bool gazebo::physics::World::GetEnablePhysicsEngine () [*inline*]

check if physics engine is enabled/disabled.

Parameters

<i>True</i>	if the physics engine is enabled.
-------------	-----------------------------------

10.272.3.9 **EntityPtr** gazebo::physics::World::GetEntity (const std::string & *_name*)

Get a pointer to an **Entity** (p. 404) based on a name.

This function is the same as GetByName, but limits the search to only Entities.

Parameters

<i>in</i>	<i>_name</i>	The name of the Entity (p. 404) to find.
-----------	--------------	---

Returns

A pointer to the **Entity** (p. 404), or NULL if no **Entity** (p. 404) was found.

10.272.3.10 **EntityPtr** gazebo::physics::World::GetEntityBelowPoint (const math::Vector3 & *_pt*)

Get the nearest entity below a point.

Projects a Ray down (-Z axis) starting at the given point. The first entity hit by the Ray is returned.

Parameters

<i>in</i>	<i>_pt</i>	The 3D point to search below
-----------	------------	------------------------------

Returns

A pointer to nearest **Entity** (p. 404), NULL if none is found.

10.272.3.11 `uint32_t gazebo::physics::World::GetIterations () const`

Get the total number of iterations.

Returns

Number of iterations that simulation has taken.

10.272.3.12 `ModelPtr gazebo::physics::World::GetModel (unsigned int _index) const`

Get a model based on an index.

Get a **Model** (p. 678) using an index, where index must be greater than zero and less than **World::GetModelCount()** (p. 1245)

Parameters

<code>in</code>	<code>_index</code>	The index of the model [0..GetModelCount)
-----------------	---------------------	---

Returns

A pointer to the **Model** (p. 678). NULL if `_index` is invalid.

10.272.3.13 `ModelPtr gazebo::physics::World::GetModel (const std::string & _name)`

Get a model by name.

This function is the same as `GetByName`, but limits the search to only models.

Parameters

<code>in</code>	<code>_name</code>	The name of the Model (p. 678) to find.
-----------------	--------------------	--

Returns

A pointer to the **Model** (p. 678), or NULL if no model was found.

10.272.3.14 `ModelPtr gazebo::physics::World::GetModelBelowPoint (const math::Vector3 & _pt)`

Get the nearest model below and not encapsulating a point.

Only objects below the start point can be returned. Any object that encapsulates the start point can not be returned from this function. This function makes use of **World::GetEntityBelowPoint** (p. 1243).

Parameters

<code>in</code>	<code>_pt</code>	The 3D point to search below.
-----------------	------------------	-------------------------------

Returns

A pointer to nearest **Model** (p. 678), NULL if none is found.

10.272.3.15 `unsigned int gazebo::physics::World::GetModelCount () const`

Get the number of models.

Returns

The number of models in the **World** (p. 1239).

10.272.3.16 `Model_V gazebo::physics::World::GetModels () const`

Get a list of all the models.

Returns

A list of all the Models in the world.

10.272.3.17 `std::string gazebo::physics::World::GetName () const`

Get the name of the world.

Returns

The name of the world.

10.272.3.18 `common::Time gazebo::physics::World::GetPauseTime () const`

Get the amount of time simulation has been paused.

Returns

The pause time.

10.272.3.19 `PhysicsEnginePtr gazebo::physics::World::GetPhysicsEngine () const`

Return the physics engine.

Get a pointer to the physics engine used by the world.

Returns

Pointer to the physics engine.

10.272.3.20 `common::Time gazebo::physics::World::GetRealTime () const`

Get the real time (elapsed time).

Returns

The real time.

10.272.3.21 `bool gazebo::physics::World::GetRunning () const`

Return the running state of the world.

Returns

True if the world is running.

10.272.3.22 `msgs::Scene gazebo::physics::World::GetSceneMsg () const`

Get the current scene in message form.

Returns

The scene state as a protobuf message.

10.272.3.23 `EntityPtr gazebo::physics::World::GetSelectedEntity () const`

Get the selected **Entity** (p. 404).

The selected entity is set via the GUI.

Returns

A point to the **Entity** (p. 404), NULL if nothing is selected.

10.272.3.24 `boost::mutex* gazebo::physics::World::GetSetWorldPoseMutex () const` `[inline]`

Get the set world pose mutex.

Returns

Pointer to the mutex.

10.272.3.25 `common::Time gazebo::physics::World::GetSimTime () const`

Get the world simulation time, note if you want the PC wall clock call `common::Time::GetWallTime` (p. 1105).

Returns

The current simulation time

10.272.3.26 `common::SphericalCoordinatesPtr gazebo::physics::World::GetSphericalCoordinates () const`

Return the spherical coordinates converter.

Returns

Pointer to the spherical coordinates converter.

10.272.3.27 `common::Time gazebo::physics::World::GetStartTime () const`

Get the wall time simulation was started.

Returns

The start time.

10.272.3.28 `void gazebo::physics::World::Init ()`

Initialize the world.

This is called after Load.

10.272.3.29 `void gazebo::physics::World::InsertModelFile (const std::string & _sdfFilename)`

Insert a model from an SDF file.

Spawns a model into the world base on and SDF file.

Parameters

<code>in</code>	<code>_sdfFilename</code>	The name of the SDF file (including path).
-----------------	---------------------------	--

10.272.3.30 `void gazebo::physics::World::InsertModelSDF (const sdf::SDF & _sdf)`

Insert a model using SDF.

Spawns a model into the world base on and SDF object.

Parameters

<code>in</code>	<code>_sdf</code>	A reference to an SDF object.
-----------------	-------------------	-------------------------------

10.272.3.31 `void gazebo::physics::World::InsertModelString (const std::string & _sdfString)`

Insert a model from an SDF string.

Spawns a model into the world base on and SDF string.

Parameters

<code>in</code>	<code>_sdfString</code>	A string containing valid SDF markup.
-----------------	-------------------------	---------------------------------------

10.272.3.32 `bool gazebo::physics::World::IsLoaded () const`

Return true if the world has been loaded.

Returns

True if **World::Load** (p. 1248) has completed.

10.272.3.33 `bool gazebo::physics::World::IsPaused () const`

Returns the state of the simulation true if paused.

Returns

True if paused.

10.272.3.34 `void gazebo::physics::World::Load (sdf::ElementPtr _sdf)`

Load the world using SDF parameters.

Load a world from and SDF pointer.

Parameters

in	_sdf	SDF parameters.
----	------	-----------------

10.272.3.35 `void gazebo::physics::World::LoadPlugin (const std::string & _filename, const std::string & _name, sdf::ElementPtr _sdf)`

Load a plugin.

Parameters

in	_filename	The filename of the plugin.
in	_name	A unique name for the plugin.
in	_sdf	The SDF to pass into the plugin.

10.272.3.36 `void gazebo::physics::World::PrintEntityTree ()`

Print **Entity** (p. 404) tree.

Prints alls the entities to stdout.

10.272.3.37 `void gazebo::physics::World::PublishModelPose (physics::ModelPtr _model)`

Publish pose updates for a model.

This list of models to publish is processed and cleared once every iteration.

Parameters

in	<code>_model</code>	Pointer to the model to publish.
----	---------------------	----------------------------------

10.272.3.38 void gazebo::physics::World::RemovePlugin (const std::string & *_name*)

Remove a running plugin.

Parameters

in	<code>_name</code>	The unique name of the plugin to remove.
----	--------------------	--

10.272.3.39 void gazebo::physics::World::Reset ()

Reset time and model poses, configurations in simulation.

10.272.3.40 void gazebo::physics::World::ResetEntities (Base::EntityType *_type* = Base::BASE)

Reset with options.

The `_type` parameter specifies which type of entities to reset. See **Base::EntityType** (p. 172).

Parameters

in	<code>_type</code>	The type of reset.
----	--------------------	--------------------

10.272.3.41 void gazebo::physics::World::ResetTime ()

Reset simulation time back to zero.

10.272.3.42 void gazebo::physics::World::Run (unsigned int *_iterations* = 0)

Run the world in a thread.

Run the update loop.

Parameters

in	<code>_iterations</code>	Run for this many iterations, then stop. A value of zero disables run stop.
----	--------------------------	---

10.272.3.43 void gazebo::physics::World::RunBlocking (unsigned int *_iterations* = 0)

Run the world. This call blocks. Run the update loop.

Todo In gazebo 3.0 this should be move to the proper section.

Parameters

in	<code>_iterations</code>	Run for this many iterations, then stop. A value of zero disables run stop.
----	--------------------------	---

10.272.3.44 `void gazebo::physics::World::Save (const std::string & filename)`

Save a world to a file.

Save the current world and its state to a file.

Parameters

<code>in</code>	<code><i>filename</i></code>	Name of the file to save into.
-----------------	------------------------------	--------------------------------

10.272.3.45 `void gazebo::physics::World::SetPaused (bool p)`

Set whether the simulation is paused.

Parameters

<code>in</code>	<code><i>p</i></code>	True pauses the simulation. False runs the simulation.
-----------------	-----------------------	--

10.272.3.46 `void gazebo::physics::World::SetSimTime (const common::Time & t)`

Set the sim time.

Parameters

<code>in</code>	<code><i>t</i></code>	The new simulation time
-----------------	-----------------------	-------------------------

10.272.3.47 `void gazebo::physics::World::SetState (const WorldState & state)`

Set the current world state.

Parameters

<code><i>state</i></code>	The state to set the World (p. 1239) to.	
---------------------------	---	--

10.272.3.48 `void gazebo::physics::World::Step (unsigned int steps)`

Step the world forward in time.

Parameters

<code>in</code>	<code><i>steps</i></code>	The number of steps the World (p. 1239) should take.
-----------------	---------------------------	---

10.272.3.49 `void gazebo::physics::World::StepWorld (int steps)`

Step the world forward in time.

Parameters

<code>in</code>	<code><i>steps</i></code>	The number of steps the World (p. 1239) should take.
-----------------	---------------------------	---

Note

Deprecated. Please use **World::Step** (p. 1250)

10.272.3.50 `void gazebo::physics::World::Stop ()`

Stop the world.

Stop the update loop.

10.272.3.51 `std::string gazebo::physics::World::StripWorldName (const std::string & _name) const`

Return a version of the name with "<world_name>:" removed.

Parameters

<code>in</code>	<code>_name</code>	Usually the name of an entity.
-----------------	--------------------	--------------------------------

Returns

The stripped world name.

10.272.3.52 `void gazebo::physics::World::UpdateStateSDF ()`

Update the state SDF value from the current state.

10.272.4 Member Data Documentation

10.272.4.1 `std::list<Entity*> gazebo::physics::World::dirtyPoses`

when physics engine makes an update and changes a link pose, this flag is set to trigger **Entity::SetWorldPose** (p. 414) on the **physics::Link** (p. 595) in **World::Update**.

The documentation for this class was generated from the following file:

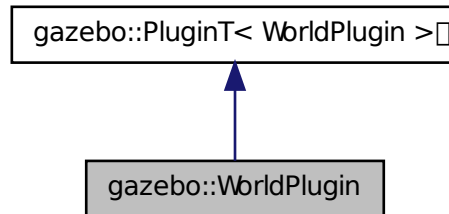
- **World.hh**

10.273 gazebo::WorldPlugin Class Reference

A plugin with access to **physics::World** (p. 1239).

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::WorldPlugin:



Public Member Functions

- **WorldPlugin** ()
Constructor.
- virtual **~WorldPlugin** ()
Destructor.
- virtual void **Init** ()
- virtual void **Load** (**physics::WorldPtr** _world, sdf::ElementPtr _sdf)=0
Load function.
- virtual void **Reset** ()

Additional Inherited Members

10.273.1 Detailed Description

A plugin with access to **physics::World** (p. 1239).

See [reference](#).

10.273.2 Constructor & Destructor Documentation

10.273.2.1 gazebo::WorldPlugin::WorldPlugin () [inline]

Constructor.

References [gazebo::WORLD_PLUGIN](#).

10.273.2.2 virtual gazebo::WorldPlugin::~~WorldPlugin () [inline],[virtual]

Destructor.

10.273.3 Member Function Documentation

10.273.3.1 virtual void gazebo::WorldPlugin::Init () [inline],[virtual]

10.273.3.2 virtual void gazebo::WorldPlugin::Load (physics::WorldPtr _world, sdf::ElementPtr _sdf) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

in	<code>_world</code>	Pointer the World
in	<code>_sdf</code>	Pointer the the SDF element of the plugin.

10.273.3.3 virtual void gazebo::WorldPlugin::Reset () [inline],[virtual]

The documentation for this class was generated from the following file:

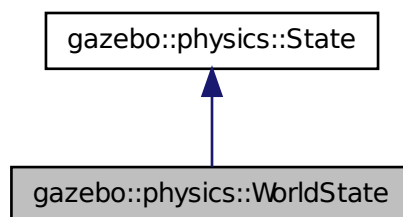
- **Plugin.hh**

10.274 gazebo::physics::WorldState Class Reference

Store state information of a **physics::World** (p. 1239) object.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::WorldState:



Public Member Functions

- **WorldState** ()
Default constructor.
- **WorldState** (const **WorldPtr** _world)
Constructor.
- **WorldState** (const sdf::ElementPtr _sdf)

Constructor.

- virtual `~WorldState ()`

Destructor.

- void `FillSDF (sdf::ElementPtr _sdf)`

Populate a state SDF element with data from the object.

- `ModelState GetModelState (const std::string &_modelName) const`

Get a model state by model name.

- unsigned int `GetModelStateCount () const`

Get the number of model states.

- `ModelState_M GetModelStates (const boost::regex &_regex) const`

Get model states based on a regular expression.

- const `ModelState_M & GetModelStates () const`

Get the model states.

- bool `HasModelState (const std::string &_modelName) const`

*Return true if **WorldState** (p. 1253) has a **ModelState** (p. 698) with the given name.*

- bool `IsZero () const`

Return true if the values in the state are zero.

- void `Load (const WorldPtr _world)`

*Load from a **World** (p. 1239) pointer.*

- virtual void `Load (const sdf::ElementPtr _elem)`

Load state from SDF element.

- `WorldState operator+` (const `WorldState &_state`) const

Addition operator.

- `WorldState operator-` (const `WorldState &_state`) const

Subtraction operator.

- `WorldState & operator=` (const `WorldState &_state`)

Assignment operator.

- virtual void `SetRealTime (const common::Time &_time)`

Set the real time when this state was generated.

- virtual void `SetSimTime (const common::Time &_time)`

Set the sim time when this state was generated.

- virtual void `SetWallTime (const common::Time &_time)`

Set the wall time when this state was generated.

- void `SetWorld (const WorldPtr _world)`

Set the world.

Friends

- `std::ostream & operator<<` (`std::ostream &_out`, const `gazebo::physics::WorldState &_state`)

Stream insertion operator.

Additional Inherited Members

10.274.1 Detailed Description

Store state information of a `physics::World` (p. 1239) object.

Instances of this class contain the state of a `World` (p. 1239) at a specific time. `World` (p. 1239) state includes the state of all models, and their children.

10.274.2 Constructor & Destructor Documentation

10.274.2.1 gazebo::physics::WorldState::WorldState ()

Default constructor.

10.274.2.2 gazebo::physics::WorldState::WorldState (const WorldPtr *_world*) [explicit]

Constructor.

Generate a **WorldState** (p. 1253) from an instance of a **World** (p. 1239).

Parameters

in	<i>_world</i>	Pointer to a world
----	---------------	--------------------

10.274.2.3 gazebo::physics::WorldState::WorldState (const sdf::ElementPtr *_sdf*) [explicit]

Constructor.

Build a **WorldState** (p. 1253) from SDF data

Parameters

in	<i>_sdf</i>	SDF data to load a world state from.
----	-------------	--------------------------------------

10.274.2.4 virtual gazebo::physics::WorldState::~~WorldState () [virtual]

Destructor.

10.274.3 Member Function Documentation

10.274.3.1 void gazebo::physics::WorldState::FillSDF (sdf::ElementPtr *_sdf*)

Populate a state SDF element with data from the object.

Parameters

out	<i>_sdf</i>	SDF element to populate.
-----	-------------	--------------------------

10.274.3.2 ModelState gazebo::physics::WorldState::GetModelState (const std::string & *_modelName*) const

Get a model state by model name.

Parameters

in	<i>_modelName</i>	Name of the model state to get.
----	-------------------	---------------------------------

Returns

The model state.

Exceptions

<i>common::Exception</i> (p. 444)	When the <code>_modelName</code> doesn't exist.
---	---

10.274.3.3 `unsigned int gazebo::physics::WorldState::GetModelStateCount () const`

Get the number of model states.

Returns the number of models in this instance.

Returns

Number of models.

10.274.3.4 `ModelState_M gazebo::physics::WorldState::GetModelStates (const boost::regex & _regex) const`

Get model states based on a regular expression.

Parameters

<code>in</code>	<code>_regex</code>	The regular expression.
-----------------	---------------------	-------------------------

Returns

List of model states whose names match the regular expression.

10.274.3.5 `const ModelState_M& gazebo::physics::WorldState::GetModelStates () const`

Get the model states.

Returns

A vector of model states.

10.274.3.6 `bool gazebo::physics::WorldState::HasModelState (const std::string & _modelName) const`

Return true if **WorldState** (p. 1253) has a **ModelState** (p. 698) with the given name.

Parameters

<code>in</code>	<code>_modelName</code>	Name of the model to search for.
-----------------	-------------------------	----------------------------------

Returns

True if the **ModelState** (p. 698) exists.

10.274.3.7 `bool gazebo::physics::WorldState::IsZero () const`

Return true if the values in the state are zero.

This will check to see if the all model states are zero.

Returns

True if the values in the state are zero.

10.274.3.8 `void gazebo::physics::WorldState::Load (const WorldPtr _world)`

Load from a **World** (p. 1239) pointer.

Generate a **WorldState** (p. 1253) from an instance of a **World** (p. 1239).

Parameters

<code>in</code>	<code>_world</code>	Pointer to a world
-----------------	---------------------	--------------------

10.274.3.9 `virtual void gazebo::physics::WorldState::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Set a **WorldState** (p. 1253) from an SDF element containing **WorldState** (p. 1253) info.

Parameters

<code>in</code>	<code>_elem</code>	Pointer to the WorldState (p. 1253) SDF element.
-----------------	--------------------	---

Reimplemented from **gazebo::physics::State** (p. 1070).

10.274.3.10 `WorldState gazebo::physics::WorldState::operator+ (const WorldState & _state) const`

Addition operator.

Parameters

<code>in</code>	<code>_pt</code>	A state to add.
-----------------	------------------	-----------------

Returns

The resulting state.

10.274.3.11 `WorldState gazebo::physics::WorldState::operator- (const WorldState & _state) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_pt</code>	A state to subtract.
-----------------	------------------	----------------------

Returns

The resulting state.

10.274.3.12 `WorldState& gazebo::physics::WorldState::operator= (const WorldState & _state)`

Assignment operator.

Parameters

<code>in</code>	<code>_state</code>	State (p. 1068) value
-----------------	---------------------	------------------------------

Returns

Reference to this

10.274.3.13 `virtual void gazebo::physics::WorldState::SetRealTime (const common::Time & _time) [virtual]`

Set the real time when this state was generated.

Parameters

<code>in</code>	<code>_time</code>	Clock time since simulation was stated.
-----------------	--------------------	---

Reimplemented from `gazebo::physics::State` (p. 1071).

10.274.3.14 `virtual void gazebo::physics::WorldState::SetSimTime (const common::Time & _time) [virtual]`

Set the sim time when this state was generated.

Parameters

<code>in</code>	<code>_time</code>	Simulation time when the data was recorded.
-----------------	--------------------	---

Reimplemented from `gazebo::physics::State` (p. 1071).

10.274.3.15 `virtual void gazebo::physics::WorldState::SetWallTime (const common::Time & _time) [virtual]`

Set the wall time when this state was generated.

Parameters

<code>in</code>	<code>_time</code>	The absolute clock time when the State (p. 1068) data was recorded.
-----------------	--------------------	--

Reimplemented from `gazebo::physics::State` (p. 1072).

10.274.3.16 `void gazebo::physics::WorldState::SetWorld (const WorldPtr _world)`

Set the world.

Parameters

in	<code>_world</code>	Pointer to the world.
----	---------------------	-----------------------

10.274.4 Friends And Related Function Documentation

10.274.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::physics::WorldState & _state) [friend]`

Stream insertion operator.

Parameters

in	<code>_out</code>	output stream
in	<code>_state</code>	World (p. 1239) state to output

Returns

the stream

The documentation for this class was generated from the following file:

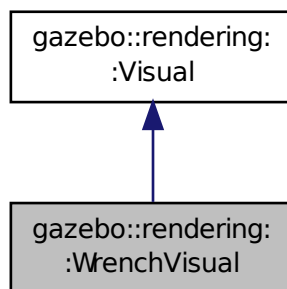
- **WorldState.hh**

10.275 gazebo::rendering::WrenchVisual Class Reference

Visualization for sonar data.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::WrenchVisual:



Public Member Functions

- **WrenchVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)
Constructor.

- virtual `~WrenchVisual ()`
Destructor.
- void `Load (ConstJointPtr &_msg)`
Load the visual based on a message.
- void `SetEnabled (bool _enabled)`
Set to true to enable wrench visualization.

Additional Inherited Members

10.275.1 Detailed Description

Visualization for sonar data.

10.275.2 Constructor & Destructor Documentation

10.275.2.1 `gazebo::rendering::WrenchVisual::WrenchVisual (const std::string & _name, VisualPtr _vis, const std::string & _topicName)`

Constructor.

Parameters

in	<code>_name</code>	Name of the visual.
in	<code>_vis</code>	Pointer to the parent Visual (p. 1196).
in	<code>_topicName</code>	Name of the topic that has sonar data.

10.275.2.2 `virtual gazebo::rendering::WrenchVisual::~~WrenchVisual () [virtual]`

Destructor.

10.275.3 Member Function Documentation

10.275.3.1 `void gazebo::rendering::WrenchVisual::Load (ConstJointPtr & _msg)`

Load the visual based on a message.

Parameters

in	<code>_msg</code>	Joint message
----	-------------------	---------------

10.275.3.2 `void gazebo::rendering::WrenchVisual::SetEnabled (bool _enabled)`

Set to true to enable wrench visualization.

Parameters

in	<code>_enabled</code>	True to show wrenches, false to hide.
----	-----------------------	---------------------------------------

The documentation for this class was generated from the following file:

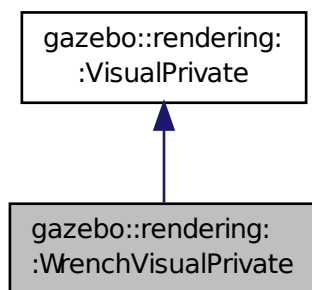
- **WrenchVisual.hh**

10.276 gazebo::rendering::WrenchVisualPrivate Class Reference

Private data for the Wrench **Visual** (p. 1196) class.

```
#include <WrenchVisualPrivate.hh>
```

Inheritance diagram for gazebo::rendering::WrenchVisualPrivate:



Public Attributes

- Ogre::SceneNode * **coneXNode**
Scene (p. 879) node for X torque visualization.
- Ogre::SceneNode * **coneYNode**
Scene (p. 879) node for Y torque visualization.
- Ogre::SceneNode * **coneZNode**
Scene (p. 879) node for Z torque visualization.
- std::vector< **event::ConnectionPtr** > **connections**
All the event connections.
- bool **enabled**
True if this visualization is enabled.
- **DynamicLines** * **forceLine**
Line to visualize force.
- Ogre::SceneNode * **forceNode**
Scene (p. 879) node for force visualization.
- boost::mutex **mutex**
Mutex to protect the contact message.
- **transport::NodePtr** **node**
Pointer to a node that handles communication.

- bool **receivedMsg**
True if we have received a message.
- boost::shared_ptr
< msgs::WrenchStamped const > **wrenchMsg**
The current wrench message.
- **transport::SubscriberPtr wrenchSub**
Subscription to the sonar data.

Additional Inherited Members

10.276.1 Detailed Description

Private data for the Wrench **Visual** (p. 1196) class.

10.276.2 Member Data Documentation

10.276.2.1 Ogre::SceneNode* gazebo::rendering::WrenchVisualPrivate::coneXNode

Scene (p. 879) node for X torque visualization.

10.276.2.2 Ogre::SceneNode* gazebo::rendering::WrenchVisualPrivate::coneYNode

Scene (p. 879) node for Y torque visualization.

10.276.2.3 Ogre::SceneNode* gazebo::rendering::WrenchVisualPrivate::coneZNode

Scene (p. 879) node for Z torque visualization.

10.276.2.4 std::vector<event::ConnectionPtr> gazebo::rendering::WrenchVisualPrivate::connections

All the event connections.

10.276.2.5 bool gazebo::rendering::WrenchVisualPrivate::enabled

True if this visualization is enabled.

10.276.2.6 DynamicLines* gazebo::rendering::WrenchVisualPrivate::forceLine

Line to visualize force.

10.276.2.7 Ogre::SceneNode* gazebo::rendering::WrenchVisualPrivate::forceNode

Scene (p. 879) node for force visualization.

10.276.2.8 `boost::mutex gazebo::rendering::WrenchVisualPrivate::mutex`

Mutex to protect the contact message.

10.276.2.9 `transport::NodePtr gazebo::rendering::WrenchVisualPrivate::node`

Pointer to a node that handles communication.

10.276.2.10 `bool gazebo::rendering::WrenchVisualPrivate::receivedMsg`

True if we have received a message.

10.276.2.11 `boost::shared_ptr<msgs::WrenchStamped const> gazebo::rendering::WrenchVisualPrivate::wrenchMsg`

The current wrench message.

10.276.2.12 `transport::SubscriberPtr gazebo::rendering::WrenchVisualPrivate::wrenchSub`

Subscription to the sonar data.

The documentation for this class was generated from the following file:

- **WrenchVisualPrivate.hh**

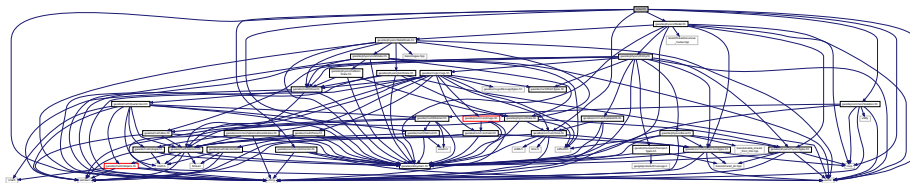
Chapter 11

File Documentation

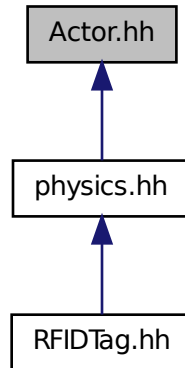
11.1 Actor.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "gazebo/physics/Model.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/common/Skeleton.hh"
#include "gazebo/common/Animation.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Actor.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Actor**

Actor (p. 139) class enables GPU based mesh model / skeleton scriptable animation.

- class **gazebo::physics::TrajectoryInfo**

Information about a trajectory for an Actor (p. 139).

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

Common namespace.

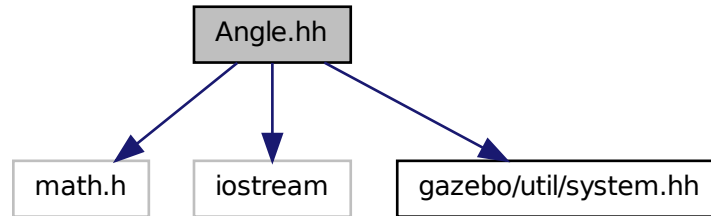
- namespace **gazebo::physics**

namespace for physics

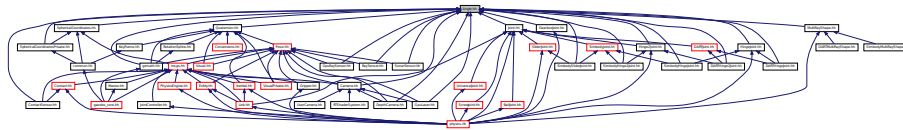
11.2 Angle.hh File Reference

```
#include <math.h>
#include <iostream>
#include "gazebo/util/system.hh"
```

Include dependency graph for Angle.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Angle**
An angle and related functions.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

Macros

- #define **GZ_DTOR**(d) ((d) * M_PI / 180)
Converts degrees to radians.
- #define **GZ_NORMALIZE**(a) (atan2(sin(a), cos(a)))
Macro tha normalizes an angle in the range -Pi to Pi.
- #define **GZ_RTOD**(r) ((r) * 180 / M_PI)
Macro that converts radians to degrees.

11.2.1 Macro Definition Documentation

11.2.1.1 `#define GZ_DTOR(d) ((d) * M_PI / 180)`

Converts degrees to radians.

Parameters

<i>in</i>	<i>degrees</i>	
-----------	----------------	--

Returns

radians

11.2.1.2 `#define GZ_NORMALIZE(a) (atan2(sin(a), cos(a)))`

Macro tha normalizes an angle in the range -Pi to Pi.

Parameters

<i>in</i>	<i>angle</i>	
-----------	--------------	--

Returns

the angle, in range

11.2.1.3 `#define GZ_RTOD(r) ((r) * 180 / M_PI)`

Macro that converts radians to degrees.

Parameters

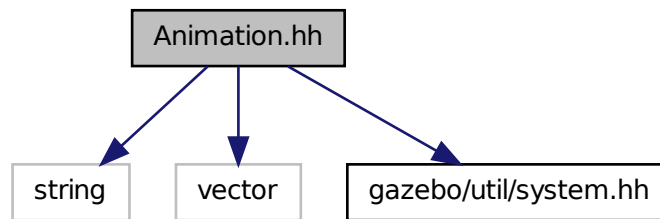
<i>in</i>	<i>radians</i>	
-----------	----------------	--

Returns

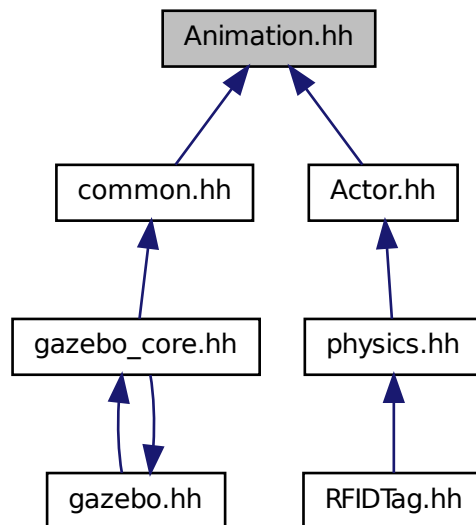
degrees

11.3 Animation.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/util/system.hh"
Include dependency graph for Animation.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

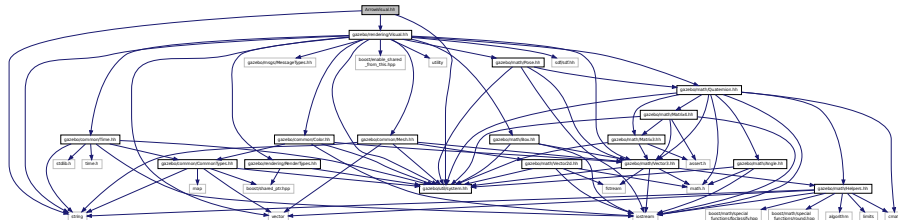
- class **gazebo::common::Animation**
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.
- class **gazebo::common::NumericAnimation**
A numeric animation.
- class **gazebo::common::PoseAnimation**
A pose animation.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::math**
Math namespace.

11.4 ArrowVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/util/system.hh"
Include dependency graph for ArrowVisual.hh:
```



Classes

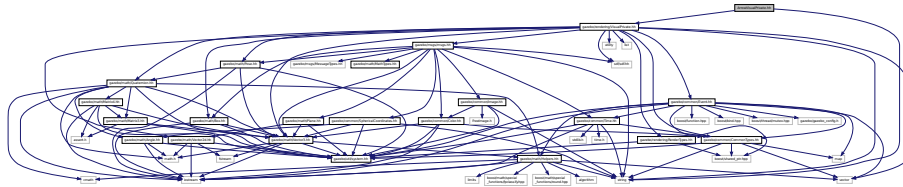
- class **gazebo::rendering::ArrowVisual**
Basic arrow visualization.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.5 ArrowVisualPrivate.hh File Reference

```
#include <string>
#include "gazebo/rendering/VisualPrivate.hh"
Include dependency graph for ArrowVisualPrivate.hh:
```



Classes

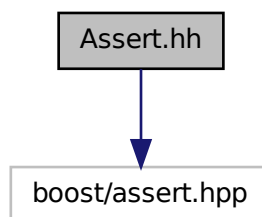
- class **gazebo::rendering::ArrowVisualPrivate**
*Private data for the Arrow **Visual** (p. 1196) class.*

Namespaces

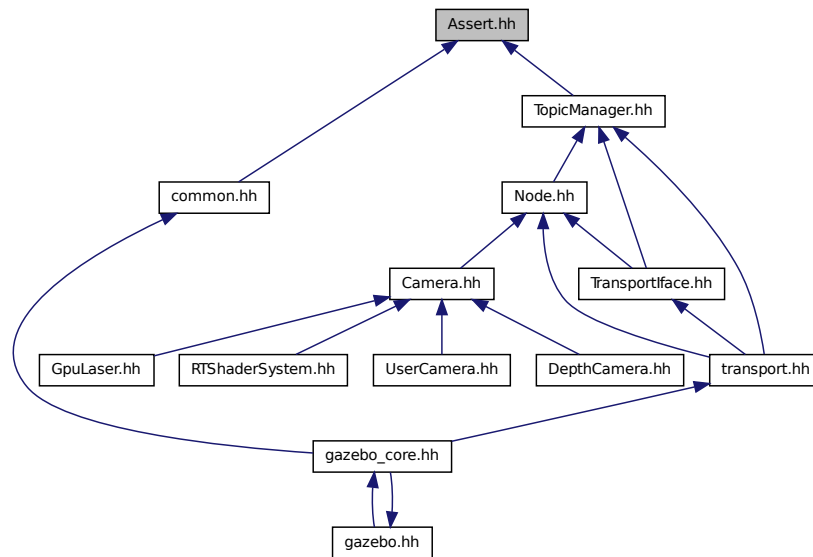
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **ogre**

11.6 Assert.hh File Reference

```
#include <boost/assert.hpp>
Include dependency graph for Assert.hh:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define GZ_ASSERT(_expr, _msg) BOOST_ASSERT_MSG(_expr, _msg)`

This macro define the standard way of launching an exception inside gazebo.

11.6.1 Macro Definition Documentation

11.6.1.1 `#define GZ_ASSERT(_expr, _msg) BOOST_ASSERT_MSG(_expr, _msg)`

This macro define the standard way of launching an exception inside gazebo.

Referenced by `gazebo::transport::TopicManager::Advertise()`.

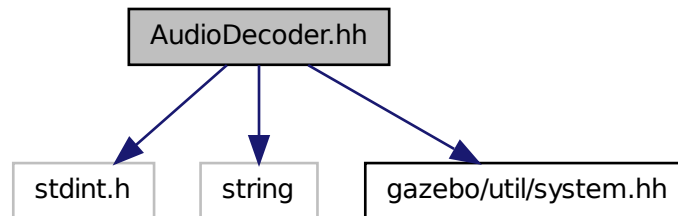
11.7 AudioDecoder.hh File Reference

```

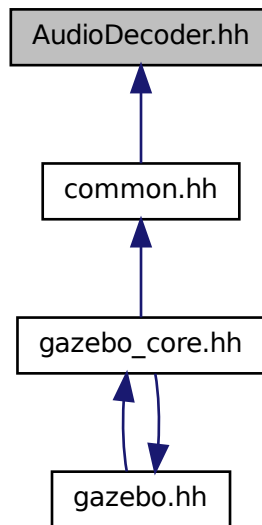
#include <stdint.h>
#include <string>
#include "gazebo/util/system.hh"

```

Include dependency graph for AudioDecoder.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::AudioDecoder**
An audio decoder based on FFMPEG.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

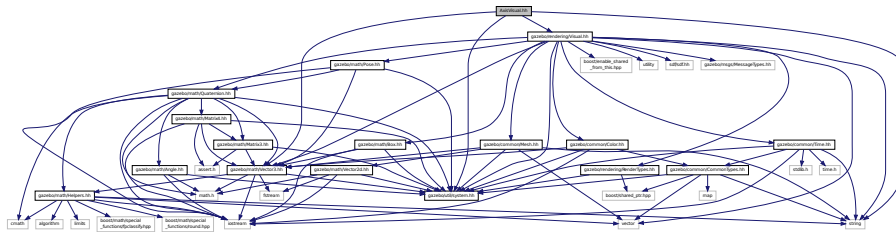
- namespace **gazebo::common**

Common namespace.

11.8 AxisVisual.hh File Reference

```
#include <string>
#include "gazebo/math/Vector3.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for AxisVisual.hh:



Classes

- class **gazebo::rendering::AxisVisual**

Basic axis visualization.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

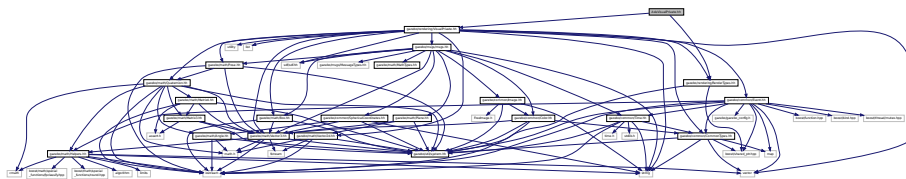
- namespace **gazebo::rendering**

Rendering namespace.

11.9 AxisVisualPrivate.hh File Reference

```
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/rendering/VisualPrivate.hh"
```

Include dependency graph for AxisVisualPrivate.hh:



Classes

- class **gazebo::rendering::AxisVisualPrivate**
*Private data for the Axis **Visual** (p. 1196) class.*

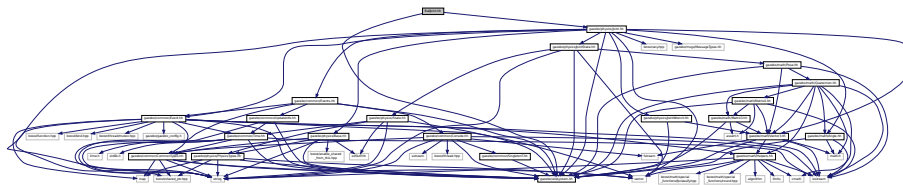
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

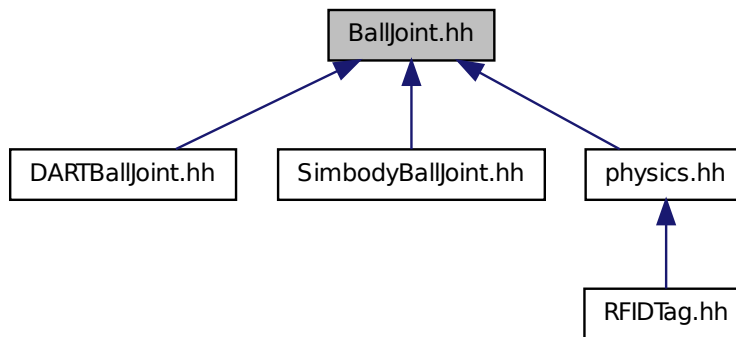
11.10 BallJoint.hh File Reference

```
#include "gazebo/physics/Joint.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for BallJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::BallJoint< T >**
***Base** (p. 168) class for a ball joint.*

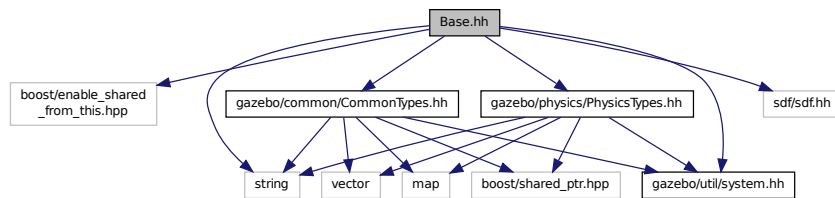
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

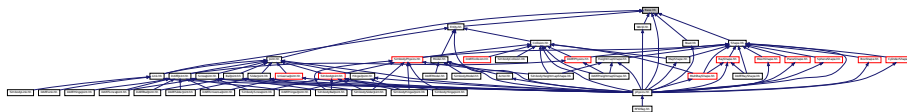
11.11 Base.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Base.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Base**
Base (p. 168) class for most physics classes.

Namespaces

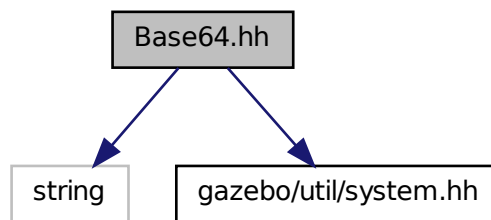
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Variables

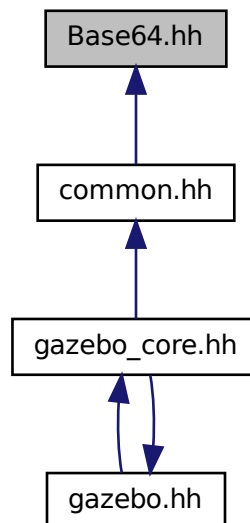
- static std::string **gazebo::physics::EntityTypename** []
String names for the different entity types.

11.12 Base64.hh File Reference

```
#include <string>
#include "gazebo/util/system.hh"
Include dependency graph for Base64.hh:
```



This graph shows which files directly or indirectly include this file:



Functions

- **GAZEBO_VISIBLE** `std::string Base64Decode (const std::string &_encodedString)`
Decode a base64 string.
- **GAZEBO_VISIBLE** `void Base64Encode (const char *_bytesToEncode, unsigned int _len, std::string &_result)`
Encode a binary string into base 64.

11.12.1 Function Documentation

11.12.1.1 **GAZEBO_VISIBLE** `std::string Base64Decode (const std::string & _encodedString)`

Decode a base64 string.

Parameters

in	<code>_encodedString</code>	A base 64 encoded string.
----	-----------------------------	---------------------------

Returns

The decoded string.

11.12.1.2 **GAZEBO_VISIBLE** `void Base64Encode (const char * _bytesToEncode, unsigned int _len, std::string & _result)`

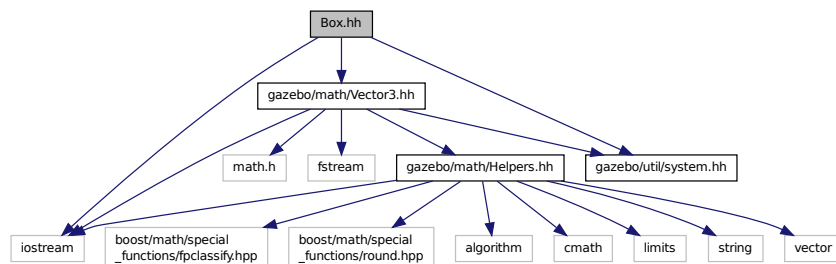
Encode a binary string into base 64.

Parameters

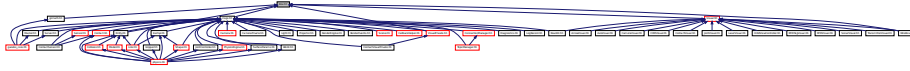
in	<code>_bytesToEncode</code>	String of bytes to encode.
in	<code>_len</code>	Length of <code>_bytesToEncode</code> .
out	<code>_result</code>	Based64 string is appended to this string.

11.13 Box.hh File Reference

```
#include <iostream>
#include "gazebo/math/Vector3.hh"
#include "gazebo/util/system.hh"
Include dependency graph for Box.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Box**

Mathematical representation of a box and related functions.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::math**

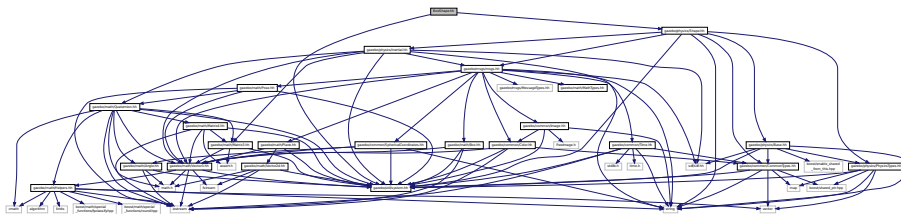
Math namespace.

11.14 BoxShape.hh File Reference

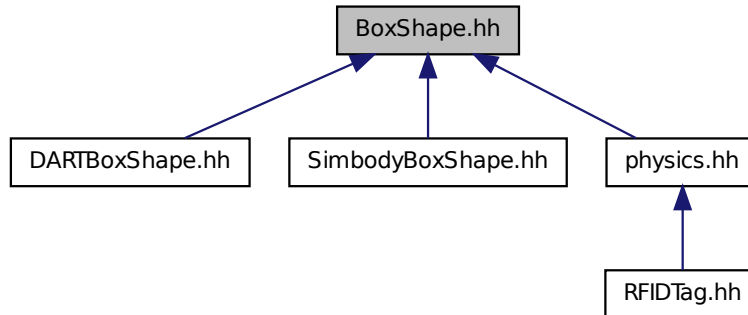
```
#include "gazebo/physics/Shape.hh"
```

```
#include "gazebo/util/system.hh"
```

Include dependency graph for BoxShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::BoxShape**

Box geometry primitive.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

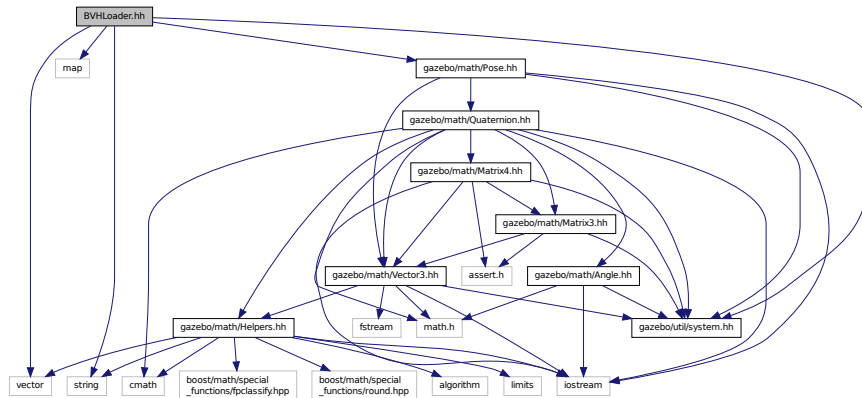
- namespace **gazebo::physics**

namespace for physics

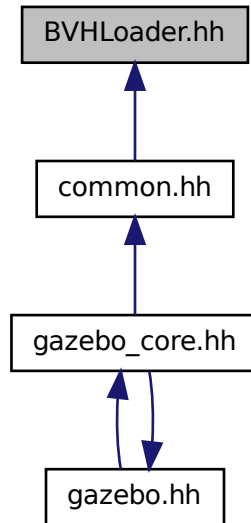
11.15 BVHLoader.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include "gazebo/math/Pose.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for BVHLoader.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::BVHLoader**
Handles loading BVH animation files.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

Macros

- `#define X_POSITION 0`
- `#define X_ROTATION 3`
- `#define Y_POSITION 1`
- `#define Y_ROTATION 4`
- `#define Z_POSITION 2`
- `#define Z_ROTATION 5`

11.15.1 Macro Definition Documentation

11.15.1.1 `#define X_POSITION 0`

11.15.1.2 `#define X_ROTATION 3`

11.15.1.3 `#define Y_POSITION 1`

11.15.1.4 `#define Y_ROTATION 4`

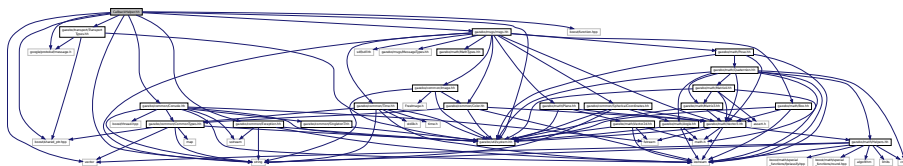
11.15.1.5 `#define Z_POSITION 2`

11.15.1.6 `#define Z_ROTATION 5`

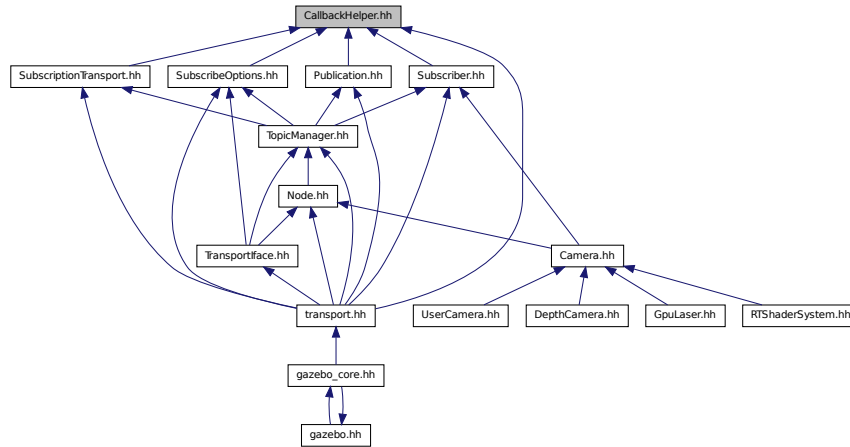
11.16 CallbackHelper.hh File Reference

```
#include <google/protobuf/message.h>
#include <boost/function.hpp>
#include <boost/shared_ptr.hpp>
#include <vector>
#include <string>
#include "gazebo/common/Console.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Exception.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for CallbackHelper.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::CallbackHelper**
A helper class to handle callbacks when messages arrive.
- class **gazebo::transport::CallbackHelperT< M >**
Callback helper Template.
- class **gazebo::transport::RawCallbackHelper**
Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

Typedefs

- typedef boost::shared_ptr
< CallbackHelper > **gazebo::transport::CallbackHelperPtr**
*boost shared pointer to **transport::CallbackHelper** (p. 192)*

11.17 Camera.hh File Reference

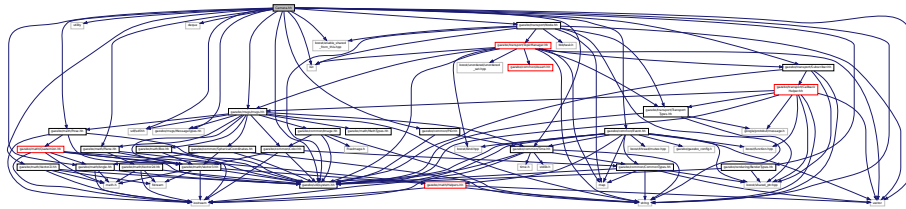
```
#include <boost/enable_shared_from_this.hpp>
```

```

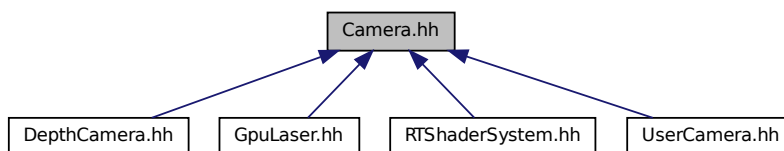
#include <string>
#include <utility>
#include <list>
#include <vector>
#include <deque>
#include <sdf/sdf.hh>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/Node.hh"
#include "gazebo/transport/Subscriber.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/common/PID.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Plane.hh"
#include "gazebo/math/Vector2i.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for Camera.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::Camera**
Basic camera sensor.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

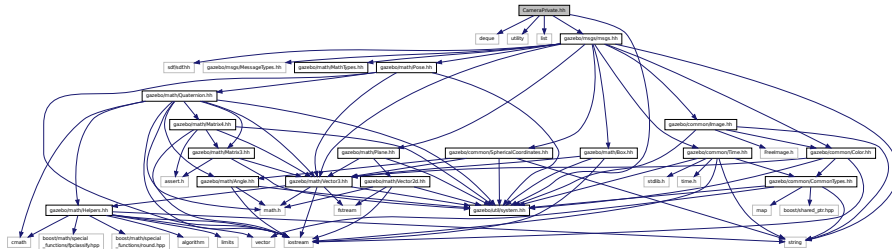
- namespace **gazebo::rendering**

Rendering namespace.

- namespace **Ogre**

11.18 CameraPrivate.hh File Reference

```
#include <deque>
#include <utility>
#include <list>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/util/system.hh"
Include dependency graph for CameraPrivate.hh:
```



Classes

- class **gazebo::rendering::CameraPrivate**

Private data for the **Camera** (p. 197) class.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

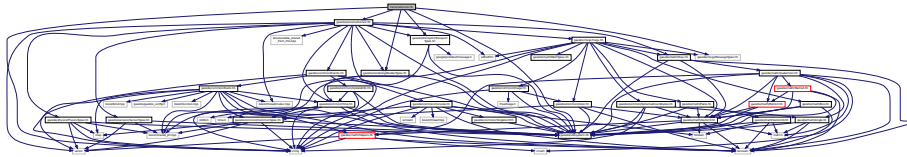
Rendering namespace.

- namespace **Ogre**

11.19 CameraSensor.hh File Reference

```
#include <string>
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for CameraSensor.hh:



Classes

- class **gazebo::sensors::CameraSensor**
Basic camera sensor.

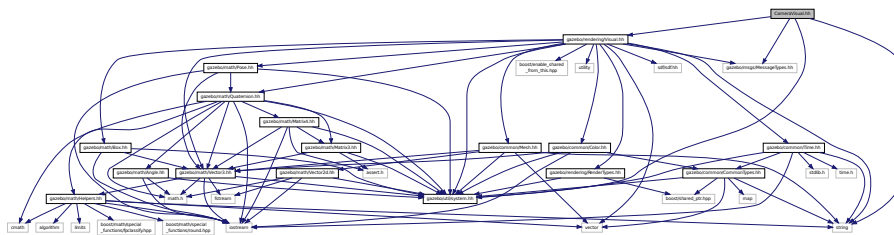
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.20 CameraVisual.hh File Reference

```
#include <string>
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for CameraVisual.hh:



Classes

- class **gazebo::rendering::CameraVisual**
Basic camera visualization.

Namespaces

- namespace **gazebo**

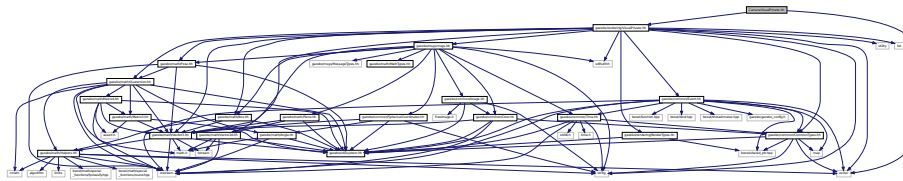
Forward declarations for the common classes.

- namespace **gazebo::rendering**

Rendering namespace.

11.21 CameraVisualPrivate.hh File Reference

```
#include <vector>
#include "gazebo/rendering/VisualPrivate.hh"
Include dependency graph for CameraVisualPrivate.hh:
```



Classes

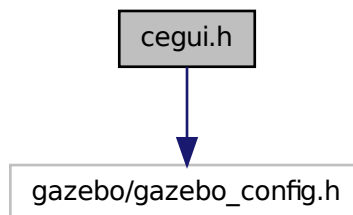
- class **gazebo::rendering::CameraVisualPrivate**

Namespaces

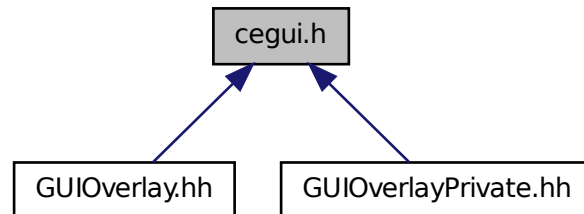
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.22 cegui.h File Reference

```
#include "gazebo/gazebo_config.h"
Include dependency graph for cegui.h:
```



This graph shows which files directly or indirectly include this file:



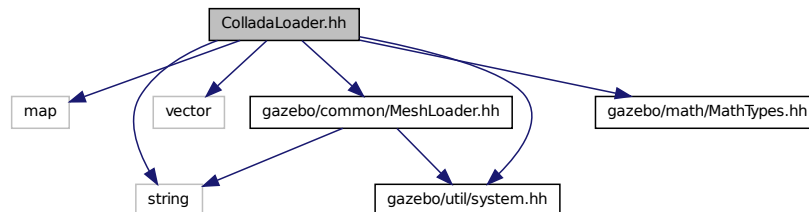
11.23 ColladaLoader.hh File Reference

```

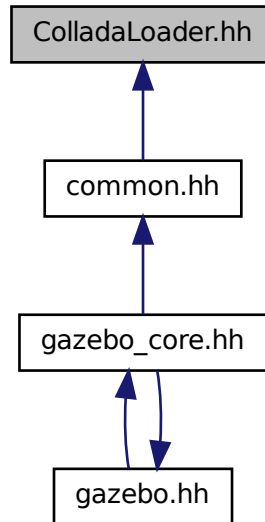
#include <map>
#include <string>
#include <vector>
#include "gazebo/common/MeshLoader.hh"
#include "gazebo/math/MathTypes.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for `ColladaLoader.hh`:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::ColladaLoader**
Class used to load Collada mesh files.

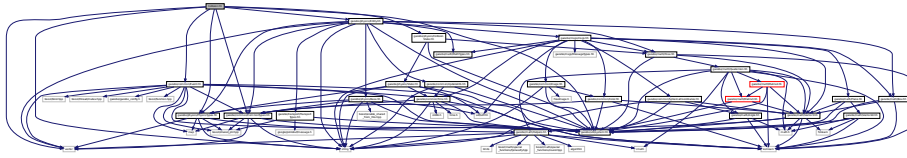
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

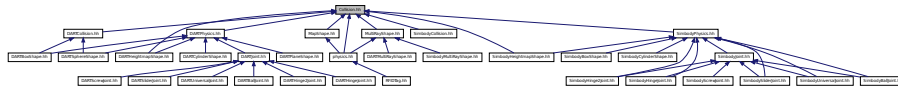
11.24 Collision.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/common/Event.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/CollisionState.hh"
#include "gazebo/physics/Entity.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Collision.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Collision**
Base (p. 168) class for all collision entities.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

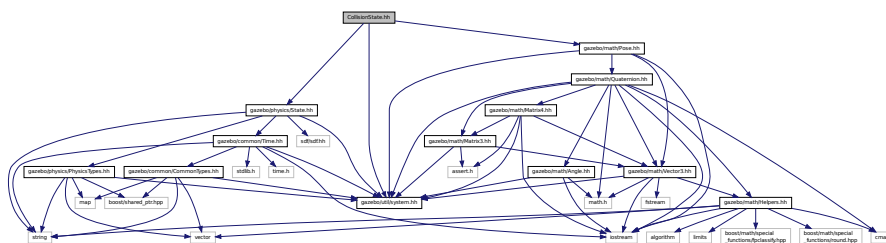
11.25 CollisionState.hh File Reference

```
#include "gazebo/physics/State.hh"
```

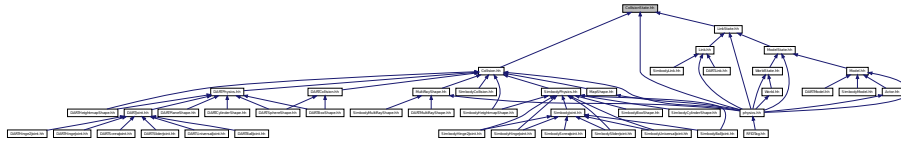
```
#include "gazebo/math/Pose.hh"
```

```
#include "gazebo/util/system.hh"
```

Include dependency graph for CollisionState.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::CollisionState**
*Store state information of a **physics::Collision** (p. 235) object.*

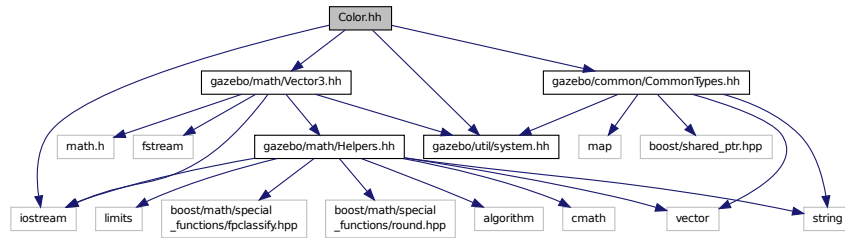
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.26 Color.hh File Reference

```
#include <iostream>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Color.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Color**

Defines a color.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

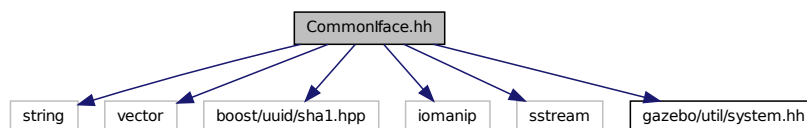
- namespace **gazebo::common**

Common namespace.

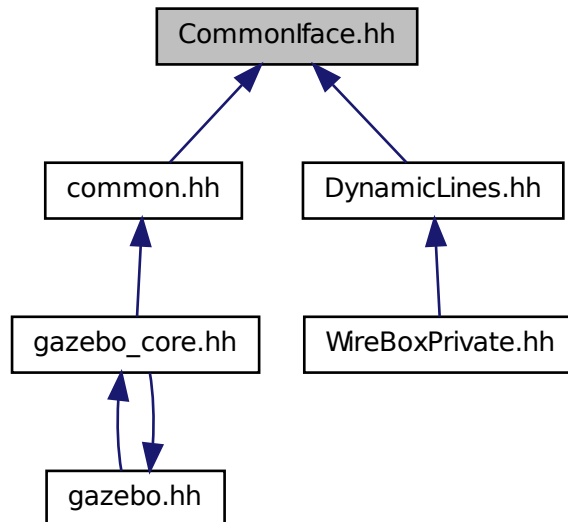
11.27 Commonface.hh File Reference

```
#include <string>
#include <vector>
#include <boost/uuid/sha1.hpp>
#include <iomanip>
#include <sstream>
#include "gazebo/util/system.hh"
```

Include dependency graph for Commonface.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

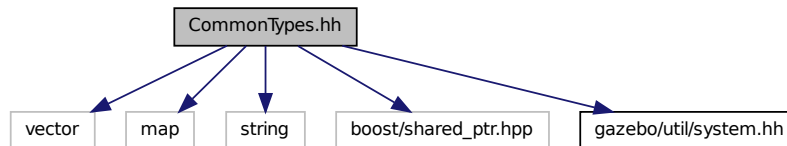
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

Functions

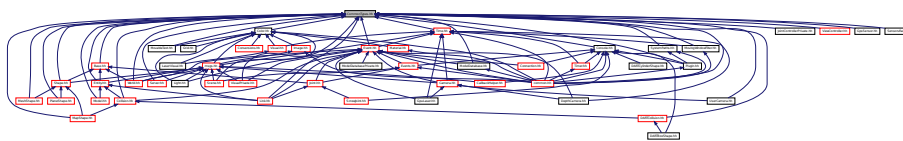
- **GAZEBO_VISIBLE** void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)
*add path suffix to **common::SystemPaths** (p. 1092)*
- **GAZEBO_VISIBLE** std::string **gazebo::common::find_file** (const std::string &_file)
*search for file in **common::SystemPaths** (p. 1092)*
- **GAZEBO_VISIBLE** std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath)
*search for file in **common::SystemPaths** (p. 1092)*
- **GAZEBO_VISIBLE** std::string **gazebo::common::find_file_path** (const std::string &_file)
*search for a file in **common::SystemPaths** (p. 1092)*
- template<typename T >
GAZEBO_VISIBLE std::string **gazebo::common::get_sha1** (const T &_buffer)
Compute the SHA1 hash of an array of bytes.
- **GAZEBO_VISIBLE** void **gazebo::common::load** ()
Load the common library.

11.28 CommonTypes.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
#include "gazebo/util/system.hh"
Include dependency graph for CommonTypes.hh:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::event**
Event (p. 416) namespace.

Macros

- #define **GAZEBO_DEPRECATED**(version) ()
- #define **GAZEBO_FORCEINLINE**
- #define **NULL** 0

Typedefs

- typedef boost::shared_ptr
< Animation > **gazebo::common::AnimationPtr**
- typedef std::vector
< ConnectionPtr > **gazebo::event::Connection_V**

- typedef boost::shared_ptr
< Connection > **gazebo::event::ConnectionPtr**
- typedef boost::shared_ptr
< DiagnosticTimer > **gazebo::common::DiagnosticTimerPtr**
- typedef boost::shared_ptr
< GUIPlugin > **gazebo::GUIPluginPtr**
- typedef boost::shared_ptr
< ModelPlugin > **gazebo::ModelPluginPtr**
- typedef boost::shared_ptr
< NumericAnimation > **gazebo::common::NumericAnimationPtr**
- typedef std::vector
< common::Param * > **gazebo::common::Param_V**
- typedef boost::shared_ptr
< PoseAnimation > **gazebo::common::PoseAnimationPtr**
- typedef boost::shared_ptr
< SensorPlugin > **gazebo::SensorPluginPtr**
- typedef boost::shared_ptr
< SphericalCoordinates > **gazebo::common::SphericalCoordinatesPtr**
- typedef std::map< std::string,
std::string > **gazebo::common::StrStr_M**
- typedef boost::shared_ptr
< SystemPlugin > **gazebo::SystemPluginPtr**
- typedef boost::shared_ptr
< VisualPlugin > **gazebo::VisualPluginPtr**
- typedef boost::shared_ptr
< WorldPlugin > **gazebo::WorldPluginPtr**

Variables

- static const double **gazebo::common::SpeedOfLight** = 299792458
Speed of light.

11.28.1 Detailed Description

11.28.2 Macro Definition Documentation

11.28.2.1 **#define GAZEBO_DEPRECATED(*version*)**

11.28.2.2 **#define GAZEBO_FORCEINLINE**

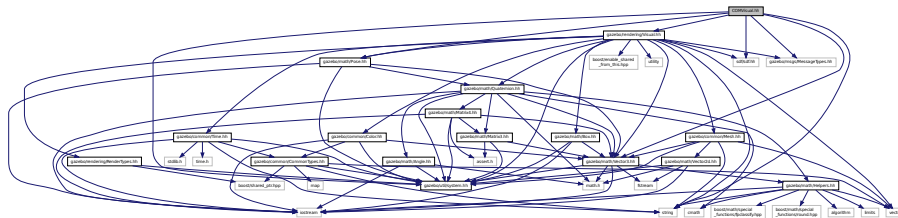
11.28.2.3 **#define NULL 0**

Referenced by gazebo::transport::TopicManager::Advertise(), gazebo::event::EventT< T >::Disconnect(), gazebo::transport::PublishTask::execute(), gazebo::transport::ConnectionReadTask::execute(), gazebo::common::get_sha1(), gazebo::transport::CallbackHelperT< M >::GetMsgType(), gazebo::transport::SubscribeOptions::Init(), gazebo::PluginT< ModelPlugin >::PluginT(), gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::DARTCylinderShape::SetSize(), gazebo::physics::DARTBoxShape::SetSize(), and gazebo::common::MovingWindowFilter< T >::~MovingWindowFilter().

11.29 COMVisual.hh File Reference

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for COMVisual.hh:



Classes

- class **gazebo::rendering::COMVisual**

Basic Center of Mass visualization.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

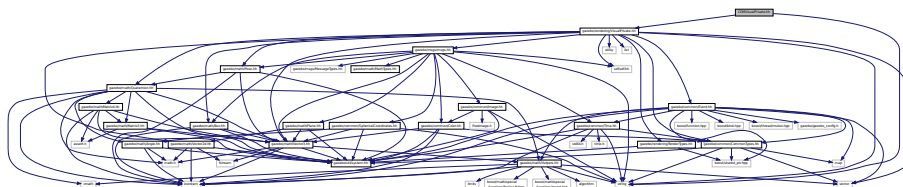
- namespace **gazebo::rendering**

Rendering namespace.

11.30 COMVisualPrivate.hh File Reference

```
#include <string>
#include "gazebo/rendering/VisualPrivate.hh"
```

Include dependency graph for COMVisualPrivate.hh:



Classes

- class **gazebo::rendering::COMVisualPrivate**

Private data for the COM Visual (p. 1196) class.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

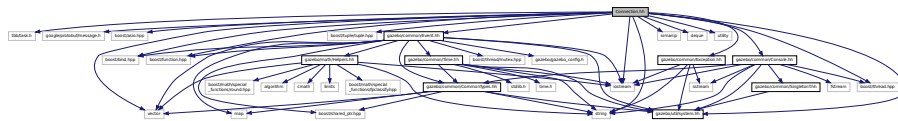
Rendering namespace.

- namespace **ogre**

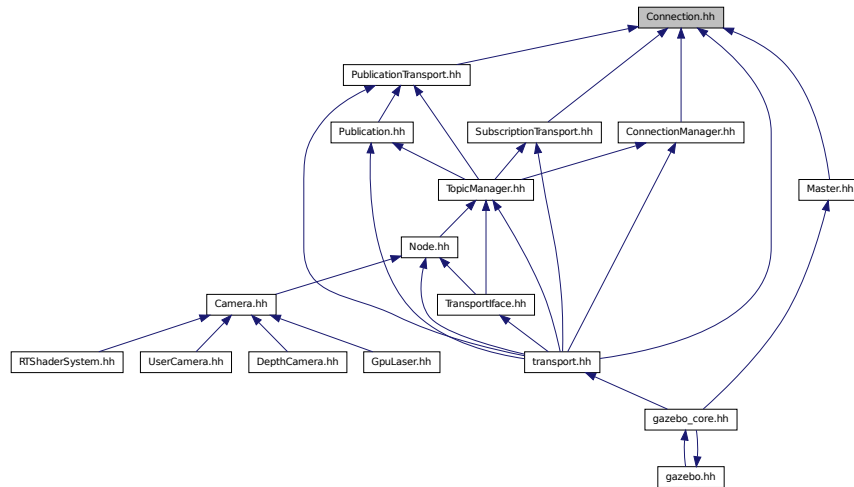
11.31 Connection.hh File Reference

```
#include <tbb/task.h>
#include <google/protobuf/message.h>
#include <boost/asio.hpp>
#include <boost/bind.hpp>
#include <boost/function.hpp>
#include <boost/thread.hpp>
#include <boost/tuple/tuple.hpp>
#include <string>
#include <vector>
#include <iostream>
#include <iomanip>
#include <deque>
#include <utility>
#include "gazebo/common/Event.hh"
#include "gazebo/common/Console.hh"
#include "gazebo/common/Exception.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Connection.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Connection**
Single TCP/IP connection manager.
- class **gazebo::transport::ConnectionReadTask**

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

Macros

- #define **HEADER_LENGTH** 8

Typedefs

- typedef boost::shared_ptr
< Connection > **gazebo::transport::ConnectionPtr**

Functions

- bool **gazebo::transport::is_stopped** ()
Is the transport system stopped?

11.31.1 Macro Definition Documentation

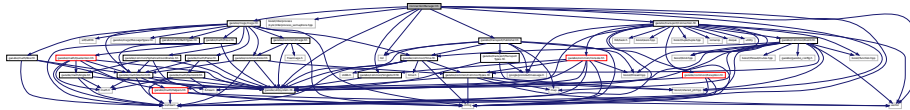
11.31.1.1 #define HEADER_LENGTH 8

Referenced by gazebo::transport::Connection::AsyncRead().

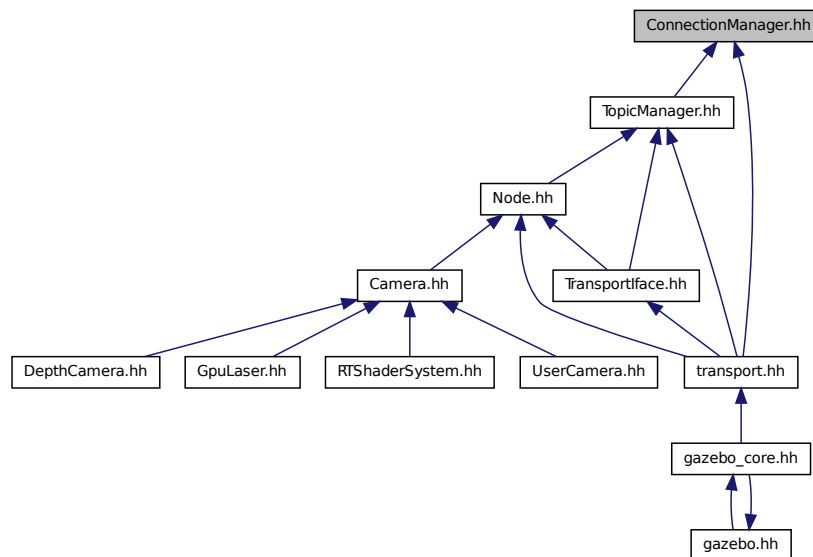
11.32 ConnectionManager.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <boost/interprocess/sync/interprocess_semaphore.hpp>
#include <string>
#include <list>
#include <vector>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/transport/Publisher.hh"
#include "gazebo/transport/Connection.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for ConnectionManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::ConnectionManager**
Manager of connections.

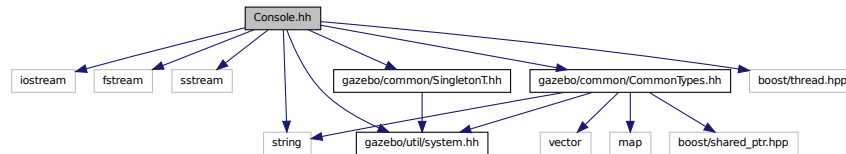
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

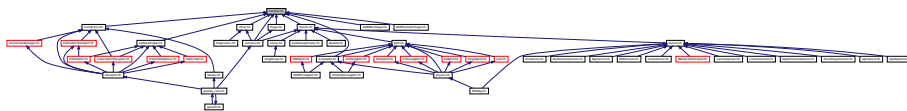
11.33 Console.hh File Reference

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <boost/thread.hpp>
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Console.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Logger::Buffer**
String buffer for the base logger.
- class **gazebo::common::FileLogger::Buffer**
String buffer for the file logger.
- class **gazebo::common::Console**
Container for loggers, and global logging options (such as verbose vs.
- class **gazebo::common::FileLogger**

A logger that outputs messages to a file.

- class **gazebo::common::Logger**

Terminal logger.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

Common namespace.

Macros

- #define **gzdbg** (**gazebo::common::Console::dbg**(__FILE__, __LINE__))

Output a debug message.

- #define **gzerr** (**gazebo::common::Console::err**(__FILE__, __LINE__))

Output an error message.

- #define **gzlog** (**gazebo::common::Console::log**())

Output a message to a log file.

- #define **gzLogInit**(_str) (**gazebo::common::Console::log**.Init(_str))

Initialize log file with filename given by _str.

- #define **gzmsg** (**gazebo::common::Console::msg**())

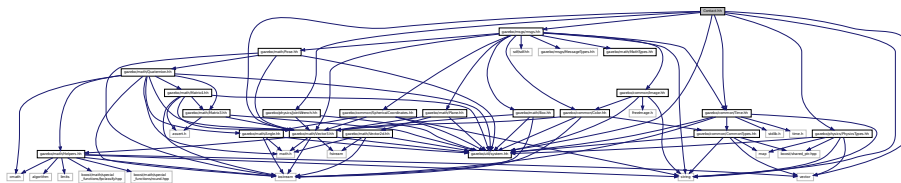
- #define **gzwarn** (**gazebo::common::Console::warn**(__FILE__, __LINE__))

Output a warning message.

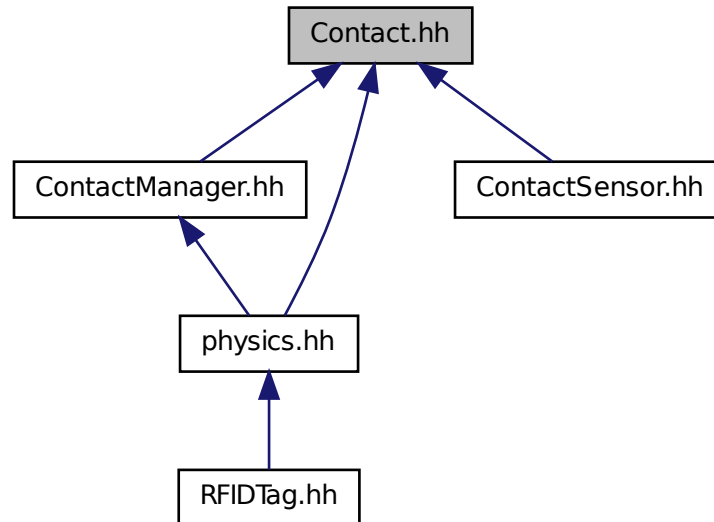
11.34 Contact.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/JointWrench.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Contact.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Contact**
A contact between two collisions.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Macros

- #define **MAX_COLLIDE_RETURNS** 250
- #define **MAX_CONTACT_JOINTS** 32

11.34.1 Macro Definition Documentation

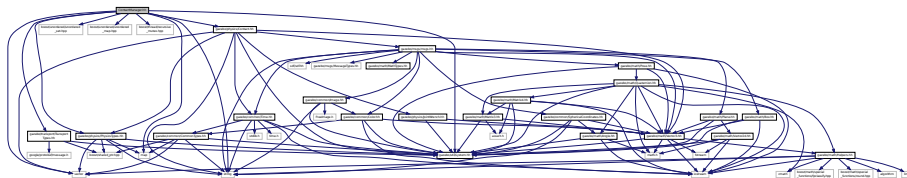
11.34.1.1 #define MAX_COLLIDE_RETURNS 250

11.34.1.2 #define MAX_CONTACT_JOINTS 32

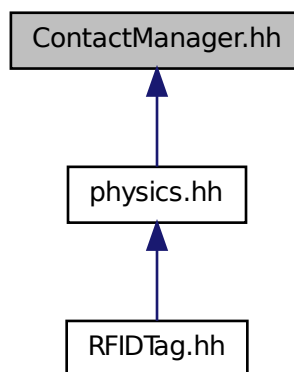
11.35 ContactManager.hh File Reference

```
#include <vector>
#include <string>
#include <map>
#include <boost/unordered/unordered_set.hpp>
#include <boost/unordered/unordered_map.hpp>
#include <boost/thread/recursive_mutex.hpp>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Contact.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for ContactManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ContactManager**
Aggregates all the contact information generated by the collision detection engine.
- class **gazebo::physics::ContactPublisher**
*A custom contact publisher created for each contact filter in the **Contact** (p. 279) Manager.*

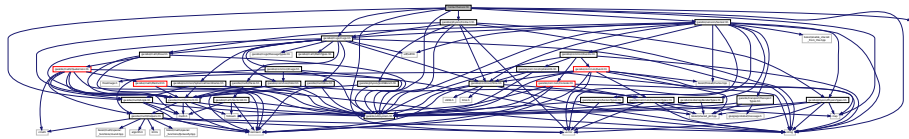
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.36 ContactSensor.hh File Reference

```
#include <vector>
#include <map>
#include <list>
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/physics/Contact.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for ContactSensor.hh:



Classes

- class **gazebo::sensors::ContactSensor**
Contact sensor.

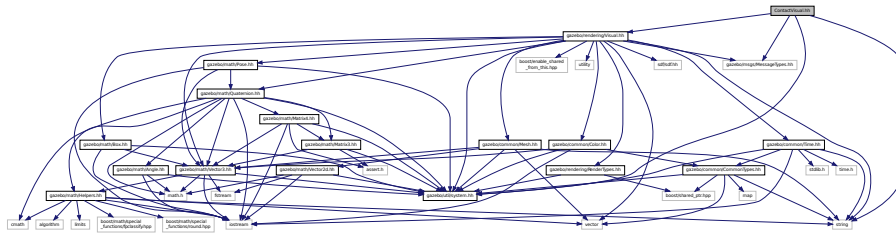
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.37 ContactVisual.hh File Reference

```
#include <string>
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for ContactVisual.hh:



Classes

- class **gazebo::rendering::ContactVisual**
Contact visualization.

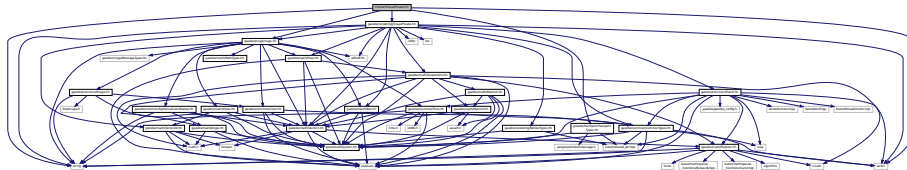
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.38 ContactVisualPrivate.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/VisualPrivate.hh"
```

Include dependency graph for ContactVisualPrivate.hh:



Classes

- class **gazebo::rendering::ContactVisualPrivate::ContactPoint**
A contact point visualization.
- class **gazebo::rendering::ContactVisualPrivate**
*Private data for the Arrow **Visual** (p. 1196) class.*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

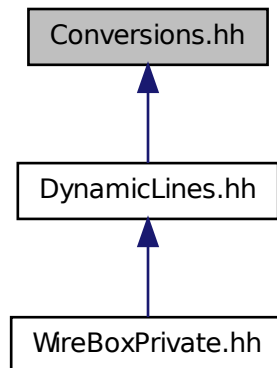
11.39 Conversions.hh File Reference

```
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/common/Color.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Quaternion.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Conversions.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::Conversions**
Conversions (p. 296) *Conversions.hh* (p. 1306) *rendering/Conversions.hh* (p. 1306).

Namespaces

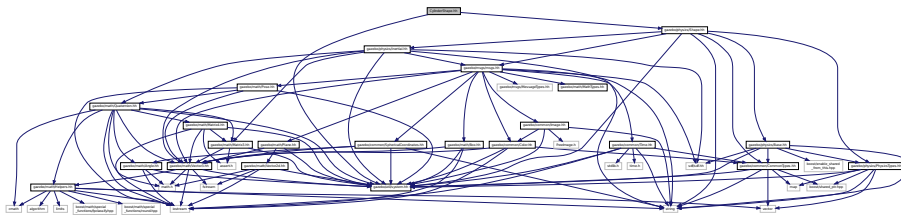
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.40 CylinderShape.hh File Reference

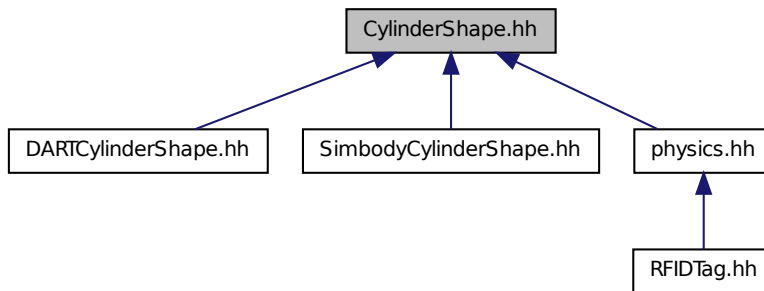
```
#include "gazebo/physics/Shape.hh"
```

```
#include "gazebo/util/system.hh"
```

Include dependency graph for CylinderShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::CylinderShape**
Cylinder collision.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

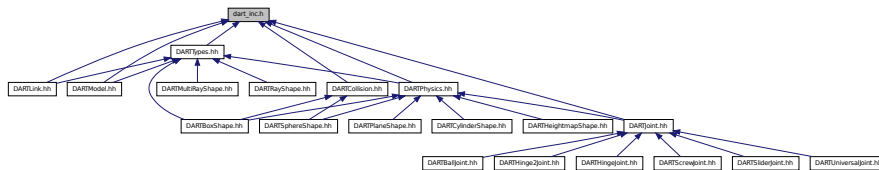
11.41 dart_inc.h File Reference

```
#include <dart/math/Helpers.h>
#include <dart/math/Geometry.h>
#include <dart/collision/CollisionDetector.h>
#include <dart/collision/dart/DARTCollisionDetector.h>
#include <dart/integration/Integrator.h>
#include <dart/integration/EulerIntegrator.h>
#include <dart/integration/RK4Integrator.h>
#include <dart/dynamics/BallJoint.h>
#include <dart/dynamics/BodyNode.h>
#include <dart/dynamics/BoxShape.h>
#include <dart/dynamics/CylinderShape.h>
#include <dart/dynamics/EllipsoidShape.h>
#include <dart/dynamics/FreeJoint.h>
#include <dart/dynamics/GenCoord.h>
#include <dart/dynamics/Joint.h>
#include <dart/dynamics/MeshShape.h>
#include <dart/dynamics/PrismaticJoint.h>
#include <dart/dynamics/RevoluteJoint.h>
#include <dart/dynamics/Shape.h>
#include <dart/dynamics/Skeleton.h>
#include <dart/dynamics/ScrewJoint.h>
#include <dart/dynamics/UniversalJoint.h>
#include <dart/dynamics/WeldJoint.h>
#include <dart/constraint/Constraint.h>
#include <dart/constraint/ConstraintDynamics.h>
#include <dart/simulation/World.h>
```

Include dependency graph for dart_inc.h:



This graph shows which files directly or indirectly include this file:



11.42 DARTBallJoint.hh File Reference

```
#include "gazebo/physics/BallJoint.hh"
```



```
#include "gazebo/physics/dart/DARTJoint.hh"
#include "gazebo/util/system.hh"
Include dependency graph for DARTBallJoint.hh:
```



Classes

- class **gazebo::physics::DARTBallJoint**
An *DARTBallJoint* (p. 301).

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.43 DARTBoxShape.hh File Reference

```
#include "gazebo/common/Console.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/dart/DARTPhysics.hh"
#include "gazebo/physics/dart/DARTTypes.hh"
#include "gazebo/physics/dart/DARTCollision.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/BoxShape.hh"
#include "gazebo/util/system.hh"
Include dependency graph for DARTBoxShape.hh:
```



Classes

- class **gazebo::physics::DARTBoxShape**
DART Box shape.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

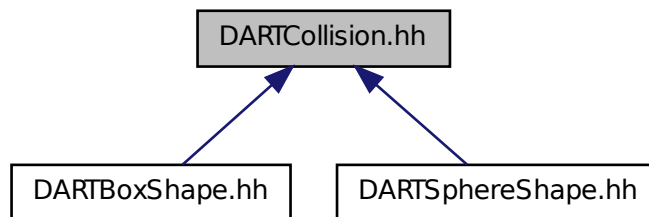
11.44 DARTCollision.hh File Reference

```
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/physics/dart/dart_inc.h"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTCollision.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::DARTCollision**

Base (p. 168) class for all DART collisions.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

11.45 DARTCylinderShape.hh File Reference

```
#include "gazebo/common/Console.hh"
#include "gazebo/physics/CylinderShape.hh"
#include "gazebo/physics/dart/DARTPhysics.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTCylinderShape.hh:



Classes

- class **gazebo::physics::DARTCylinderShape**
DART cylinder shape.

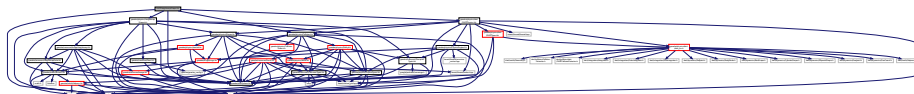
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.46 DARTHeightmapShape.hh File Reference

```
#include <vector>
#include "gazebo/physics/HeightmapShape.hh"
#include "gazebo/physics/dart/DARTPhysics.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTHeightmapShape.hh:



Classes

- class **gazebo::physics::DARTHeightmapShape**
DART Height map collision.

Namespaces

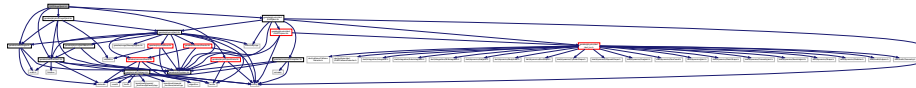
- namespace **gazebo**
Forward declarations for the common classes.

- namespace **gazebo::physics**
namespace for physics

11.47 DARTHinge2Joint.hh File Reference

```
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/Hinge2Joint.hh"
#include "gazebo/physics/dart/DARTJoint.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTHinge2Joint.hh:



Classes

- class **gazebo::physics::DARTHinge2Joint**
A two axis hinge joint.

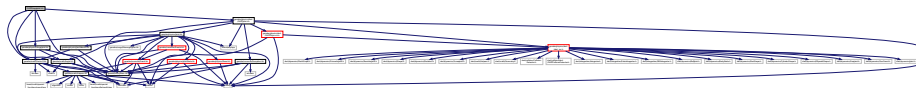
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.48 DARTHingeJoint.hh File Reference

```
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/HingeJoint.hh"
#include "gazebo/physics/dart/DARTJoint.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTHingeJoint.hh:



Classes

- class **gazebo::physics::DARTHingeJoint**
A single axis hinge joint.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

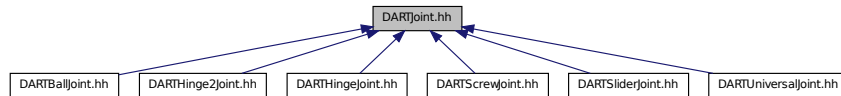
11.49 DARTJoint.hh File Reference

```
#include <boost/any.hpp>
#include <string>
#include "gazebo/common/Exception.hh"
#include "gazebo/physics/Joint.hh"
#include "gazebo/physics/dart/dart_inc.h"
#include "gazebo/physics/dart/DARTPhysics.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::DARTJoint**
DART joint interface.

Namespaces

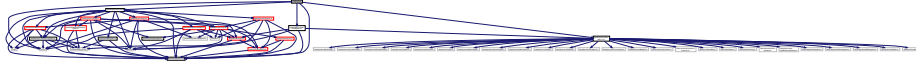
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.50 DARTLink.hh File Reference

```
#include <vector>
```

```
#include "gazebo/physics/Link.hh"
#include "gazebo/physics/dart/dart_inc.h"
#include "gazebo/physics/dart/DARTTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTLink.hh:



Classes

- class **gazebo::physics::DARTLink**
DART Link (p. 595) class.

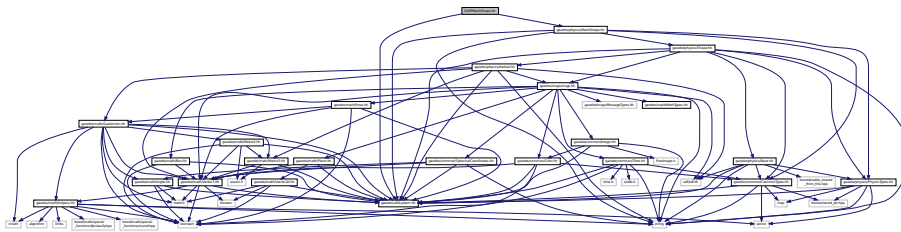
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.51 DARTMeshShape.hh File Reference

```
#include "gazebo/physics/MeshShape.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTMeshShape.hh:



Classes

- class **gazebo::physics::DARTMeshShape**
Triangle mesh collision.

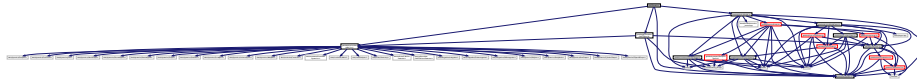
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.52 DARTModel.hh File Reference

```
#include "gazebo/physics/dart/dart_inc.h"
#include "gazebo/physics/dart/DARTTypes.hh"
#include "gazebo/physics/Model.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTModel.hh:



Classes

- class **gazebo::physics::DARTModel**
DART model class.

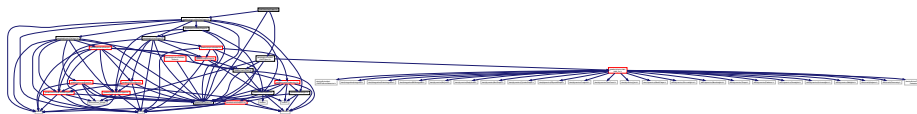
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.53 DARTMultiRayShape.hh File Reference

```
#include "gazebo/physics/MultiRayShape.hh"
#include "gazebo/physics/dart/DARTTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTMultiRayShape.hh:



Classes

- class **gazebo::physics::DARTMultiRayShape**
*DART specific version of **MultiRayShape** (p. 723).*

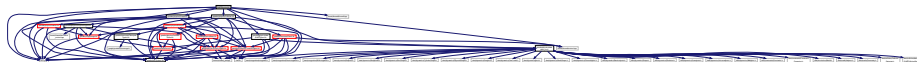
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

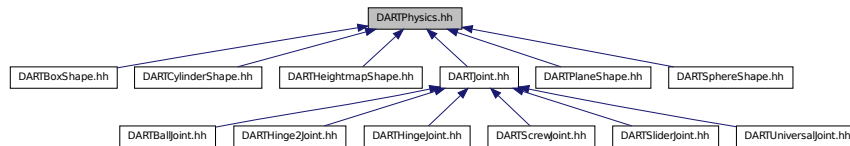
11.54 DARTPhysics.hh File Reference

```
#include <string>
#include <boost/thread/thread.hpp>
#include <boost/thread/mutex.hpp>
#include "gazebo/physics/PhysicsEngine.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/physics/dart/dart_inc.h"
#include "gazebo/physics/dart/DARTTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTPhysics.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::DARTPhysics**

DART physics engine.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.55 DARTPlaneShape.hh File Reference

```
#include "gazebo/physics/PlaneShape.hh"
#include "gazebo/physics/dart/DARTPhysics.hh"
#include "gazebo/util/system.hh"
```


Include dependency graph for DARTPlaneShape.hh:



Classes

- class **gazebo::physics::DARTPlaneShape**
An DART Plane shape.

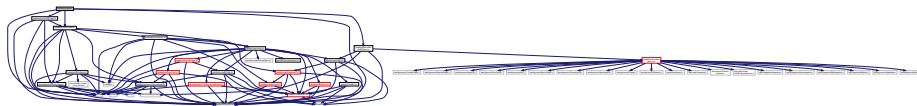
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.56 DARTRayShape.hh File Reference

```
#include <string>
#include "gazebo/physics/RayShape.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/physics/dart/DARTTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTRayShape.hh:



Classes

- class **gazebo::physics::DARTRayShape**
Ray collision.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.57 DARTScrewJoint.hh File Reference

```
#include "gazebo/physics/ScrewJoint.hh"  
#include "gazebo/physics/dart/DARTJoint.hh"  
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTScrewJoint.hh:



Classes

- class **gazebo::physics::DARTScrewJoint**
A screw joint.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.58 DARTSliderJoint.hh File Reference

```
#include "gazebo/physics/SliderJoint.hh"  
#include "gazebo/physics/dart/DARTJoint.hh"  
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTSliderJoint.hh:



Classes

- class **gazebo::physics::DARTSliderJoint**
A slider joint.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.59 DARTSphereShape.hh File Reference

```
#include "gazebo/physics/dart/DARTPhysics.hh"
#include "gazebo/physics/dart/DARTCollision.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/SphereShape.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTSphereShape.hh:



Classes

- class **gazebo::physics::DARTSphereShape**

A DART sphere shape.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

11.60 DARTTypes.hh File Reference

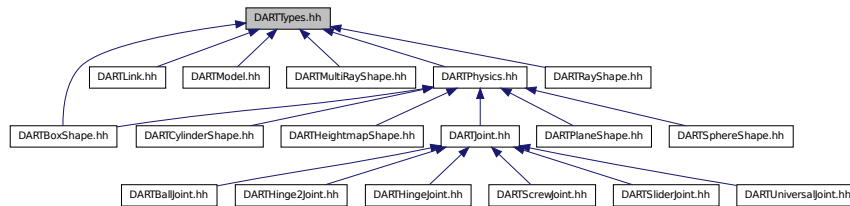
DART wrapper forward declarations and typedefs.

```
#include <boost/shared_ptr.hpp>
#include "gazebo/math/Pose.hh"
#include "gazebo/physics/dart/dart_inc.h"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTTypes.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::DARTTypes**
A set of functions for converting between the math types used by gazebo and dart.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Typedefs

- typedef boost::shared_ptr
 < DARTCollision > **gazebo::physics::DARTCollisionPtr**
- typedef boost::shared_ptr
 < DARTJoint > **gazebo::physics::DARTJointPtr**
- typedef boost::shared_ptr
 < DARTLink > **gazebo::physics::DARTLinkPtr**
- typedef boost::shared_ptr
 < DARTModel > **gazebo::physics::DARTModelPtr**
- typedef boost::shared_ptr
 < DARTPhysics > **gazebo::physics::DARTPhysicsPtr**
- typedef boost::shared_ptr
 < DARTRayShape > **gazebo::physics::DARTRayShapePtr**

11.60.1 Detailed Description

DART wrapper forward declarations and typedefs.

11.61 DARTUniversalJoint.hh File Reference

```
#include "gazebo/physics/UniversalJoint.hh"
#include "gazebo/physics/dart/DARTJoint.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DARTUniversalJoint.hh:



Classes

- class **gazebo::physics::DARTUniversalJoint**

A universal joint.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

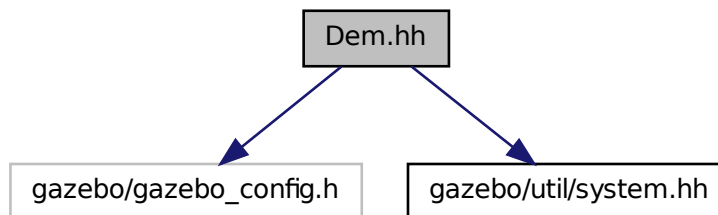
namespace for physics

11.62 Dem.hh File Reference

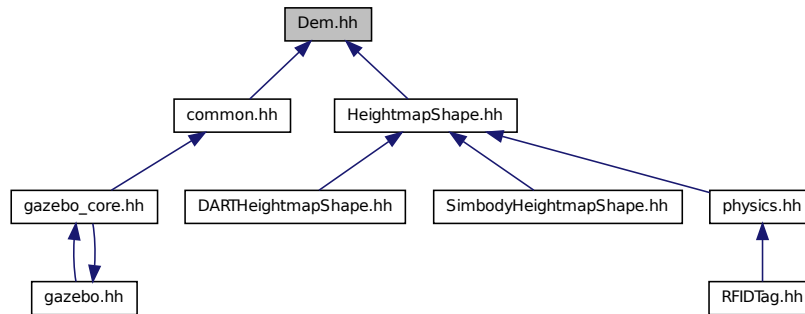
```
#include <gazebo/gazebo_config.h>
```

```
#include <gazebo/util/system.hh>
```

Include dependency graph for Dem.hh:



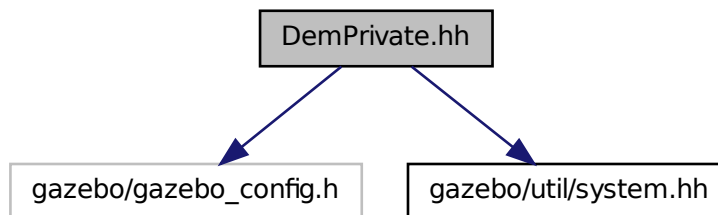
This graph shows which files directly or indirectly include this file:



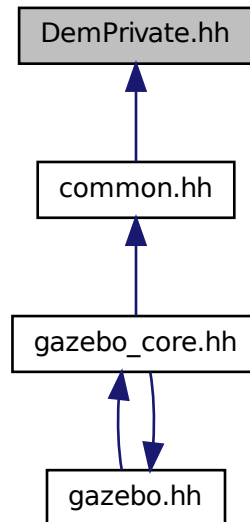
11.63 DemPrivate.hh File Reference

```
#include <gazebo/gazebo_config.h>
#include <gazebo/util/system.hh>
```

Include dependency graph for `DemPrivate.hh`:



This graph shows which files directly or indirectly include this file:



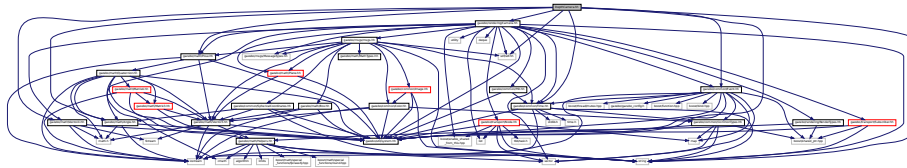
11.64 DepthCamera.hh File Reference

```

#include <string>
#include <sdf/sdf.hh>
#include "gazebo/common/Event.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Vector2i.hh"
#include "gazebo/rendering/Camera.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for DepthCamera.hh:



Classes

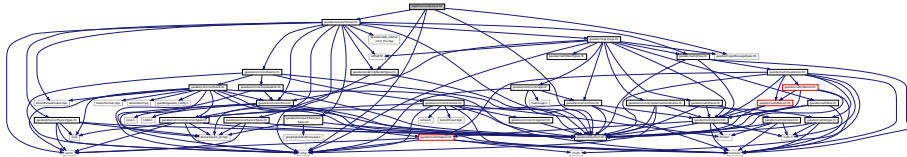
- class **gazebo::rendering::DepthCamera**
Depth camera used to render depth data into an image buffer.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.65 DepthCameraSensor.hh File Reference

```
#include <string>
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"
Include dependency graph for DepthCameraSensor.hh:
```



Classes

- class **gazebo::sensors::DepthCameraSensor**

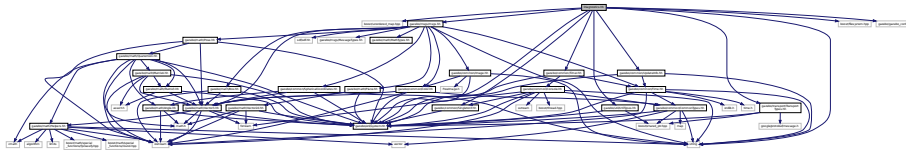
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.66 Diagnostics.hh File Reference

```
#include <boost/unordered_map.hpp>
#include <string>
#include <boost/filesystem.hpp>
#include "gazebo/gazebo_config.h"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/Timer.hh"
#include "gazebo/util/UtilTypes.hh"
#include "gazebo/util/system.hh"
```


Include dependency graph for Diagnostics.hh:



Classes

- class **gazebo::util::DiagnosticManager**
A diagnostic manager class.
- class **gazebo::util::DiagnosticTimer**
A timer designed for diagnostics.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::util**

Macros

- #define **DIAG_TIMER_LAP**(_name, _prefix) ((void)0)
- #define **DIAG_TIMER_START**(_name) ((void) 0)
- #define **DIAG_TIMER_STOP**(_name) ((void) 0)

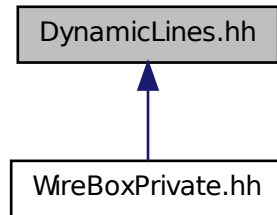
11.67 DynamicLines.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/common/CommonIface.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/rendering/Conversions.hh"
#include "gazebo/rendering/DynamicRenderable.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DynamicLines.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::DynamicLines**

Class for drawing lines that can change.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

Rendering namespace.

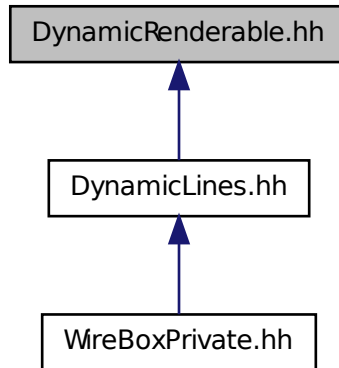
11.68 DynamicRenderable.hh File Reference

```
#include <string>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for DynamicRenderable.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::DynamicRenderable**

Abstract base class providing mechanisms for dynamically growing hardware buffers.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

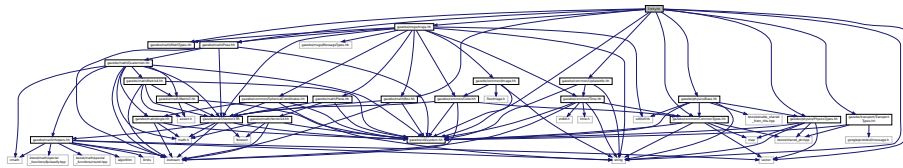
- namespace **gazebo::rendering**

Rendering namespace.

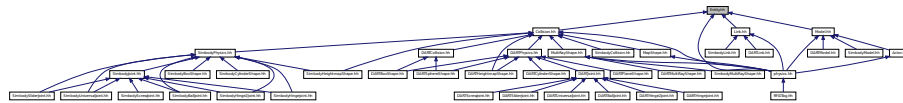
11.69 Entity.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/math/MathTypes.hh"
#include "gazebo/math/Box.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Base.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Entity.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Entity**
Base (p. 168) class for all physics objects in Gazebo.

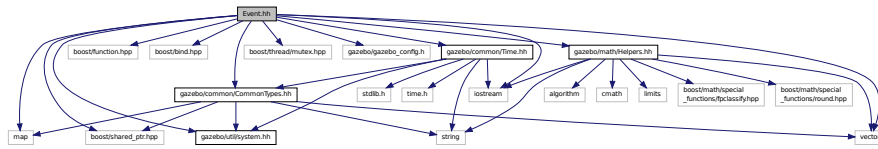
Namespaces

- namespace **boost**
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.70 Event.hh File Reference

```
#include <iostream>
#include <vector>
#include <map>
#include <boost/function.hpp>
#include <boost/bind.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/thread/mutex.hpp>
#include <gazebo/gazebo_config.h>
#include <gazebo/common/Time.hh>
#include <gazebo/common/CommonTypes.hh>
#include <gazebo/math/Helpers.hh>
#include "gazebo/util/system.hh"
```

Include dependency graph for Event.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::event::Connection**
A class that encapsulates a connection.
- class **gazebo::event::ConnectionPrivate**
- class **gazebo::event::Event**
Base class for all events.
- class **gazebo::event::EventPrivate**
- class **gazebo::event::EventT < T >**
A class for event processing.
- class **gazebo::event::EventTPrivate < T >**

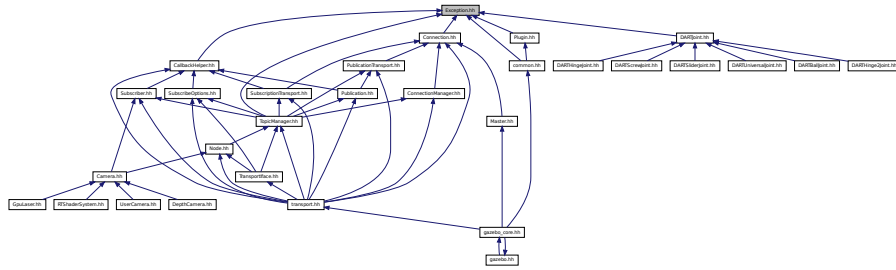
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::event**
Event (p. 416) namespace.

11.71 Events.hh File Reference

```
#include <string>
#include "gazebo/common/Console.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/util/system.hh"
```


This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::AssertionInternalError**

Class for generating Exceptions which come from gazebo assertions.

- class **gazebo::common::Exception**

Class for generating exceptions.

- class **gazebo::common::InternalError**

Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

Common namespace.

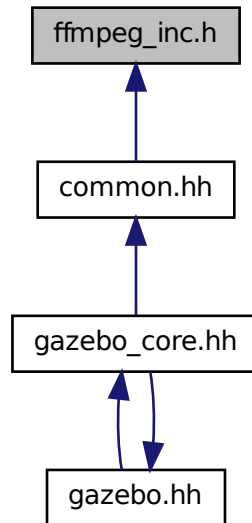
Macros

- **#define gzthrow(msg)**

This macro logs an error to the throw stream and throws an exception that contains the file name and line number.

11.73 ffmpeg_inc.h File Reference

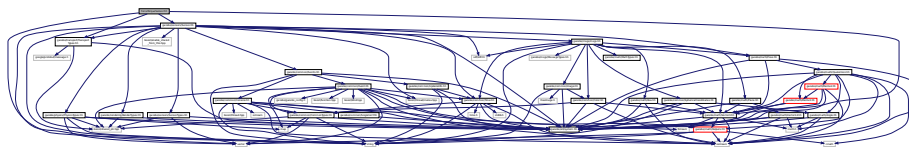
This graph shows which files directly or indirectly include this file:



11.74 ForceTorqueSensor.hh File Reference

```
#include <string>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for ForceTorqueSensor.hh:



Classes

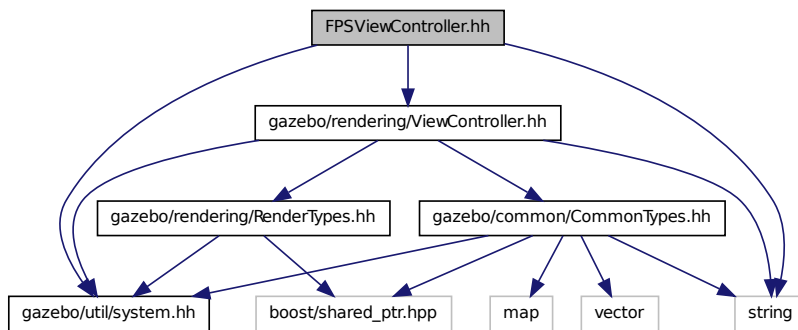
- class `gazebo::sensors::ForceTorqueSensor`
Sensor (p. 907) for measure force and torque on a joint.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.75 FPSViewController.hh File Reference

```
#include <string>
#include "gazebo/rendering/ViewController.hh"
#include "gazebo/util/system.hh"
Include dependency graph for FPSViewController.hh:
```



Classes

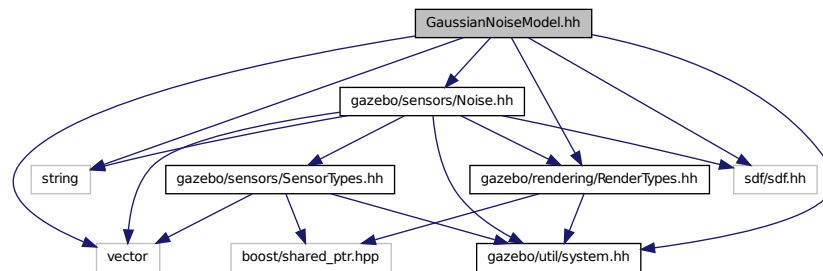
- class **gazebo::rendering::FPSViewController**
First Person Shooter style view controller.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.76 GaussianNoiseModel.hh File Reference

```
#include <vector>
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/sensors/Noise.hh"
#include "gazebo/util/system.hh"
Include dependency graph for GaussianNoiseModel.hh:
```



Classes

- class **gazebo::sensors::GaussianNoiseModel**
Gaussian noise class.
- class **gazebo::sensors::ImageGaussianNoiseModel**

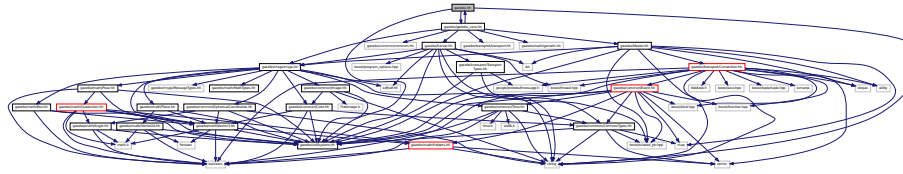
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.
- namespace **Ogre**

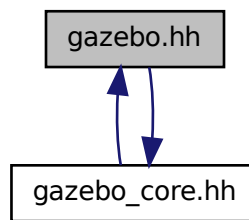
11.77 gazebo.hh File Reference

```
#include <gazebo/gazebo_core.hh>
#include <string>
#include "gazebo/util/system.hh"
```

Include dependency graph for gazebo.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.

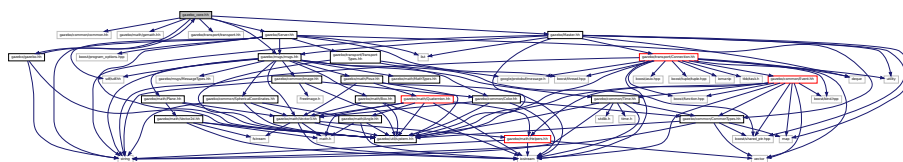
Functions

- **GAZEBO_VISIBLE** void **gazebo::add_plugin** (const std::string &_filename) **GAZEBO_DEPRECATED**(2.3)
Deprecated.
- **GAZEBO_VISIBLE** void **gazebo::addPlugin** (const std::string &_filename)
Add a system plugin.
- **GAZEBO_VISIBLE** std::string **gazebo::find_file** (const std::string &_file) **GAZEBO_DEPRECATED**(2.3)
Not implemented.
- **GAZEBO_VISIBLE** void **gazebo::fini** () **GAZEBO_DEPRECATED**(2.3)
Deprecated.
- **GAZEBO_VISIBLE** bool **gazebo::init** () **GAZEBO_DEPRECATED**(2.3)
Deprecated.
- **GAZEBO_VISIBLE** bool **gazebo::load** (int _argc=0, char **_argv=0) **GAZEBO_DEPRECATED**(2.3)
Deprecated.
- **GAZEBO_VISIBLE**
gazebo::physics::WorldPtr gazebo::loadWorld (const std::string &_worldFile)
Create and load a new world from an SDF world file.

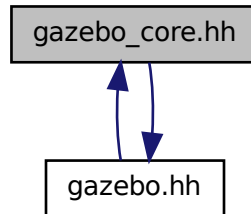
- **GAZEBO_VISIBLE** void **gazebo::print_version** () **GAZEBO_DEPRECATED(2.3)**
Deprecated.
- **GAZEBO_VISIBLE** void **gazebo::printVersion** ()
Output version information to the terminal.
- **GAZEBO_VISIBLE** void **gazebo::run** () **GAZEBO_DEPRECATED(2.3)**
Deprecated.
- **GAZEBO_VISIBLE** void **gazebo::runWorld** (**gazebo::physics::WorldPtr** _world, unsigned int _iterations)
Run a world for a specific number of iterations.
- **GAZEBO_VISIBLE** bool **gazebo::setupClient** (int _argc=0, char **_argv=0)
Start a gazebo client.
- **GAZEBO_VISIBLE** bool **gazebo::setupServer** (int _argc=0, char **_argv=0)
Start a gazebo server.
- **GAZEBO_VISIBLE** bool **gazebo::shutdown** ()
Stop and cleanup simulation.
- **GAZEBO_VISIBLE** void **gazebo::stop** () **GAZEBO_DEPRECATED(2.3)**
Deprecated.

11.78 gazebo_core.hh File Reference

```
#include <gazebo/common/common.hh>
#include <gazebo/math/gzmath.hh>
#include <gazebo/messages/messages.hh>
#include <gazebo/transport/transport.hh>
#include <gazebo/Server.hh>
#include <gazebo/Master.hh>
#include <gazebo/gazebo.hh>
Include dependency graph for gazebo_core.hh:
```



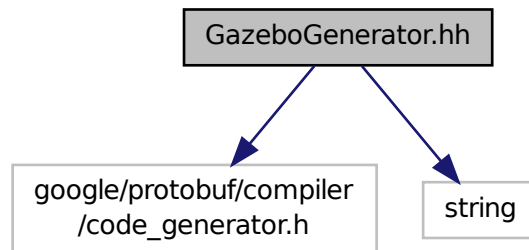
This graph shows which files directly or indirectly include this file:



11.79 GazeboGenerator.hh File Reference

```
#include <google/protobuf/compiler/code_generator.h>  
#include <string>
```

Include dependency graph for GazeboGenerator.hh:



Classes

- class **google::protobuf::compiler::cpp::GazeboGenerator**
*Google protobuf message generator for **gazebo::msgs** (p. 113).*

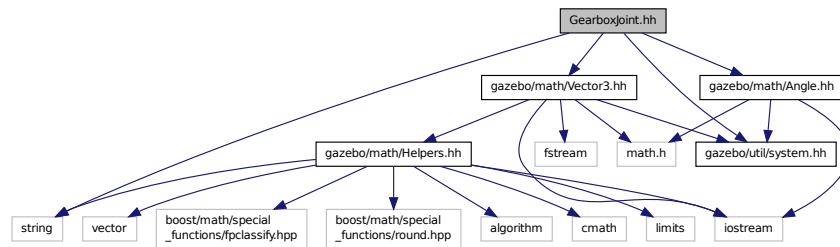
Namespaces

- namespace **google**
- namespace **google::protobuf**
- namespace **google::protobuf::compiler**
- namespace **google::protobuf::compiler::cpp**

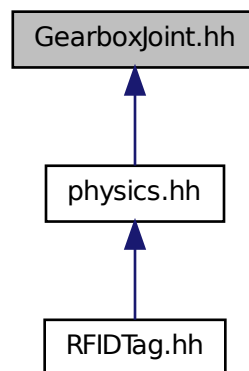
11.80 GearboxJoint.hh File Reference

```
#include <string>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for GearboxJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::GearboxJoint**< T >
A double axis gearbox joint.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.

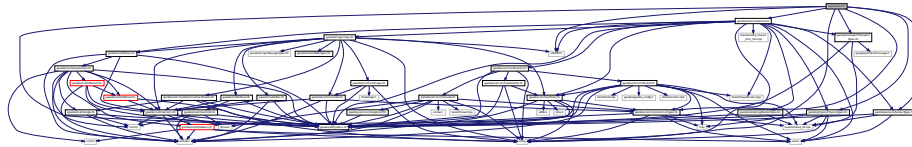
- namespace **gazebo::physics**

namespace for physics

11.81 GpsSensor.hh File Reference

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/SensorTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for GpsSensor.hh:



Classes

- class **gazebo::sensors::GpsSensor**
GpsSensor (p. 464) to provide position measurement.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.82 GpuLaser.hh File Reference

```
#include <string>
#include <vector>
#include <sdf/sdf.hh>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/rendering/Camera.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Vector2i.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for GpuLaser.hh:



Classes

- class **gazebo::rendering::GpuLaser**
GPU based laser distance sensor.

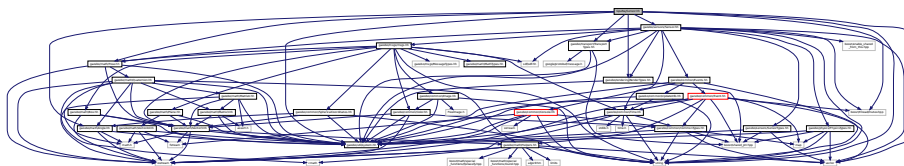
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.83 GpuRaySensor.hh File Reference

```
#include <vector>
#include <string>
#include <boost/thread/mutex.hpp>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for GpuRaySensor.hh:



Classes

- class **gazebo::sensors::GpuRaySensor**

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.84 Grid.hh File Reference

```
#include <stdint.h>
#include <vector>
#include <string>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/common/Color.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Grid.hh:



Classes

- class **gazebo::rendering::Grid**
Displays a grid of cells, drawn with lines.

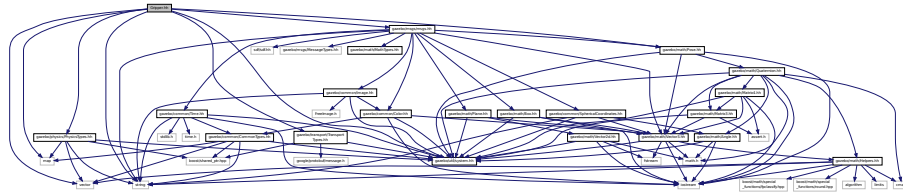
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

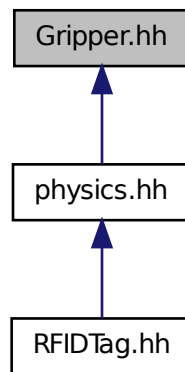
11.85 Gripper.hh File Reference

```
#include <map>
#include <vector>
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Gripper.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Gripper**
A gripper abstraction.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.86 GUIOverlay.hh File Reference

```
#include <string>
```


Classes

- class **gazebo::rendering::GUIOverlayPrivate**
*Private data for the **GUIOverlay** (p. 494) class.*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.88 Heightmap.hh File Reference

```
#include <string>
#include <vector>
#include <boost/filesystem.hpp>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/common/Image.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2d.hh"
#include "gazebo/rendering/Scene.hh"
#include "gazebo/util/system.hh"
Include dependency graph for Heightmap.hh:
```



Classes

- class **gazebo::rendering::DummyPageProvider**
Pretends to provide procedural page content to avoid page loading.
- class **gazebo::rendering::GzTerrainMatGen**
- class **gazebo::rendering::Heightmap**
Rendering a terrain using heightmap information.
- class **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg**
Keeping the CG shader for reference.
- class **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL**
Utility class to help with generating shaders for GLSL.
- class **gazebo::rendering::GzTerrainMatGen::SM2Profile**
Shader model 2 profile target.

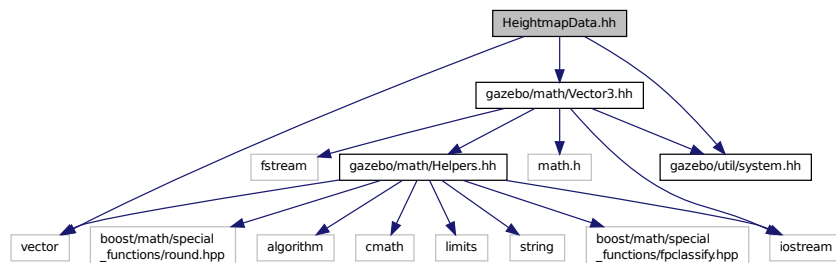
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.

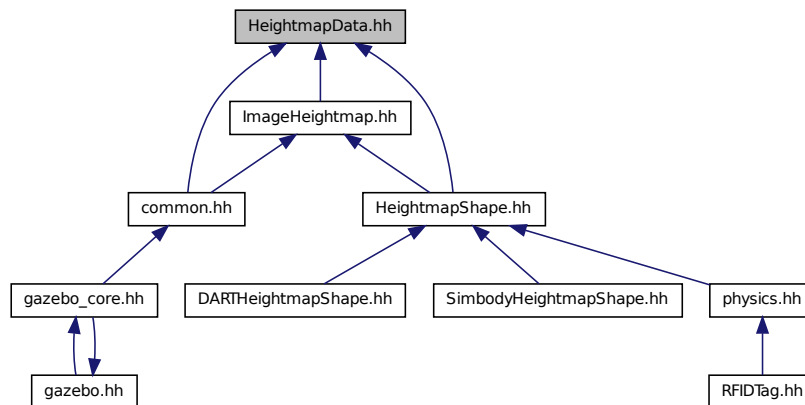
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.89 HeightmapData.hh File Reference

```
#include <vector>
#include "gazebo/math/Vector3.hh"
#include "gazebo/util/system.hh"
Include dependency graph for HeightmapData.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::HeightmapData**
Encapsulates a generic heightmap data file.

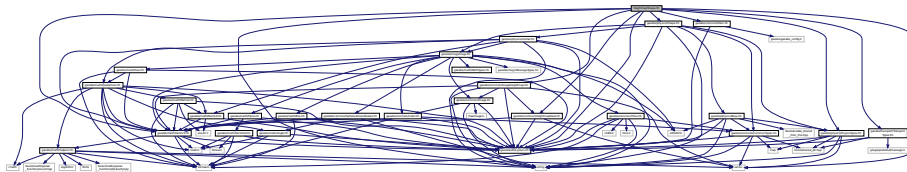
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

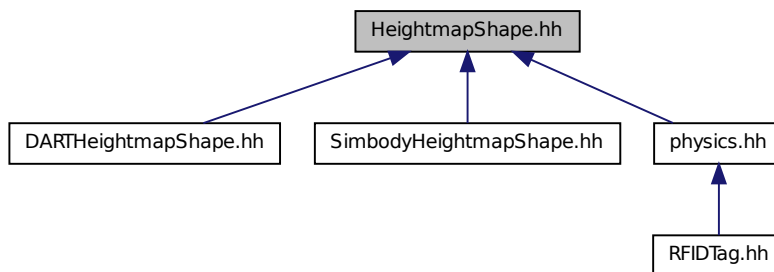
11.90 HeightmapShape.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/common/ImageHeightmap.hh"
#include "gazebo/common/HeightmapData.hh"
#include "gazebo/common/Dem.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for HeightmapShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::HeightmapShape**
HeightmapShape (p. 506) collision shape builds a heightmap from an image.

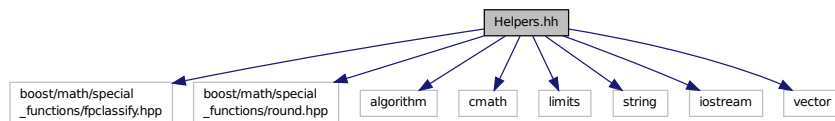
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

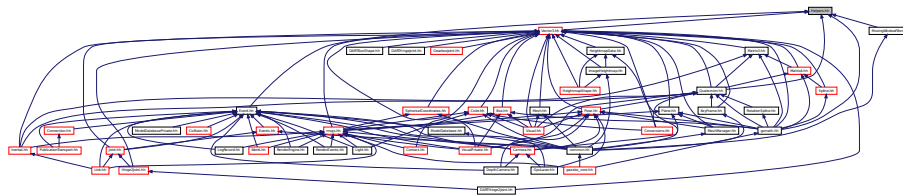
11.91 Helpers.hh File Reference

```
#include <boost/math/special_functions/fpclassify.hpp>
#include <boost/math/special_functions/round.hpp>
#include <algorithm>
#include <cmath>
#include <limits>
#include <string>
#include <iostream>
#include <vector>
```

Include dependency graph for Helpers.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

Macros

- **#define GZ_DBL_MAX** `std::numeric_limits<double>::max()`
Double maximum value.

- **#define GZ_DBL_MIN** std::numeric_limits<double>::min()
Double min value.
- **#define GZ_FLT_MAX** std::numeric_limits<float>::max()
Float maximum value.
- **#define GZ_FLT_MIN** std::numeric_limits<float>::min()
Float minimum value.
- **#define GZ_INT32_MAX** std::numeric_limits<int32_t>::max()
32bit integer maximum value
- **#define GZ_INT32_MIN** std::numeric_limits<int32_t>::min()
32bit integer minimum value
- **#define GZ_UINT32_MAX** std::numeric_limits<uint32_t>::max()
32bit unsigned integer maximum value
- **#define GZ_UINT32_MIN** std::numeric_limits<uint32_t>::min()
32bit unsigned integer minimum value

Functions

- template<typename T >
T gazebo::math::clamp (T _v, T _min, T _max)
Simple clamping function.
- template<typename T >
bool gazebo::math::equal (const T &_a, const T &_b, const T &_epsilon=1e-6)
check if two values are equal, within a tolerance
- float **gazebo::math::fixnan** (float _v)
Fix a nan value.
- double **gazebo::math::fixnan** (double _v)
Fix a nan value.
- bool **gazebo::math::isnan** (float _v)
check if a float is NaN
- bool **gazebo::math::isnan** (double _v)
check if a double is NaN
- bool **gazebo::math::isPowerOfTwo** (unsigned int _x)
is this a power of 2?
- template<typename T >
T gazebo::math::max (const std::vector< T > &_values)
get the maximum value of vector of values
- template<typename T >
T gazebo::math::mean (const std::vector< T > &_values)
get mean of vector of values
- template<typename T >
T gazebo::math::min (const std::vector< T > &_values)
get the minimum value of vector of values
- double **gazebo::math::parseFloat** (const std::string &_input)
parse string into float
- int **gazebo::math::parseInt** (const std::string &_input)
parse string into an integer
- template<typename T >
T gazebo::math::precision (const T &_a, const unsigned int &_precision)

get value at a specified precision

- unsigned int **gazebo::math::roundUpPowerOfTwo** (unsigned int _x)
Get the smallest power of two that is greater or equal to a given value.
- template<typename T >
T **gazebo::math::variance** (const std::vector< T > &_values)
get variance of vector of values

Variables

- static const double **gazebo::math::NAN_D** = std::numeric_limits<double>::quiet_NaN()
Returns the representation of a quiet not a number (NaN)
- static const int **gazebo::math::NAN_I** = std::numeric_limits<int>::quiet_NaN()
Returns the representation of a quiet not a number (NaN)

11.91.1 Macro Definition Documentation

11.91.1.1 **#define GZ_DBL_MAX** std::numeric_limits<double>::max()

Double maximum value.

11.91.1.2 **#define GZ_DBL_MIN** std::numeric_limits<double>::min()

Double min value.

11.91.1.3 **#define GZ_FLT_MAX** std::numeric_limits<float>::max()

Float maximum value.

11.91.1.4 **#define GZ_FLT_MIN** std::numeric_limits<float>::min()

Float minimum value.

11.91.1.5 **#define GZ_INT32_MAX** std::numeric_limits<int32_t>::max()

32bit integer maximum value

11.91.1.6 **#define GZ_INT32_MIN** std::numeric_limits<int32_t>::min()

32bit integer minimum value

11.91.1.7 **#define GZ_UINT32_MAX** std::numeric_limits<uint32_t>::max()

32bit unsigned integer maximum value

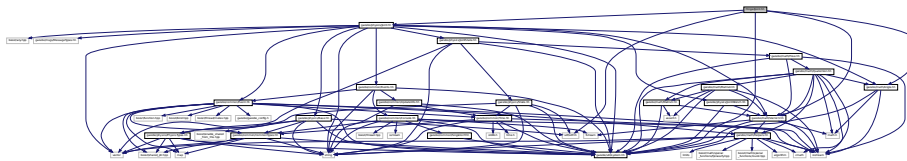
11.91.1.8 `#define GZ_UINT32_MIN std::numeric_limits<uint32_t>::min()`

32bit unsigned integer minimum value

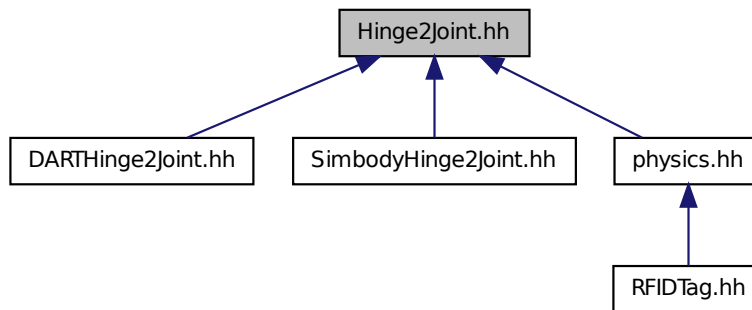
11.92 Hinge2Joint.hh File Reference

```
#include <sdf/sdf.hh>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/Joint.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Hinge2Joint.hh:



This graph shows which files directly or indirectly include this file:



Classes

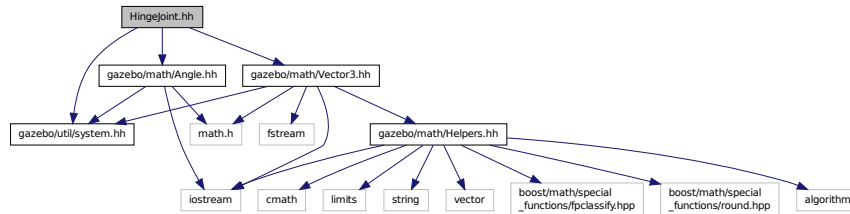
- class **gazebo::physics::Hinge2Joint**< T >
A two axis hinge joint.

Namespaces

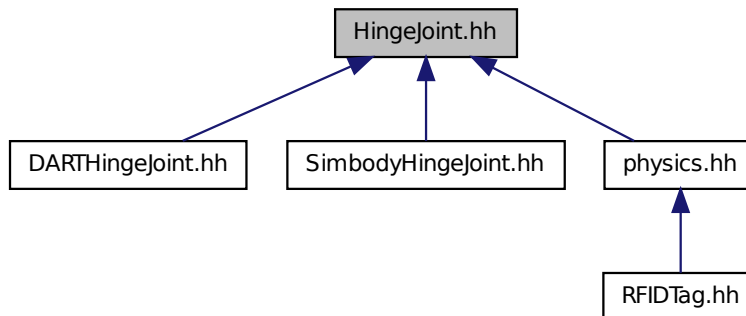
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.93 HingeJoint.hh File Reference

```
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/util/system.hh"
Include dependency graph for HingeJoint.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::HingeJoint**< T >
A single axis hinge joint.

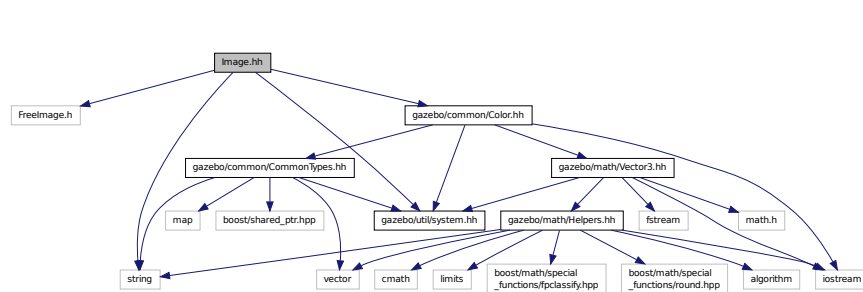
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.94 Image.hh File Reference

```
#include <FreeImage.h>
#include <string>
#include "gazebo/common/Color.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Image.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Image**
Encapsulates an image.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

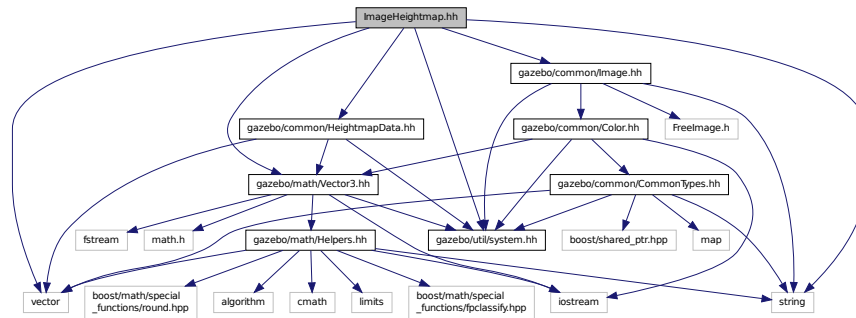
Variables

- static std::string **gazebo::common::PixelFormatNames []**
String names for the pixel formats.

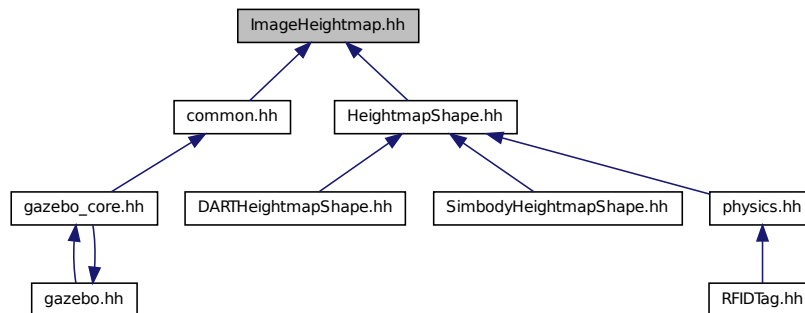
11.95 ImageHeightmap.hh File Reference

```
#include <string>
```

```
#include <vector>
#include "gazebo/common/HeightmapData.hh"
#include "gazebo/common/Image.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/util/system.hh"
Include dependency graph for ImageHeightmap.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::ImageHeightmap**
Encapsulates an image that will be interpreted as a heightmap.

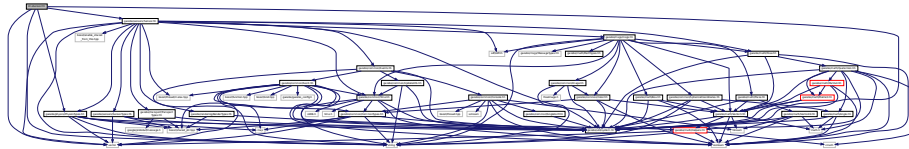
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.96 ImuSensor.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for ImuSensor.hh:



Classes

- class **gazebo::sensors::ImuSensor**
An IMU sensor.

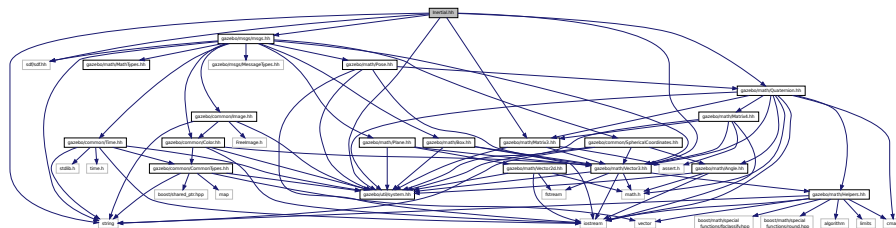
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

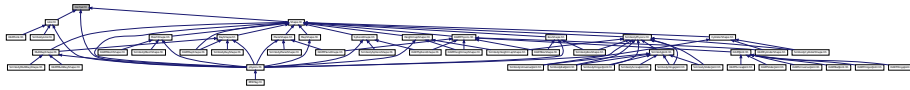
11.97 Inertial.hh File Reference

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/math/Quaternion.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Matrix3.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Inertial.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Inertial**

A class for inertial information about a link.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

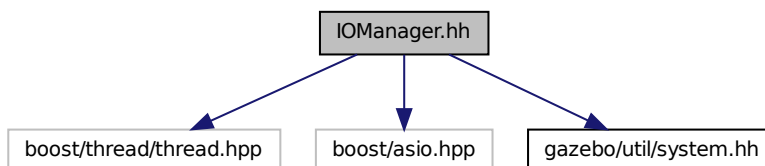
11.98 IOManager.hh File Reference

```
#include <boost/thread/thread.hpp>
```

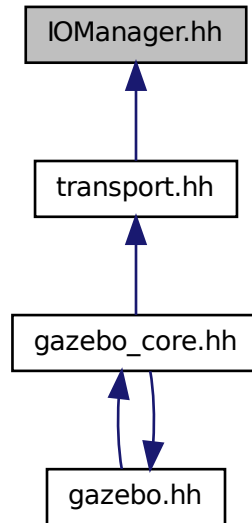
```
#include <boost/asio.hpp>
```

```
#include "gazebo/util/system.hh"
```

Include dependency graph for IOManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::IOManager**

Manages boost::asio IO.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::transport**

11.99 Joint.hh File Reference

```
#include <string>
```

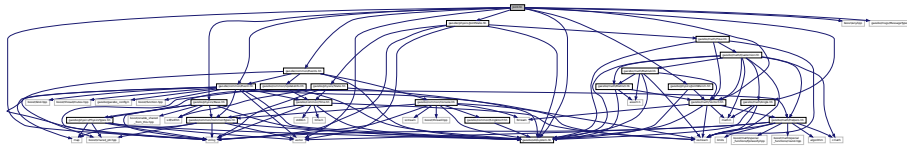


```

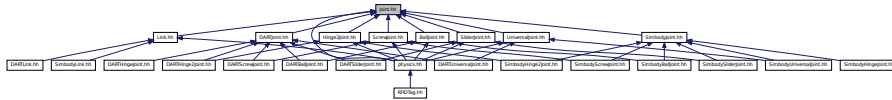
#include <vector>
#include <boost/any.hpp>
#include "gazebo/common/Event.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo msgs/MessageTypes.hh"
#include "gazebo/physics/JointState.hh"
#include "gazebo/physics/Base.hh"
#include "gazebo/physics/JointWrench.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for Joint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Joint**
Base (p. 168) class for all joints.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Macros

- #define **MAX_JOINT_AXIS** 2
maximum number of axis per joint anticipated.

11.99.1 Macro Definition Documentation

11.99.1.1 #define MAX_JOINT_AXIS 2

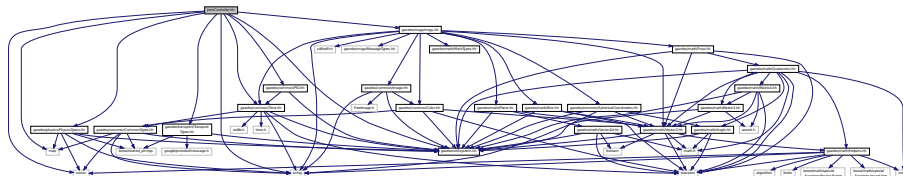
maximum number of axis per joint anticipated.

Currently, this is 2 as 3-axis joints (e.g. ball) actuation, control is not there yet.

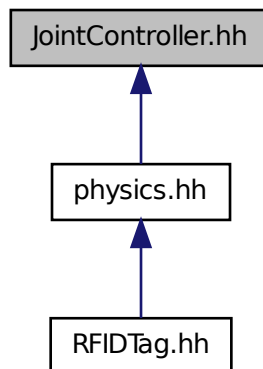
11.100 JointController.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include "gazebo/common/PID.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for JointController.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::JointController**

*A class for manipulating **physics::Joint** (p. 541).*

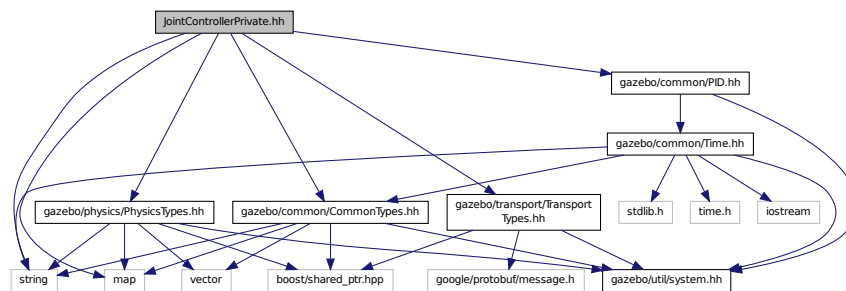
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.101 JointControllerPrivate.hh File Reference

```
#include <string>
#include <map>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/PID.hh"
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for JointControllerPrivate.hh:



Classes

- class **gazebo::physics::JointControllerPrivate**

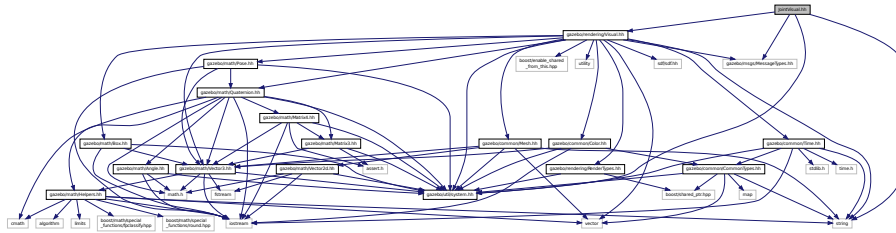
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.102 JointState.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/physics/State.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/util/system.hh"
```


Include dependency graph for JointVisual.hh:



Classes

- class **gazebo::rendering::JointVisual**
Visualization for joints.

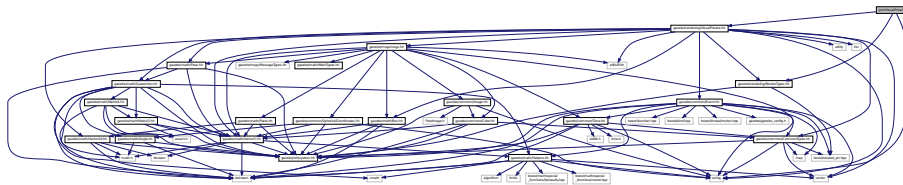
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.104 JointVisualPrivate.hh File Reference

```
#include <string>
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/rendering/VisualPrivate.hh"
```

Include dependency graph for JointVisualPrivate.hh:



Classes

- class **gazebo::rendering::JointVisualPrivate**
*Private data for the Joint **Visual** (p. 1196) class.*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

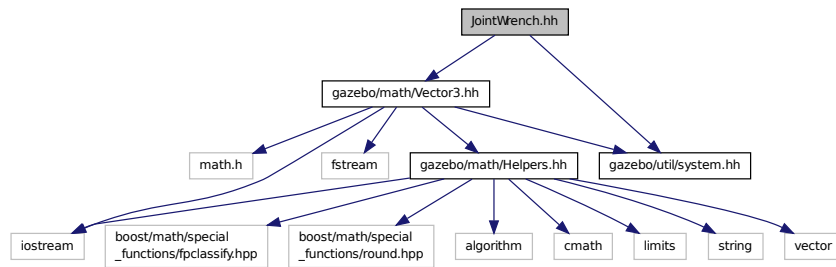
Rendering namespace.

11.105 JointWrench.hh File Reference

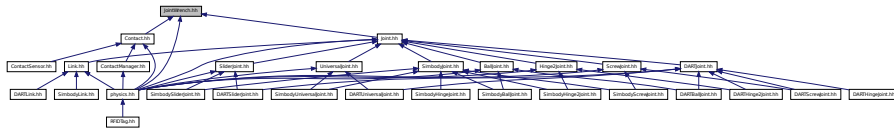
```
#include "gazebo/math/Vector3.hh"
```

```
#include "gazebo/util/system.hh"
```

Include dependency graph for JointWrench.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::JointWrench**

Wrench information from a joint.

Namespaces

- namespace **gazebo**

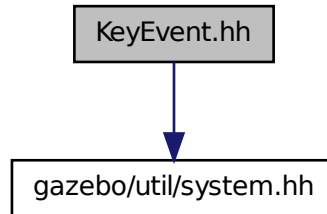
Forward declarations for the common classes.

- namespace **gazebo::physics**

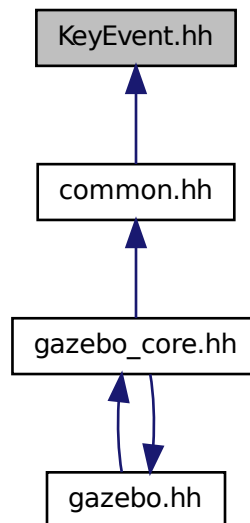
namespace for physics

11.106 KeyEvent.hh File Reference

```
#include "gazebo/util/system.hh"
Include dependency graph for KeyEvent.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::KeyEvent**
Generic description of a keyboard event.

Namespaces

- namespace **gazebo**

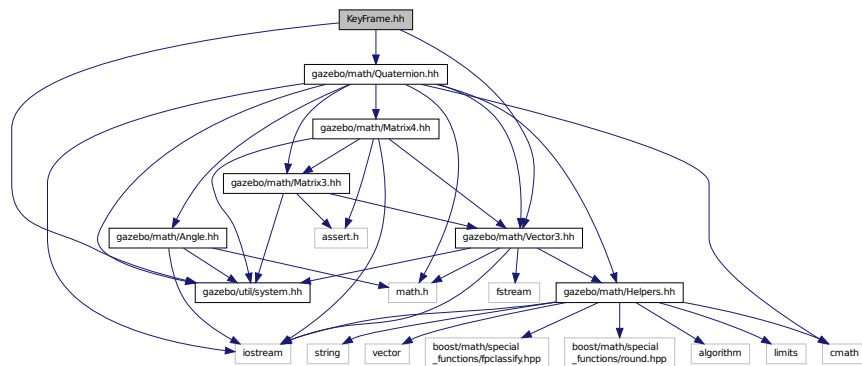
Forward declarations for the common classes.

- namespace **gazebo::common**

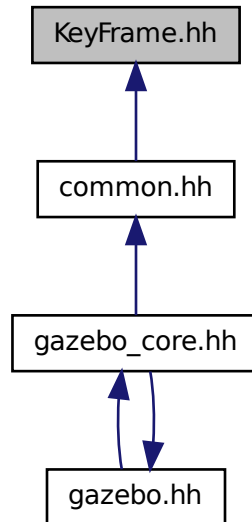
Common namespace.

11.107 KeyFrame.hh File Reference

```
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Quaternion.hh"
#include "gazebo/util/system.hh"
Include dependency graph for KeyFrame.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::KeyFrame**
A key frame in an animation.
- class **gazebo::common::NumericKeyFrame**
*A keyframe for a **NumericAnimation** (p. 752).*
- class **gazebo::common::PoseKeyFrame**
*A keyframe for a **PoseAnimation** (p. 805).*

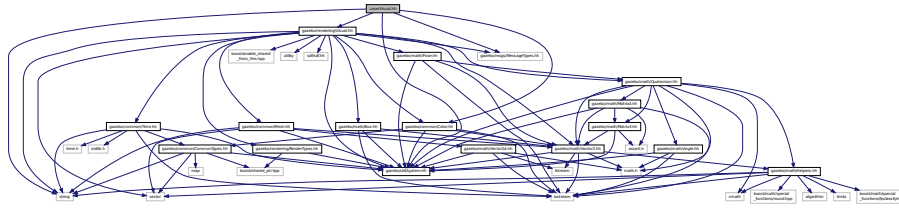
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.108 LaserVisual.hh File Reference

```
#include <string>
#include "gazebo/common/Color.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for LaserVisual.hh:



Classes

- class **gazebo::rendering::LaserVisual**
Visualization for laser data.

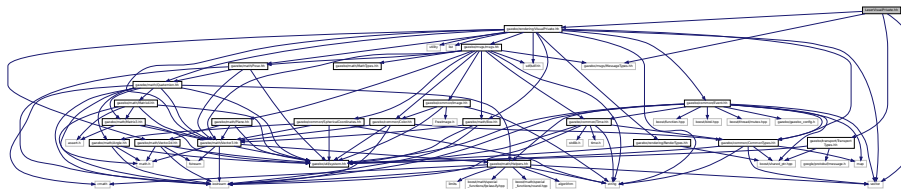
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.109 LaserVisualPrivate.hh File Reference

```
#include <vector>
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/VisualPrivate.hh"
```

Include dependency graph for LaserVisualPrivate.hh:



Classes

- class **gazebo::rendering::LaserVisualPrivate**
*Private data for the Laser **Visual** (p. 1196) class.*

Namespaces

- namespace **gazebo**

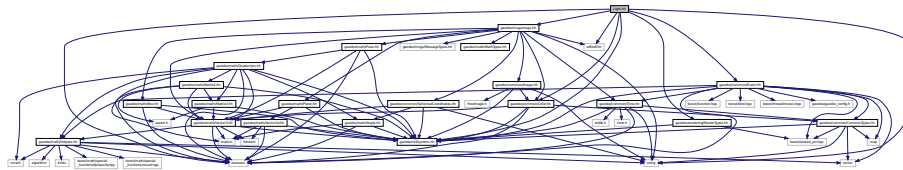
Forward declarations for the common classes.

- namespace **gazebo::rendering**

Rendering namespace.

11.110 Light.hh File Reference

```
#include <string>
#include <iostream>
#include <sdf/sdf.hh>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/common/Color.hh"
#include "gazebo/util/system.hh"
Include dependency graph for Light.hh:
```



Classes

- class **gazebo::rendering::Light**

A light source.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

Rendering namespace.

- namespace **Ogre**

11.111 Link.hh File Reference

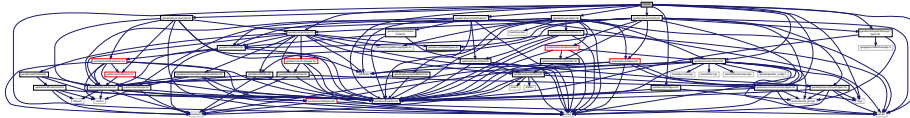
```
#include <map>
```

```

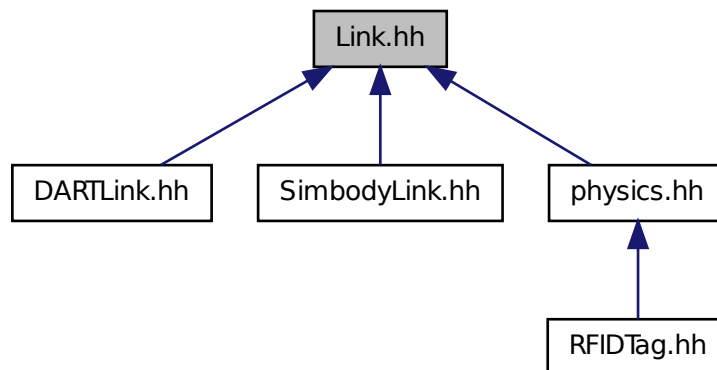
#include <vector>
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/util/UtilTypes.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/LinkState.hh"
#include "gazebo/physics/Entity.hh"
#include "gazebo/physics/Inertial.hh"
#include "gazebo/physics/Joint.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for Link.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Link**

Link (p. 595) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

Namespaces

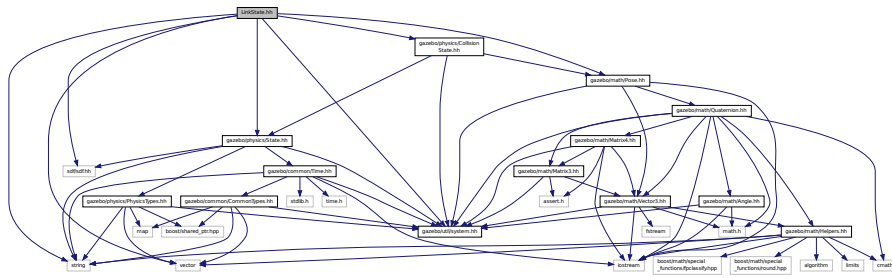
- namespace **gazebo**

Forward declarations for the common classes.

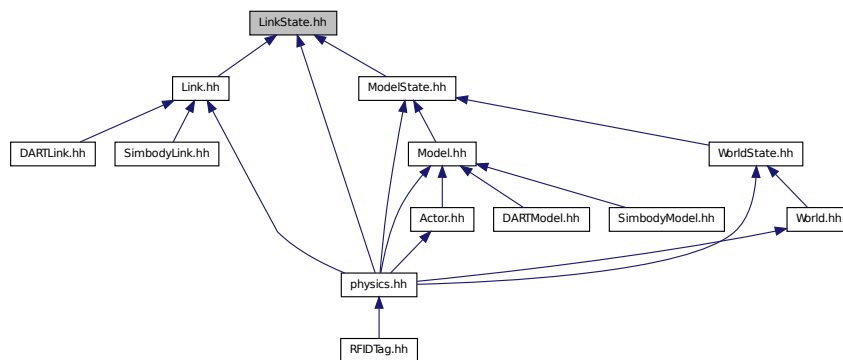
- namespace **gazebo::physics**
namespace for physics
- namespace **gazebo::util**

11.112 LinkState.hh File Reference

```
#include <vector>
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/physics/State.hh"
#include "gazebo/physics/CollisionState.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/util/system.hh"
Include dependency graph for LinkState.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::LinkState**
Store state information of a **physics::Link** (p. 595) object.

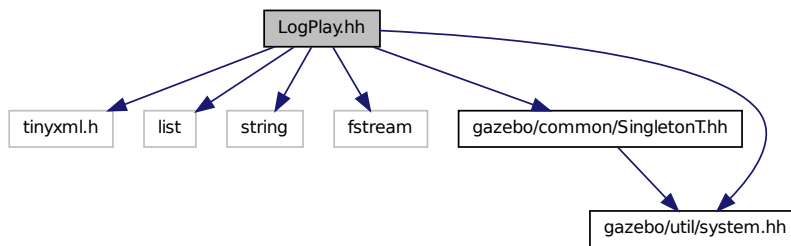
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.113 LogPlay.hh File Reference

```
#include <tinyxml.h>
#include <list>
#include <string>
#include <fstream>
#include "gazebo/common/SingletonT.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for LogPlay.hh:



Classes

- class **gazebo::util::LogPlay**

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::util**

11.114 LogRecord.hh File Reference

```
#include <fstream>
```

```

#include <string>
#include <map>
#include <boost/thread.hpp>
#include <boost/archive/iterators/base64_from_binary.hpp>
#include <boost/archive/iterators/insert_linebreaks.hpp>
#include <boost/archive/iterators/transform_width.hpp>
#include <boost/archive/iterators/ostream_iterator.hpp>
#include <boost/filesystem.hpp>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for LogRecord.hh:



Classes

- class **gazebo::util::LogRecord**
addtogroup gazebo_util

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::util**

Macros

- #define **GZ_LOG_VERSION** "1.0"

11.114.1 Macro Definition Documentation

11.114.1.1 #define GZ_LOG_VERSION "1.0"

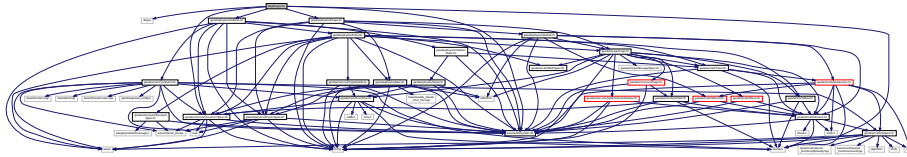
11.115 mainpage.html File Reference

11.116 MapShape.hh File Reference

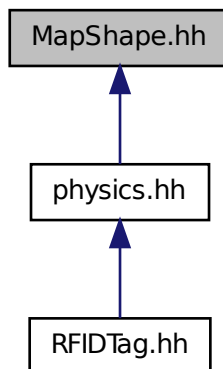
```
#include <deque>
```

```
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for MapShape.hh:



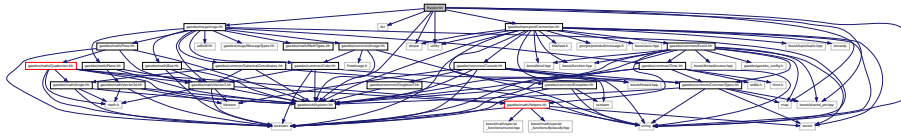
This graph shows which files directly or indirectly include this file:



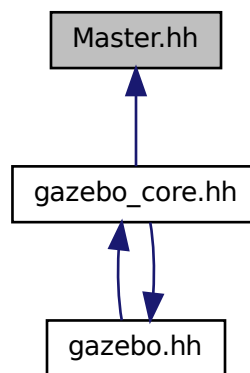
11.117 Master.hh File Reference

```
#include <string>
#include <list>
#include <deque>
#include <utility>
#include <map>
#include <boost/shared_ptr.hpp>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/Connection.hh"
#include "gazebo/util/system.hh"
```


Include dependency graph for Master.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::Master**

A manager that directs topic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

Namespaces

- namespace **gazebo**

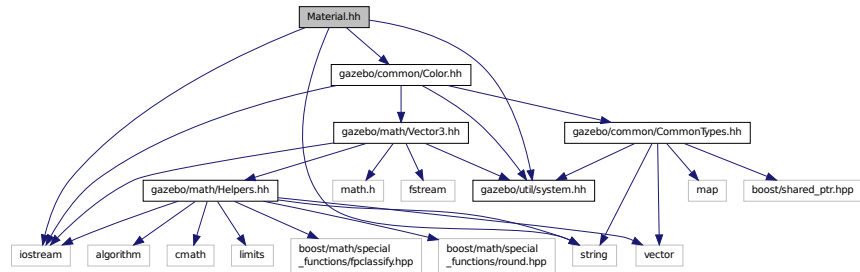
Forward declarations for the common classes.

11.118 Material.hh File Reference

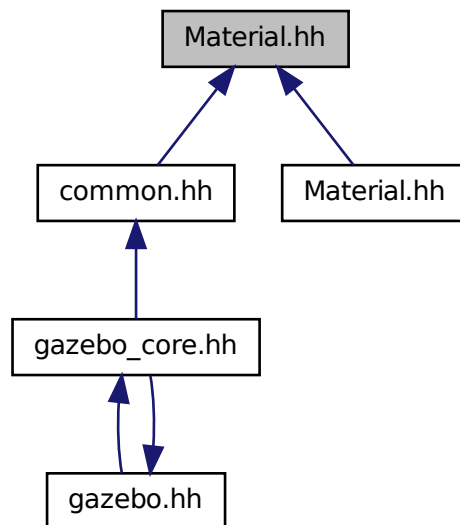
```

#include <string>
#include <iostream>
#include "gazebo/common/Color.hh"
#include "gazebo/util/system.hh"
  
```

Include dependency graph for common/Material.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Material**
Encapsulates description of a material.

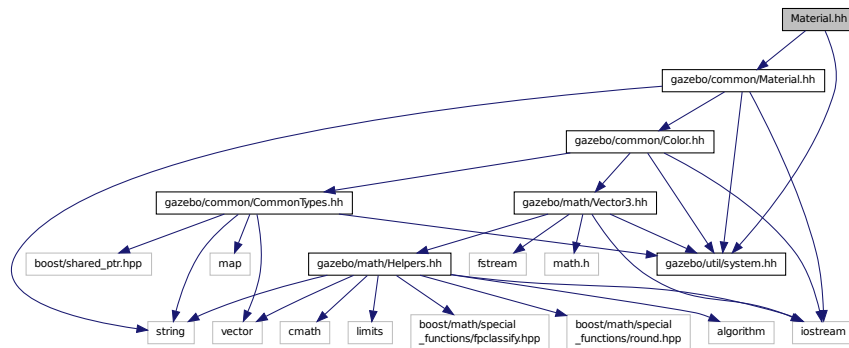
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**

Common namespace.

11.119 Material.hh File Reference

```
#include "gazebo/common/Material.hh"
#include "gazebo/util/system.hh"
Include dependency graph for rendering/Material.hh:
```



11.120 MathTypes.hh File Reference

Forward declarations for the math classes.

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

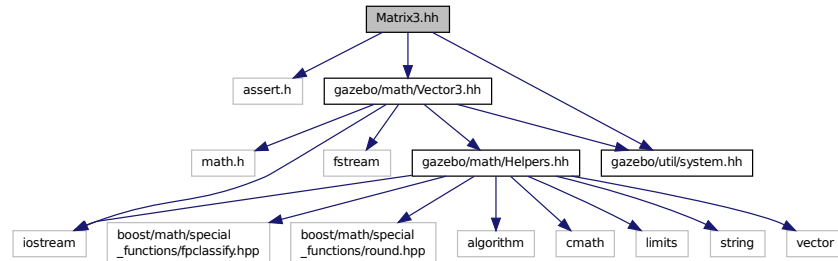
11.120.1 Detailed Description

Forward declarations for the math classes.

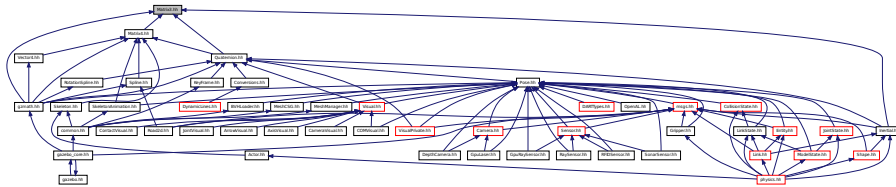
11.121 Matrix3.hh File Reference

```
#include <assert.h>
#include "gazebo/math/Vector3.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Matrix3.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Matrix3**
A 3x3 matrix class.

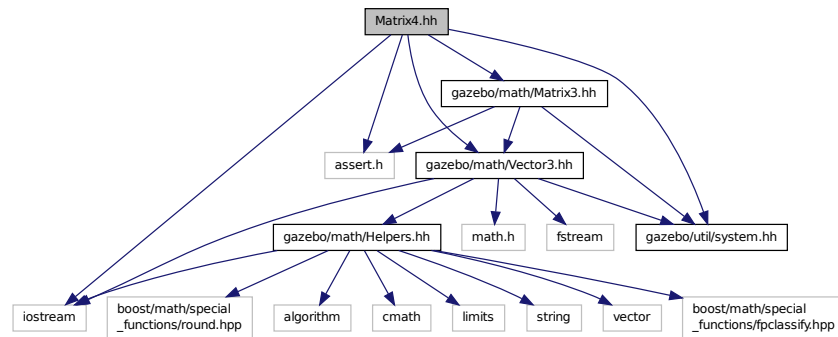
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

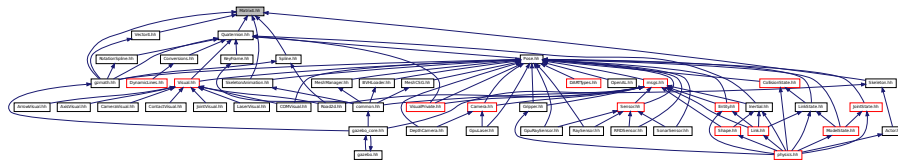
11.122 Matrix4.hh File Reference

```
#include <assert.h>
#include <iostream>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Matrix3.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Matrix4.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Matrix4**

A 3x3 matrix class.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::math**

Math namespace.

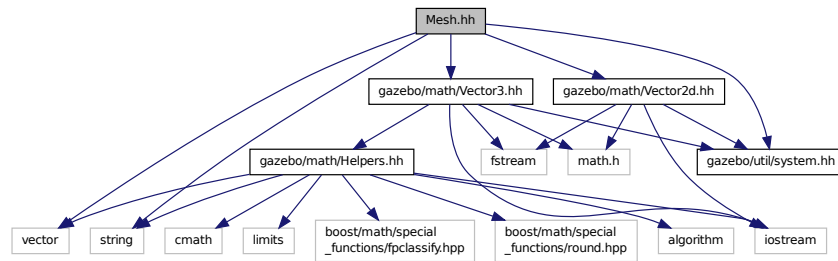
11.123 Mesh.hh File Reference

```

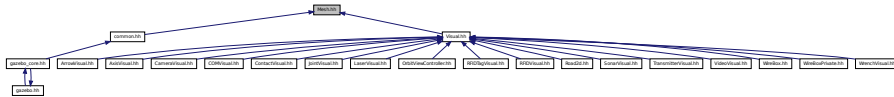
#include <vector>
#include <string>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2d.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for Mesh.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Mesh**
A 3D mesh.
- class **gazebo::common::NodeAssignment**
Vertex to node weighted assignment for skeleton animation visualization.
- class **gazebo::common::SubMesh**
A child mesh.

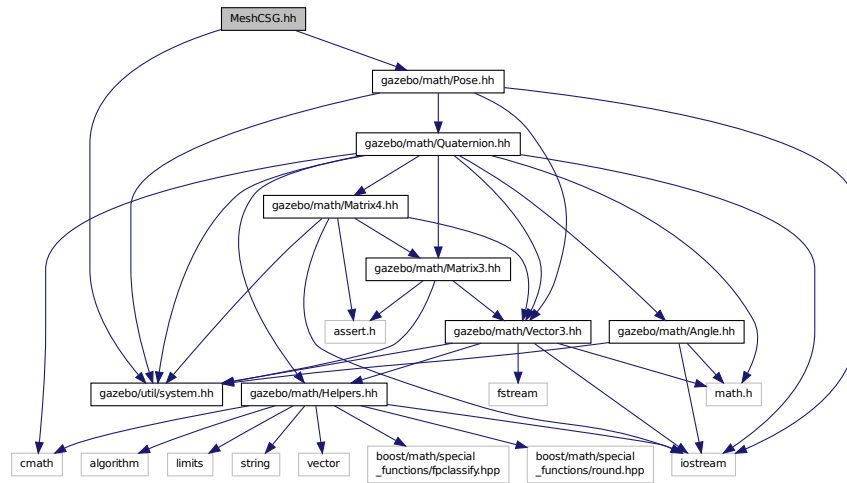
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

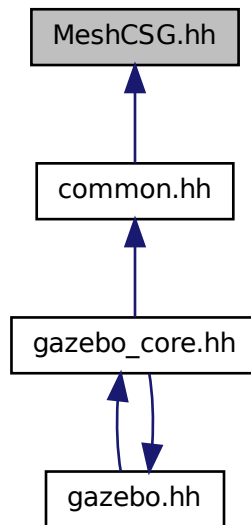
11.124 MeshCSG.hh File Reference

```
#include "gazebo/math/Pose.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for MeshCSG.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::MeshCSG**
Creates CSG meshes.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

Common namespace.

Typedefs

- typedef `_GPtrArray` **GPtrArray**
- typedef `_GtsSurface` **GtsSurface**

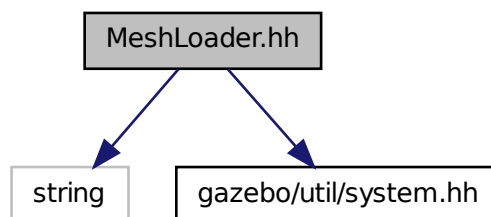
11.124.1 Typedef Documentation

11.124.1.1 typedef `_GPtrArray` **GPtrArray**

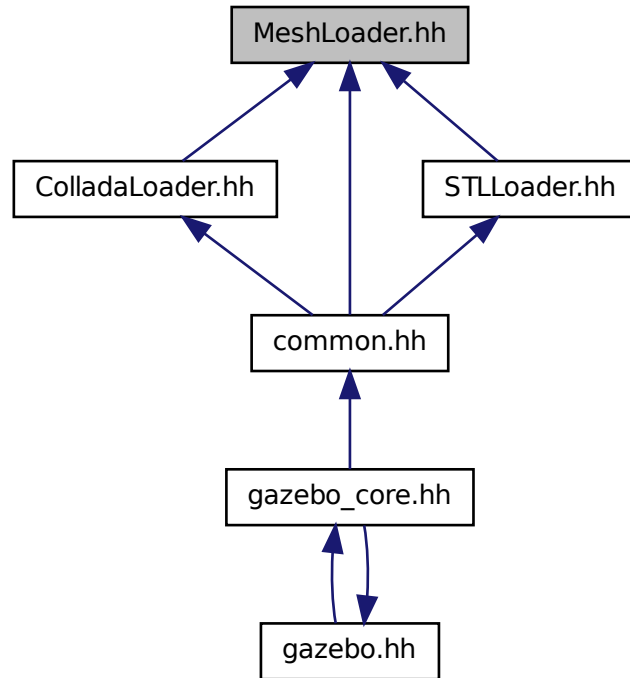
11.124.1.2 typedef `_GtsSurface` **GtsSurface**

11.125 MeshLoader.hh File Reference

```
#include <string>
#include "gazebo/util/system.hh"
Include dependency graph for MeshLoader.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::MeshLoader**

Base class for loading meshes.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.126 MeshManager.hh File Reference

```
#include <map>
```


Classes

- class **gazebo::common::MeshManager**
Maintains and manages all meshes.

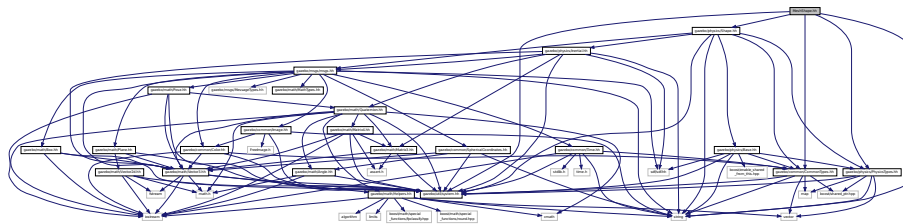
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

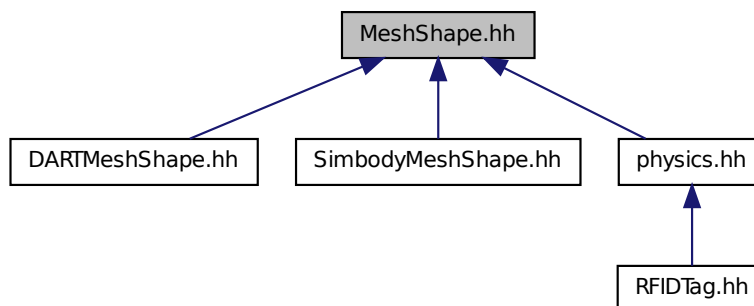
11.127 MeshShape.hh File Reference

```
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for MeshShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::MeshShape**

Triangle mesh collision shape.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

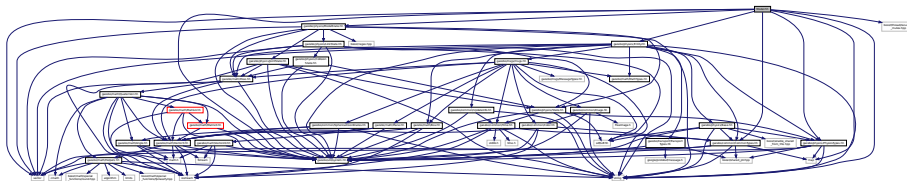
- namespace **gazebo::physics**

namespace for physics

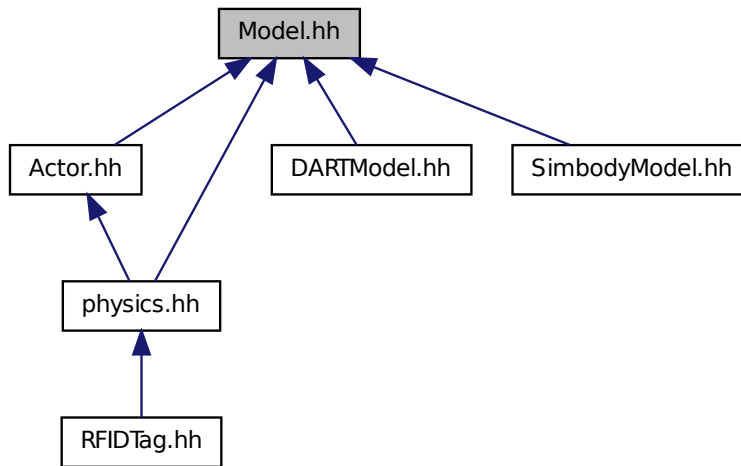
11.128 Model.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include <boost/thread/recursive_mutex.hpp>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/ModelState.hh"
#include "gazebo/physics/Entity.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Model.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Model**
A model is a collection of links, joints, and plugins.

Namespaces

- namespace **boost**
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.129 ModelDatabase.hh File Reference

```

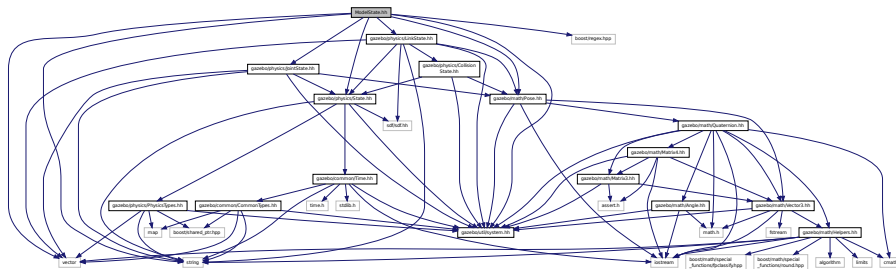
#include <string>
#include <map>
#include <utility>
#include "gazebo/common/Event.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/util/system.hh"

```

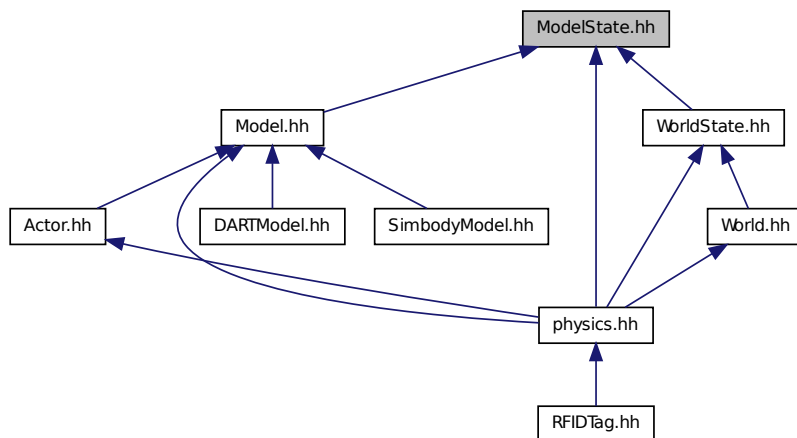

11.131 ModelState.hh File Reference

```
#include <vector>
#include <string>
#include <boost/regex.hpp>
#include "gazebo/math/Pose.hh"
#include "gazebo/physics/State.hh"
#include "gazebo/physics/LinkState.hh"
#include "gazebo/physics/JointState.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for ModelState.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ModelState**
Store state information of a **physics::Model** (p. 678) object.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

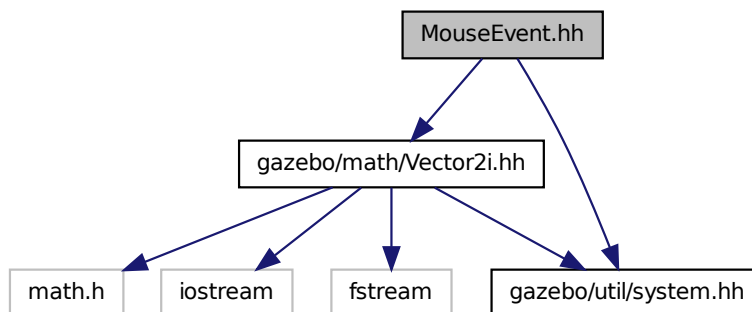
namespace for physics

11.132 MouseEvent.hh File Reference

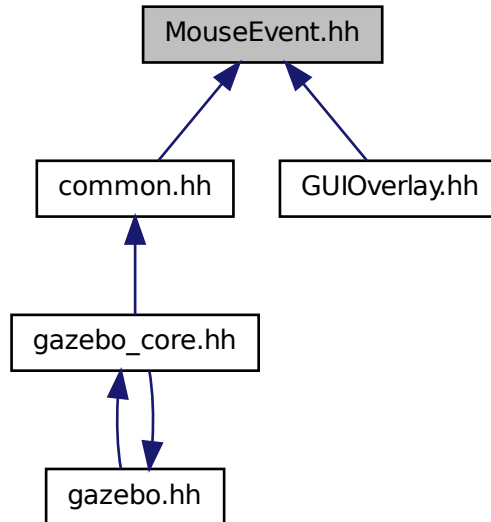
```
#include "gazebo/math/Vector2i.hh"
```

```
#include "gazebo/util/system.hh"
```

Include dependency graph for MouseEvent.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::MouseEvent**
Generic description of a mouse event.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.133 MovableText.hh File Reference

```
#include <string>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/Color.hh"
#include "gazebo/math/MathTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for MovableText.hh:



Classes

- class **gazebo::rendering::MovableText**

Movable text.

Namespaces

- namespace **boost**
- namespace **gazebo**

Forward declarations for the common classes.

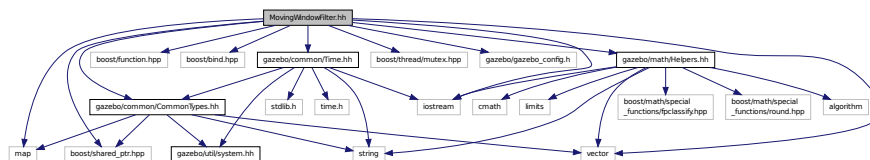
- namespace **gazebo::rendering**

Rendering namespace.

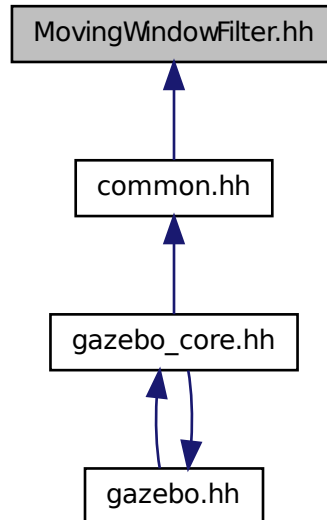
11.134 MovingWindowFilter.hh File Reference

```
#include <iostream>
#include <vector>
#include <map>
#include <boost/function.hpp>
#include <boost/bind.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/thread/mutex.hpp>
#include <gazebo/gazebo_config.h>
#include <gazebo/common/Time.hh>
#include <gazebo/common/CommonTypes.hh>
#include <gazebo/math/Helpers.hh>
```

Include dependency graph for MovingWindowFilter.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::MovingWindowFilter**< T >
*Base class for **MovingWindowFilter** (p. 715).*
- class **gazebo::common::MovingWindowFilterPrivate**< T >

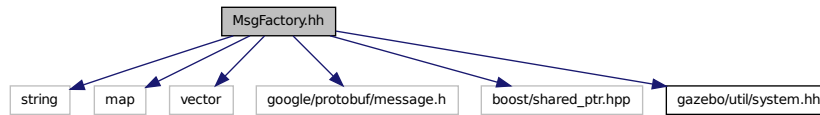
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.135 MsgFactory.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include <google/protobuf/message.h>
#include <boost/shared_ptr.hpp>
#include "gazebo/util/system.hh"
```

Include dependency graph for MsgFactory.hh:



Classes

- class **gazebo::msgs::MsgFactory**
A factory that generates protobuf message based on a string type.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::msgs**
Messages namespace.

Macros

- **#define GZ_REGISTER_STATIC_MSG(_msgtype, _classname)**
Static message registration macro.

Typedefs

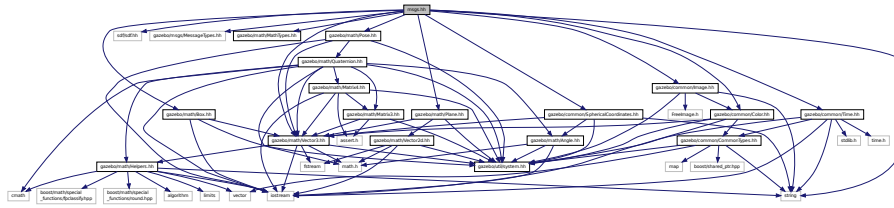
- **typedef boost::shared_ptr**
< google::protobuf::Message >(* gazebo::msgs::MsgFactoryFn)()

11.136 msgs.hh File Reference

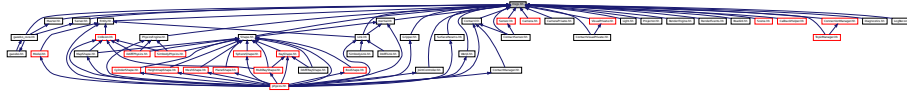
```

#include <string>
#include <sdf/sdf.hh>
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/math/MathTypes.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Plane.hh"
#include "gazebo/math/Box.hh"
#include "gazebo/common/SphericalCoordinates.hh"
#include "gazebo/common/Color.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/common/Image.hh"
  
```

Include dependency graph for msgs.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::msgs**
Messages namespace.

Functions

- **GAZEBO_VISIBLE** msgs::Vector3d **gazebo::msgs::Convert** (const math::Vector3 &_v)
*Convert a **math::Vector3** (p. 1165) to a msgs::Vector3d.*
- **GAZEBO_VISIBLE** msgs::Quaternion **gazebo::msgs::Convert** (const math::Quaternion &_q)
*Convert a **math::Quaternion** (p. 824) to a msgs::Quaternion.*
- **GAZEBO_VISIBLE** msgs::Pose **gazebo::msgs::Convert** (const math::Pose &_p)
*Convert a **math::Pose** (p. 797) to a msgs::Pose.*
- **GAZEBO_VISIBLE** msgs::Color **gazebo::msgs::Convert** (const common::Color &_c)
*Convert a **common::Color** (p. 249) to a msgs::Color.*
- **GAZEBO_VISIBLE** msgs::Time **gazebo::msgs::Convert** (const common::Time &_t)
*Convert a **common::Time** (p. 1099) to a msgs::Time.*
- **GAZEBO_VISIBLE** msgs::PlaneGeom **gazebo::msgs::Convert** (const math::Plane &_p)
*Convert a **math::Plane** (p. 788) to a msgs::PlaneGeom.*
- **GAZEBO_VISIBLE** math::Vector3 **gazebo::msgs::Convert** (const msgs::Vector3d &_v)
Convert a msgs::Vector3d to a math::Vector.
- **GAZEBO_VISIBLE** math::Quaternion **gazebo::msgs::Convert** (const msgs::Quaternion &_q)
*Convert a msgs::Quaternion to a **math::Quaternion** (p. 824).*
- **GAZEBO_VISIBLE** math::Pose **gazebo::msgs::Convert** (const msgs::Pose &_p)
*Convert a msgs::Pose to a **math::Pose** (p. 797).*
- **GAZEBO_VISIBLE** common::Color **gazebo::msgs::Convert** (const msgs::Color &_c)
*Convert a msgs::Color to a **common::Color** (p. 249).*

- **GAZEBO_VISIBLE** common::Time **gazebo::msgs::Convert** (const msgs::Time &_t)
*Convert a msgs::Time to a **common::Time** (p. 1099).*
- **GAZEBO_VISIBLE** msgs::Plane **gazebo::msgs::Convert** (const msgs::PlaneGeom &_p)
*Convert a msgs::PlaneGeom to a **common::Plane**.*
- **GAZEBO_VISIBLE** msgs::Request * **gazebo::msgs::CreateRequest** (const std::string &_request, const std::string &_data="")
Create a request message.
- **GAZEBO_VISIBLE** msgs::Fog **gazebo::msgs::FogFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::Fog from a fog SDF element.
- **GAZEBO_VISIBLE** msgs::Geometry **gazebo::msgs::GeometryFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::Geometry from a geometry SDF element.
- **GAZEBO_VISIBLE** msgs::Header * **gazebo::msgs::GetHeader** (google::protobuf::Message &_message)
Get the header from a protobuf message.
- **GAZEBO_VISIBLE** msgs::GUI **gazebo::msgs::GUIFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::GUI from a GUI SDF element.
- **GAZEBO_VISIBLE** void **gazebo::msgs::Init** (google::protobuf::Message &_message, const std::string &_id="")
Initialize a message.
- **GAZEBO_VISIBLE** msgs::Light **gazebo::msgs::LightFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::Light from a light SDF element.
- **GAZEBO_VISIBLE** msgs::MeshGeom **gazebo::msgs::MeshFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::MeshGeom from a mesh SDF element.
- **GAZEBO_VISIBLE** msgs::Scene **gazebo::msgs::SceneFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::Scene from a scene SDF element.
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (common::Image &_img, const msgs::Image &_msg)
*Convert a msgs::Image to a **common::Image** (p. 515).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Image *_msg, const common::Image &_i)
*Set a msgs::Image from a **common::Image** (p. 515).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Vector3d *_pt, const math::Vector3 &_v)
*Set a msgs::Vector3d from a **math::Vector3** (p. 1165).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Vector2d *_pt, const math::Vector2d &_v)
*Set a msgs::Vector2d from a **math::Vector3** (p. 1165).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Quaternion *_q, const math::Quaternion &_v)
*Set a msgs::Quaternion from a **math::Quaternion** (p. 824).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Pose *_p, const math::Pose &_v)
*Set a msgs::Pose from a **math::Pose** (p. 797).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Color *_c, const common::Color &_v)
*Set a msgs::Color from a **common::Color** (p. 249).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Time *_t, const common::Time &_v)
*Set a msgs::Time from a **common::Time** (p. 1099).*
- void **gazebo::msgs::Set** (msgs::SphericalCoordinates *_s, const common::SphericalCoordinates &_v)
*Set a msgs::SphericalCoordinates from a **common::SphericalCoordinates** (p. 1057) object.*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::PlaneGeom *_p, const math::Plane &_v)
*Set a msgs::Plane from a **math::Plane** (p. 788).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Stamp** (msgs::Header *_header)
Time stamp a header.
- **GAZEBO_VISIBLE** void **gazebo::msgs::Stamp** (msgs::Time *_time)

Set the time in a time message.

- **GAZEBO_VISIBLE** msgs::TrackVisual **gazebo::msgs::TrackVisualFromSDF** (sdf::ElementPtr _sdf)

Create a msgs::TrackVisual from a track visual SDF element.

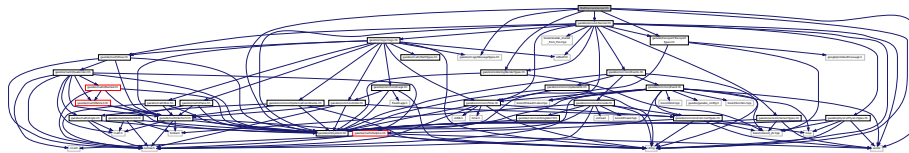
- **GAZEBO_VISIBLE** msgs::Visual **gazebo::msgs::VisualFromSDF** (sdf::ElementPtr _sdf)

Create a msgs::Visual from a visual SDF element.

11.137 MultiCameraSensor.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for MultiCameraSensor.hh:



Classes

- class **gazebo::sensors::MultiCameraSensor**

Multiple camera sensor.

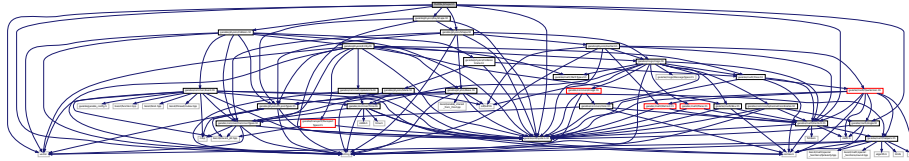
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

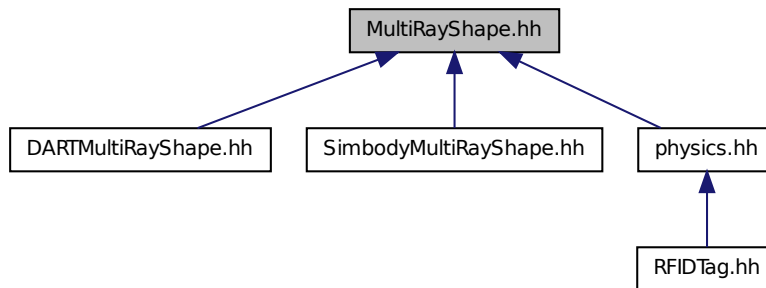
11.138 MultiRayShape.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/physics/RayShape.hh"
#include "gazebo/util/system.hh"
```


Include dependency graph for MultiRayShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::MultiRayShape**

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

Namespaces

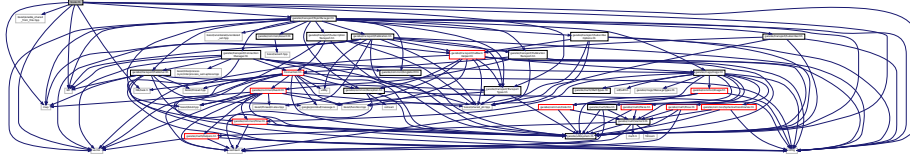
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.139 Node.hh File Reference

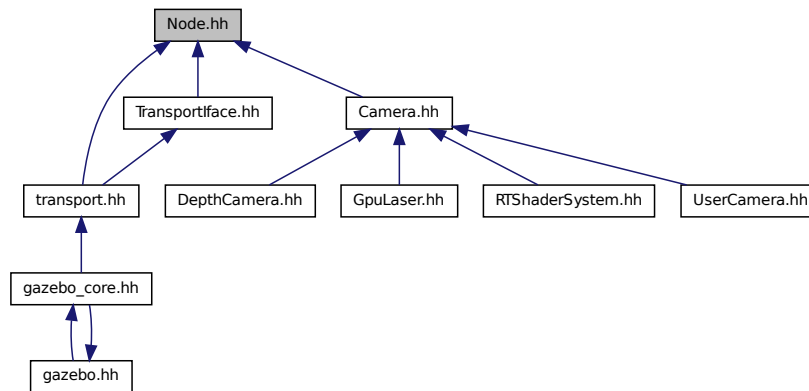
```

#include <tbb/task.h>
#include <boost/enable_shared_from_this.hpp>
#include <map>
#include <list>
#include <string>
#include <vector>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/transport/TopicManager.hh"
#include "gazebo/util/system.hh"
  
```

Include dependency graph for Node.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Node**
A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.
- class **gazebo::transport::PublishTask**

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

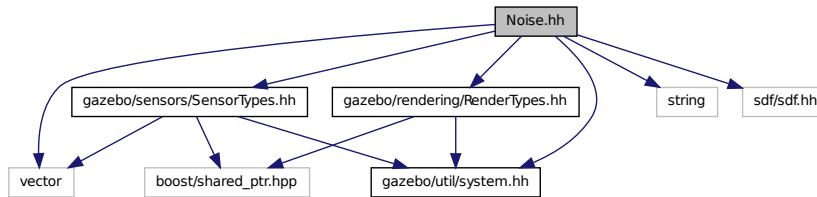
11.140 Noise.hh File Reference

```

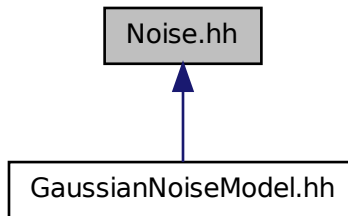
#include <vector>
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/sensors/SensorTypes.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for Noise.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::sensors::Noise**
Noise (p. 748) models for sensor output signals.
- class **gazebo::sensors::NoiseFactory**
Use this noise manager for creating and loading noise models.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.141 ogre_gazebo.h File Reference

```
#include <OGRE/Ogre.h>
```

```

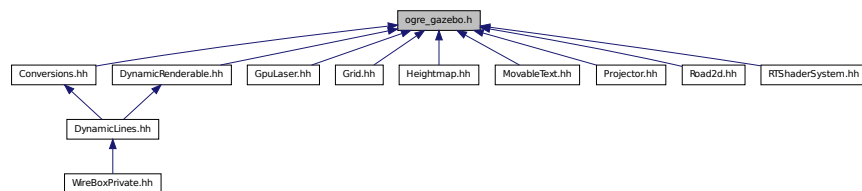
#include <OGRE/OgreImageCodec.h>
#include <OGRE/OgreMovableObject.h>
#include <OGRE/OgreRenderable.h>
#include <OGRE/OgrePlugin.h>
#include <OGRE/OgreDataStream.h>
#include <OGRE/OgreLogManager.h>
#include <OGRE/OgreWindowEventUtilities.h>
#include <OGRE/OgreSceneQuery.h>
#include <OGRE/OgreRoot.h>
#include <OGRE/OgreSceneManager.h>
#include <OGRE/OgreSceneNode.h>
#include <OGRE/OgreVector3.h>
#include <OGRE/OgreManualObject.h>
#include <OGRE/OgreMaterialManager.h>
#include <OGRE/OgreColourValue.h>
#include <OGRE/OgreQuaternion.h>
#include <OGRE/OgreMesh.h>
#include <OGRE/OgreHardwareBufferManager.h>
#include <OGRE/OgreCamera.h>
#include <OGRE/OgreNode.h>
#include <OGRE/OgreSimpleRenderable.h>
#include <OGRE/OgreFrameListener.h>
#include <OGRE/OgreTexture.h>
#include <OGRE/OgreRenderObjectListener.h>
#include <OGRE/OgreTechnique.h>
#include <OGRE/OgrePass.h>
#include <OGRE/OgreTextureUnitState.h>
#include <OGRE/OgreGpuProgramManager.h>
#include <OGRE/OgreHighLevelGpuProgramManager.h>
#include <OGRE/OgreHardwarePixelBuffer.h>
#include <OGRE/OgreShadowCameraSetupPSSM.h>
#include <OGRE/Paging/OgrePageManager.h>
#include <OGRE/Paging/OgrePagedWorld.h>
#include <OGRE/Terrain/OgreTerrainPaging.h>
#include <OGRE/Terrain/OgreTerrainMaterialGeneratorA.h>
#include <OGRE/Terrain/OgreTerrain.h>
#include <OGRE/Terrain/OgreTerrainGroup.h>
#include <OGRE/OgreFontManager.h>

```

Include dependency graph for `ogre_gazebo.h`:



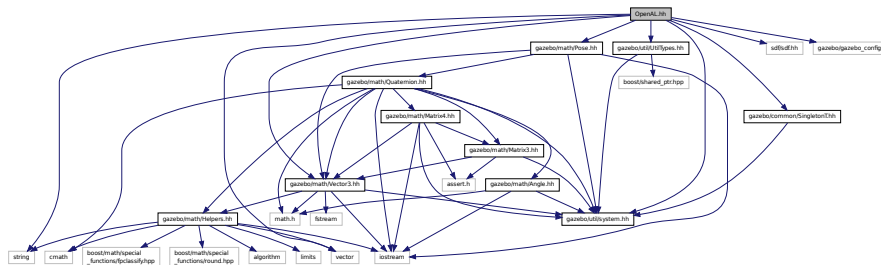
This graph shows which files directly or indirectly include this file:



11.142 OpenAL.hh File Reference

```
#include <string>
#include <vector>
#include <sdf/sdf.hh>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/util/UtilTypes.hh"
#include "gazebo/gazebo_config.h"
#include "gazebo/util/system.hh"
```

Include dependency graph for OpenAL.hh:



Classes

- class **gazebo::util::OpenAL**
3D audio setup and playback.
- class **gazebo::util::OpenALSink**
OpenAL (p. 755) Listener.
- class **gazebo::util::OpenALSource**
OpenAL (p. 755) Source.

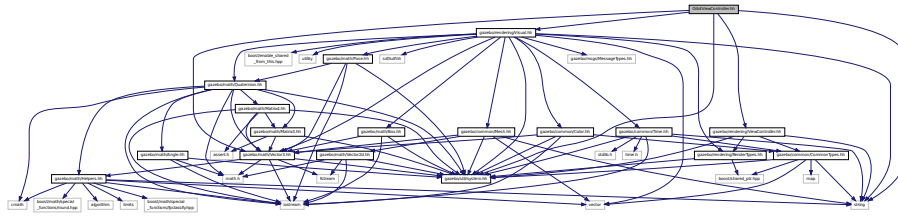
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::util**

11.143 OrbitViewController.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/rendering/ViewController.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for OrbitViewController.hh:



Classes

- class **gazebo::rendering::OrbitViewController**
Orbit view controller.

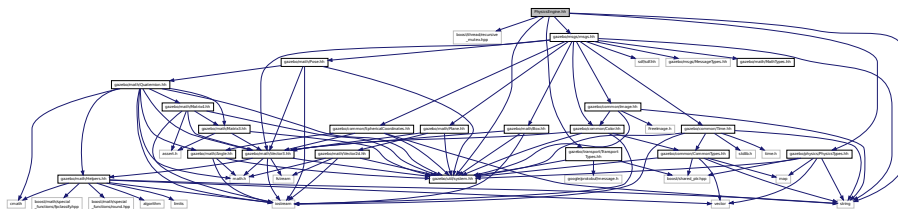
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

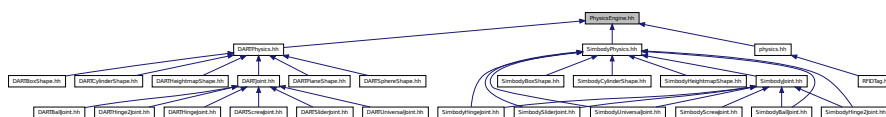
11.144 PhysicsEngine.hh File Reference

```
#include <boost/thread/recursive_mutex.hpp>
#include <string>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo_msgs/msgs.hh"
#include "gazebo_physics/PhysicsTypes.hh"
#include "gazebo_util/system.hh"
```

Include dependency graph for PhysicsEngine.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::PhysicsEngine**

Base (p. 168) class for a physics engine.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

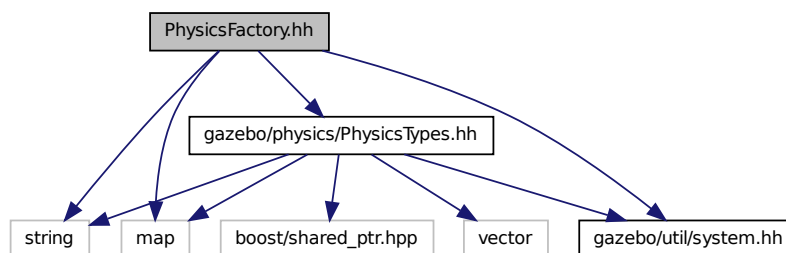
- namespace **gazebo::physics**

namespace for physics

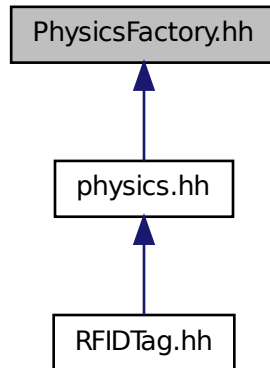
11.145 PhysicsFactory.hh File Reference

```
#include <string>
#include <map>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for PhysicsFactory.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::PhysicsFactory**
The physics factory instantiates different physics engines.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Macros

- #define **GZ_REGISTER_PHYSICS_ENGINE**(name, classname)
Static physics registration macro.

Typedefs

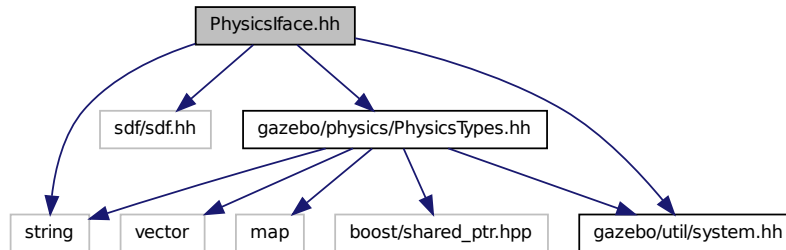
- typedef PhysicsEnginePtr(* **gazebo::physics::PhysicsFactoryFn**)(WorldPtr world)

11.146 PhysicsIface.hh File Reference

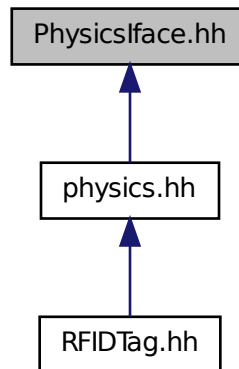
```
#include <string>
```



```
#include <sdf/sdf.hh>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/util/system.hh"
Include dependency graph for PhysicsIface.hh:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Functions

- **GAZEBO_VISIBLE** WorldPtr **gazebo::physics::create_world** (const std::string &_name="")

Create a world given a name.

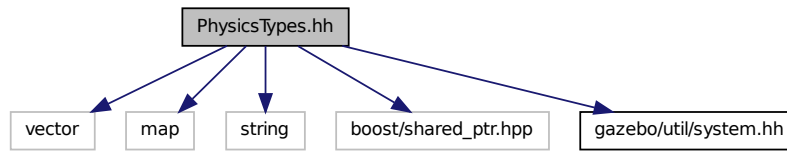
- **GAZEBO_VISIBLE** bool `gazebo::physics::fini ()`
Finalize transport by calling `gazebo::transport::fini` (p. 94).
- **GAZEBO_VISIBLE** WorldPtr `gazebo::physics::get_world (const std::string &_name="")`
Returns a pointer to a world by name.
- **GAZEBO_VISIBLE** uint32_t `gazebo::physics::getUniqueld ()`
Get a unique ID.
- **GAZEBO_VISIBLE** void `gazebo::physics::init_world (WorldPtr _world)`
Init world given a pointer to it.
- **GAZEBO_VISIBLE** void `gazebo::physics::init_worlds ()`
initialize multiple worlds stored in static variable `gazebo::g_worlds`
- **GAZEBO_VISIBLE** bool `gazebo::physics::load ()`
Setup `gazebo::SystemPlugin` (p. 1098)'s and call `gazebo::transport::init` (p. 96).
- **GAZEBO_VISIBLE** void `gazebo::physics::load_world (WorldPtr _world, sdf::ElementPtr _sdf)`
Load world from `sdf::Element` pointer.
- **GAZEBO_VISIBLE** void `gazebo::physics::load_worlds (sdf::ElementPtr _sdf)`
load multiple worlds from single `sdf::Element` pointer
- **GAZEBO_VISIBLE** void `gazebo::physics::pause_world (WorldPtr _world, bool _pause)`
Pause world by calling `World::SetPaused` (p. 1250).
- **GAZEBO_VISIBLE** void `gazebo::physics::pause_worlds (bool pause)`
pause multiple worlds stored in static variable `gazebo::g_worlds`
- **GAZEBO_VISIBLE** void `gazebo::physics::remove_worlds ()`
remove multiple worlds stored in static variable `gazebo::g_worlds`
- **GAZEBO_VISIBLE** void `gazebo::physics::run_world (WorldPtr _world, unsigned int _iterations=0)`
Run world by calling `World::Run()` (p. 1249) given a pointer to it.
- **GAZEBO_VISIBLE** void `gazebo::physics::run_worlds (unsigned int _iterations=0)`
Run multiple worlds stored in static variable `gazebo::g_worlds`.
- **GAZEBO_VISIBLE** void `gazebo::physics::stop_world (WorldPtr _world)`
Stop world by calling `World::Stop()` (p. 1251) given a pointer to it.
- **GAZEBO_VISIBLE** void `gazebo::physics::stop_worlds ()`
stop multiple worlds stored in static variable `gazebo::g_worlds`
- **GAZEBO_VISIBLE** bool `gazebo::physics::worlds_running ()`
Return true if any world is running.

11.147 PhysicsTypes.hh File Reference

default namespace for gazebo

```
#include <vector>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
#include "gazebo/util/system.hh"
```

Include dependency graph for PhysicsTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Macros

- **#define GZ_ALL_COLLIDE 0x0FFFFFFF**
Default collision bitmask.
- **#define GZ_FIXED_COLLIDE 0x00000001**
Collision object will collide only with fixed objects.
- **#define GZ_GHOST_COLLIDE 0x10000000**
Collides with everything else but other ghost.
- **#define GZ_NONE_COLLIDE 0x00000000**
Collision object will collide with nothing.
- **#define GZ_SENSOR_COLLIDE 0x00000002**
Collision object will collide only with sensors.

Typedefs

- typedef std::vector< ActorPtr > **gazebo::physics::Actor_V**
- typedef boost::shared_ptr< Actor > **gazebo::physics::ActorPtr**
- typedef std::vector< BasePtr > **gazebo::physics::Base_V**
- typedef boost::shared_ptr< Base > **gazebo::physics::BasePtr**
- typedef boost::shared_ptr
< BoxShape > **gazebo::physics::BoxShapePtr**

- typedef std::vector< CollisionPtr > **gazebo::physics::Collision_V**
- typedef boost::shared_ptr< Collision > **gazebo::physics::CollisionPtr**
- typedef boost::shared_ptr< Contact > **gazebo::physics::ContactPtr**
- typedef boost::shared_ptr< CylinderShape > **gazebo::physics::CylinderShapePtr**
- typedef boost::shared_ptr< Entity > **gazebo::physics::EntityPtr**
- typedef boost::shared_ptr< Gripper > **gazebo::physics::GripperPtr**
- typedef boost::shared_ptr< HeightmapShape > **gazebo::physics::HeightmapShapePtr**
- typedef boost::shared_ptr< Inertial > **gazebo::physics::InertialPtr**
- typedef std::vector< JointPtr > **gazebo::physics::Joint_V**
- typedef std::vector< JointControllerPtr > **gazebo::physics::JointController_V**
- typedef boost::shared_ptr< JointController > **gazebo::physics::JointControllerPtr**
- typedef boost::shared_ptr< Joint > **gazebo::physics::JointPtr**
- typedef std::map< std::string, JointState > **gazebo::physics::JointState_M**
- typedef std::vector< LinkPtr > **gazebo::physics::Link_V**
- typedef boost::shared_ptr< Link > **gazebo::physics::LinkPtr**
- typedef std::map< std::string, LinkState > **gazebo::physics::LinkState_M**
- typedef boost::shared_ptr< MeshShape > **gazebo::physics::MeshShapePtr**
- typedef std::vector< ModelPtr > **gazebo::physics::Model_V**
- typedef boost::shared_ptr< Model > **gazebo::physics::ModelPtr**
- typedef std::map< std::string, ModelState > **gazebo::physics::ModelState_M**
- typedef boost::shared_ptr< MultiRayShape > **gazebo::physics::MultiRayShapePtr**
- typedef boost::shared_ptr< PhysicsEngine > **gazebo::physics::PhysicsEnginePtr**
- typedef boost::shared_ptr< RayShape > **gazebo::physics::RayShapePtr**
- typedef boost::shared_ptr< Road > **gazebo::physics::RoadPtr**
- typedef boost::shared_ptr< Shape > **gazebo::physics::ShapePtr**
- typedef boost::shared_ptr< SphereShape > **gazebo::physics::SphereShapePtr**
- typedef boost::shared_ptr< SurfaceParams > **gazebo::physics::SurfaceParamsPtr**
- typedef boost::shared_ptr< World > **gazebo::physics::WorldPtr**

11.147.1 Detailed Description

default namespace for gazebo

11.147.2 Macro Definition Documentation

11.147.2.1 #define GZ_ALL_COLLIDE 0x0FFFFFFF

Default collision bitmask.

Collision objects will collide with everything.

11.147.2.2 #define GZ_FIXED_COLLIDE 0x00000001

Collision object will collide only with fixed objects.

11.147.2.3 #define GZ_GHOST_COLLIDE 0x10000000

Collides with everything else but other ghost.

11.147.2.4 #define GZ_NONE_COLLIDE 0x00000000

Collision object will collide with nothing.

11.147.2.5 #define GZ_SENSOR_COLLIDE 0x00000002

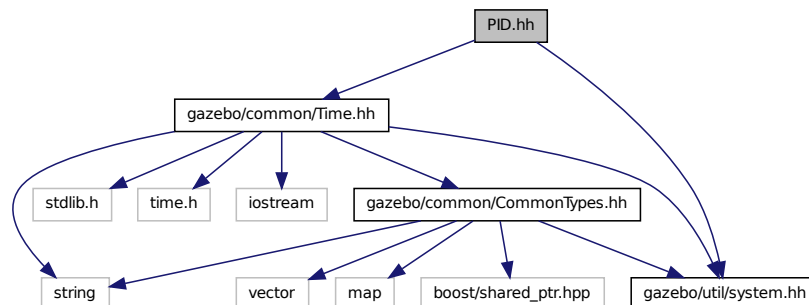
Collision object will collide only with sensors.

11.148 PID.hh File Reference

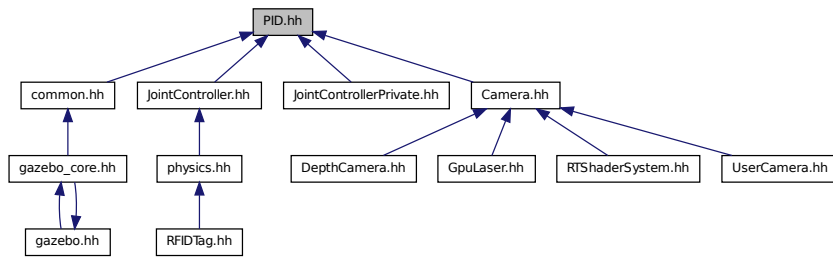
```
#include "gazebo/common/Time.hh"
```

```
#include "gazebo/util/system.hh"
```

Include dependency graph for PID.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::PID**

*Generic **PID** (p. 782) controller class.*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

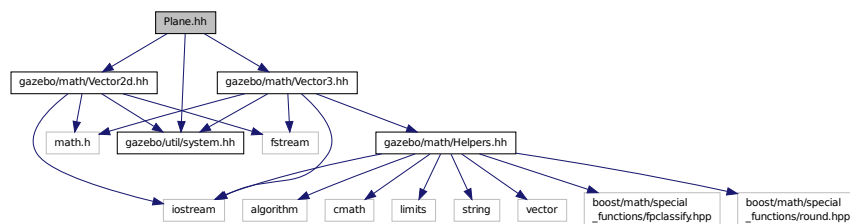
- namespace **gazebo::common**

Common namespace.

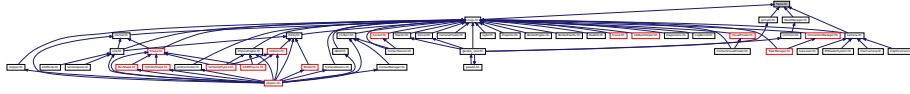
11.149 Plane.hh File Reference

```
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2d.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Plane.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Plane**

A plane and related functions.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

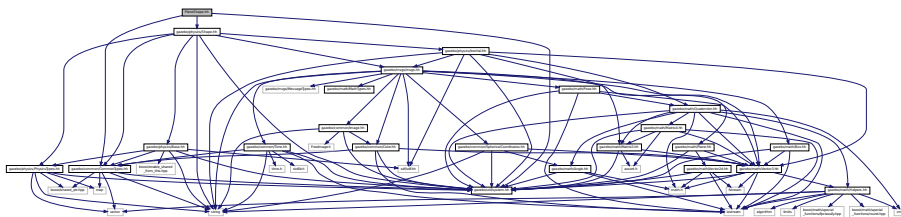
- namespace **gazebo::math**

Math namespace.

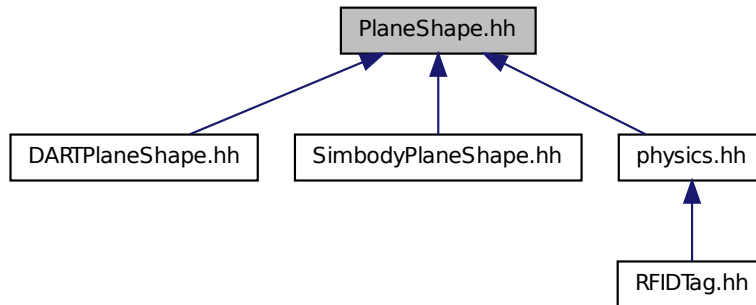
11.150 PlaneShape.hh File Reference

```
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for PlaneShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::PlaneShape**
Collision (p. 235) for an infinite plane.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

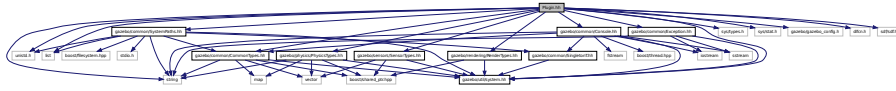
11.151 Plugin.hh File Reference

```

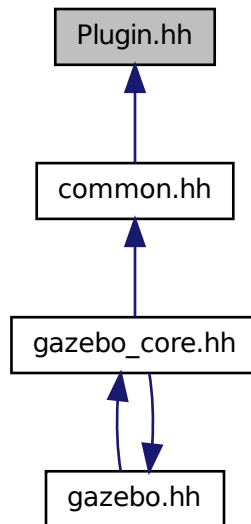
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <gazebo/gazebo_config.h>
#include <dlfcn.h>
#include <list>
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/SystemPaths.hh"
#include "gazebo/common/Console.hh"
#include "gazebo/common/Exception.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/SensorTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"

```


Include dependency graph for Plugin.hh:



This graph shows which files directly or indirectly include this file:



Classes

- union **gazebo::PluginT**< T >::fptr_union_t
Pointer to shared library registration function definition.
- class **gazebo::ModelPlugin**
*A plugin with access to **physics::Model** (p. 678).*
- class **gazebo::PluginT**< T >
A class which all plugins must inherit from.
- class **gazebo::SensorPlugin**
*A plugin with access to **physics::Sensor**.*
- class **gazebo::SystemPlugin**
A plugin loaded within the gzserver on startup.
- class **gazebo::VisualPlugin**
A plugin loaded within the gzserver on startup.
- class **gazebo::WorldPlugin**
*A plugin with access to **physics::World** (p. 1239).*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

Macros

- #define **GZ_REGISTER_MODEL_PLUGIN**(classname)
Plugin registration function for model plugin.
- #define **GZ_REGISTER_SENSOR_PLUGIN**(classname)
Plugin registration function for sensors.
- #define **GZ_REGISTER_SYSTEM_PLUGIN**(classname)
Plugin registration function for system plugin.
- #define **GZ_REGISTER_VISUAL_PLUGIN**(classname)
Plugin registration function for visual plugin.
- #define **GZ_REGISTER_WORLD_PLUGIN**(classname)
Plugin registration function for world plugin.

Enumerations

- enum **gazebo::PluginType** {
gazebo::WORLD_PLUGIN, gazebo::MODEL_PLUGIN, gazebo::SENSOR_PLUGIN, gazebo::SYSTEM_PLUGIN,
gazebo::VISUAL_PLUGIN }

Used to specify the type of plugin.

11.151.1 Macro Definition Documentation

11.151.1.1 #define GZ_REGISTER_MODEL_PLUGIN(*classname*)

Value:

```
extern "C" GAZEBO_VISIBLE gazebo::ModelPlugin *RegisterPlugin(); \
GAZEBO_VISIBLE \
gazebo::ModelPlugin *RegisterPlugin() \
{ \
    return new classname(); \
}
```

Plugin registration function for model plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.151.1.2 #define GZ_REGISTER_SENSOR_PLUGIN(*classname*)

Value:

```
extern "C" GAZEBO_VISIBLE gazebo::SensorPlugin *RegisterPlugin(); \
GAZEBO_VISIBLE \
gazebo::SensorPlugin *RegisterPlugin() \
{\
    return new classname();\
}
```

Plugin registration function for sensors.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.151.1.3 #define GZ_REGISTER_SYSTEM_PLUGIN(*classname*)

Value:

```
extern "C" GAZEBO_VISIBLE gazebo::SystemPlugin *RegisterPlugin(); \
GAZEBO_VISIBLE \
gazebo::SystemPlugin *RegisterPlugin() \
{\
    return new classname();\
}
```

Plugin registration function for system plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.151.1.4 #define GZ_REGISTER_VISUAL_PLUGIN(*classname*)

Value:

```
extern "C" GAZEBO_VISIBLE gazebo::VisualPlugin *RegisterPlugin(); \
GAZEBO_VISIBLE \
gazebo::VisualPlugin *RegisterPlugin() \
{\
    return new classname();\
}
```

Plugin registration function for visual plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.151.1.5 #define GZ_REGISTER_WORLD_PLUGIN(*classname*)

Value:

```
extern "C" GAZEBO_VISIBLE gazebo::WorldPlugin *RegisterPlugin(); \
GAZEBO_VISIBLE \
gazebo::WorldPlugin *RegisterPlugin() \
{ \
    return new classname(); \
}
```

Plugin registration function for world plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

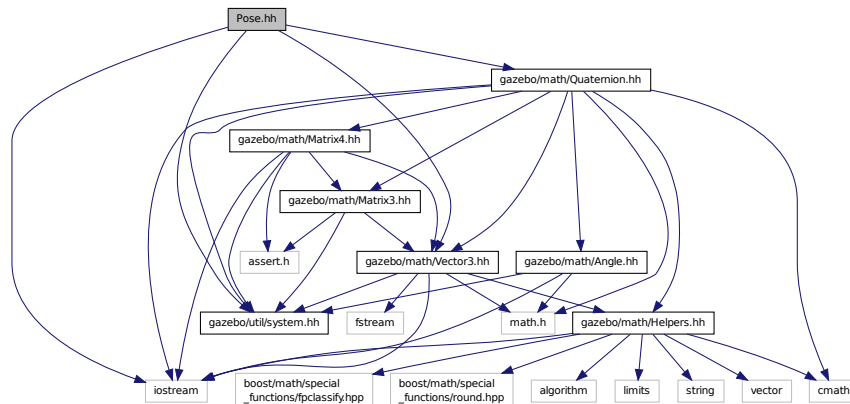
Returns

the name of the registered plugin

11.152 Pose.hh File Reference

```
#include <iostream>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Quaternion.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Pose.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class `gazebo::math::Pose`

Encapsulates a position and rotation in three space.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::math**

Math namespace.

11.153 Projector.hh File Reference

```
#include <string>
#include <map>
#include <list>
#include <sdf/sdf.hh>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/transport.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Projector.hh:



Classes

- class **gazebo::rendering::Projector**

Projects a material onto surface, light a light projector.

- class **gazebo::rendering::Projector::ProjectorFrameListener**

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

Rendering namespace.

11.154 Publication.hh File Reference

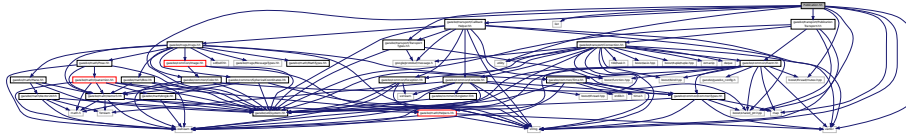
```
#include <utility>
```

```

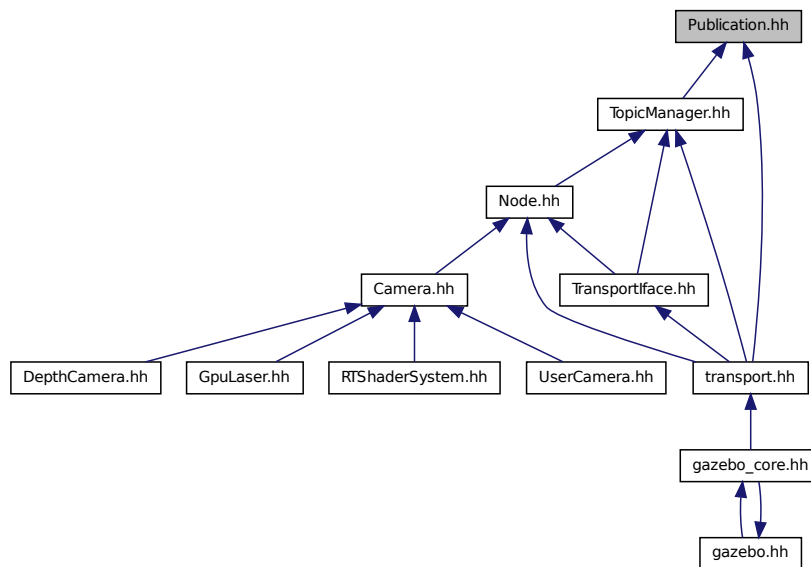
#include <boost/shared_ptr.hpp>
#include <boost/thread/mutex.hpp>
#include <list>
#include <string>
#include <vector>
#include <map>
#include "gazebo/transport/CallbackHelper.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/transport/PublicationTransport.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for Publication.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Publication**
A publication for a topic.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::transport**

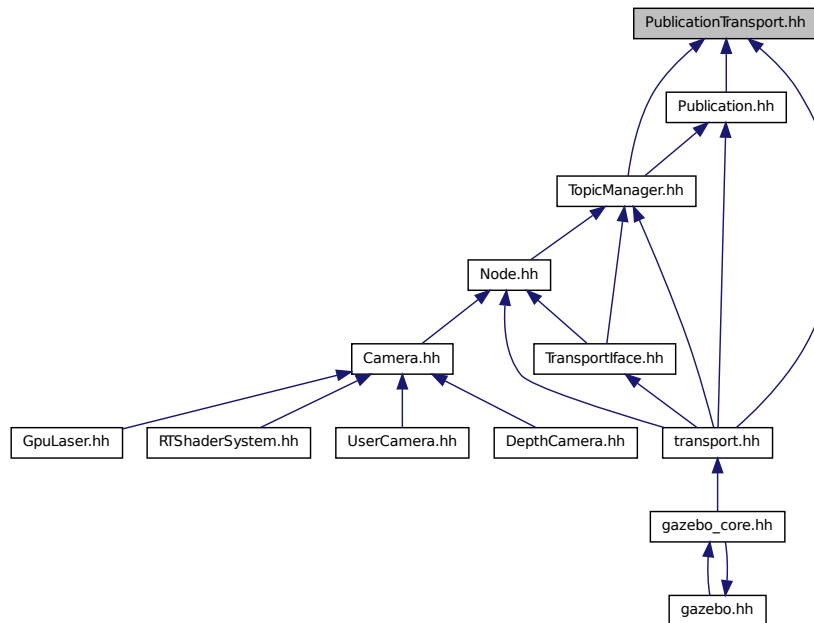
11.155 PublicationTransport.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <string>
#include "gazebo/transport/Connection.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for PublicationTransport.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::PublicationTransport**
transport/transport.hh

Namespaces

- namespace **gazebo**

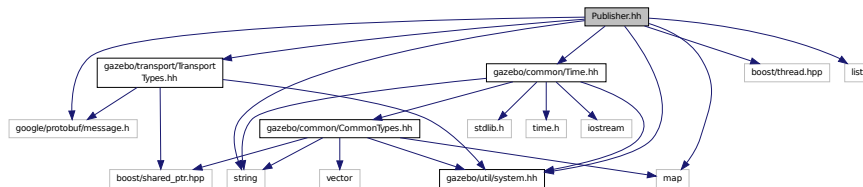
Forward declarations for the common classes.

- namespace **gazebo::transport**

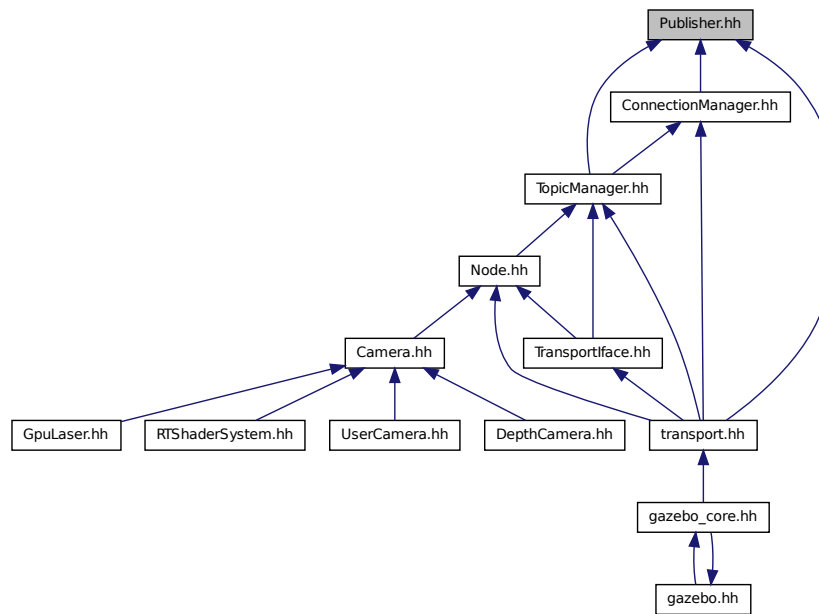
11.156 Publisher.hh File Reference

```
#include <google/protobuf/message.h>
#include <boost/thread.hpp>
#include <string>
#include <list>
#include <map>
#include "gazebo/common/Time.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Publisher.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Publisher**
A publisher of messages on a topic.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

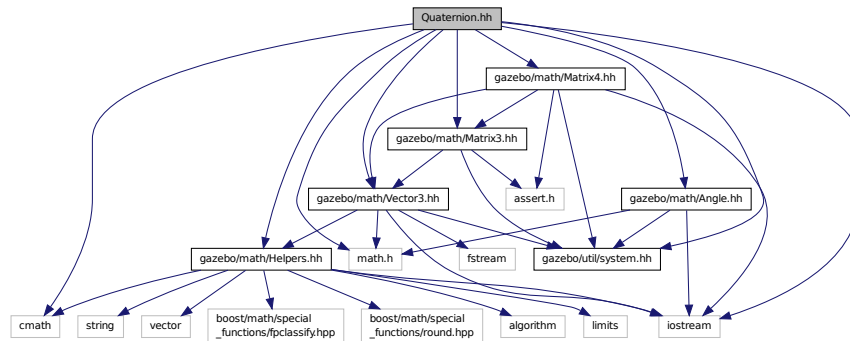
11.157 Quaternion.hh File Reference

```

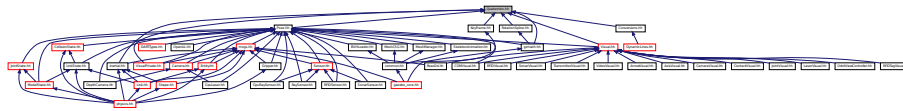
#include <math.h>
#include <iostream>
#include <cmath>
#include "gazebo/math/Helpers.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Matrix3.hh"
#include "gazebo/math/Matrix4.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for Quaternion.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Quaternion**

A quaternion class.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

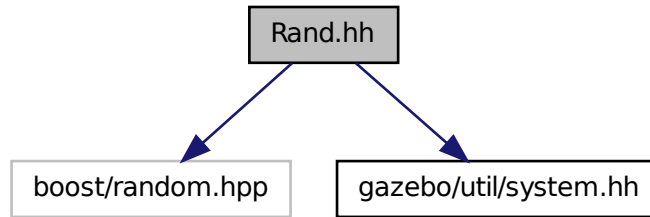
- namespace **gazebo::math**

Math namespace.

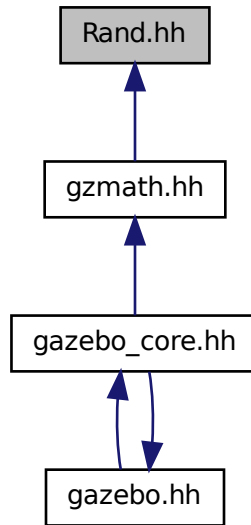
11.158 Rand.hh File Reference

```
#include <boost/random.hpp>
#include "gazebo/util/system.hh"
```

Include dependency graph for Rand.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Rand**
Random number generator class.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::math**

Math namespace.

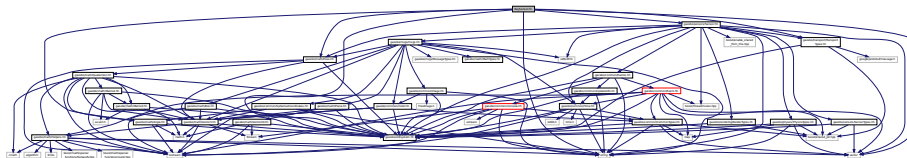
Typedefs

- typedef boost::mt19937 **gazebo::math::GeneratorType**
- typedef
boost::normal_distribution
< double > **gazebo::math::NormalRealDist**
- typedef
boost::variate_generator
< GeneratorType
&, NormalRealDist > **gazebo::math::NRealGen**
- typedef
boost::variate_generator
< GeneratorType
&, UniformIntDist > **gazebo::math::UIntGen**
- typedef boost::uniform_int< int > **gazebo::math::UniformIntDist**
- typedef boost::uniform_real
< double > **gazebo::math::UniformRealDist**
- typedef
boost::variate_generator
< GeneratorType
&, UniformRealDist > **gazebo::math::URealGen**

11.159 RaySensor.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for RaySensor.hh:



Classes

- class **gazebo::sensors::RaySensor**
Sensor (p. 907) with one or more rays.

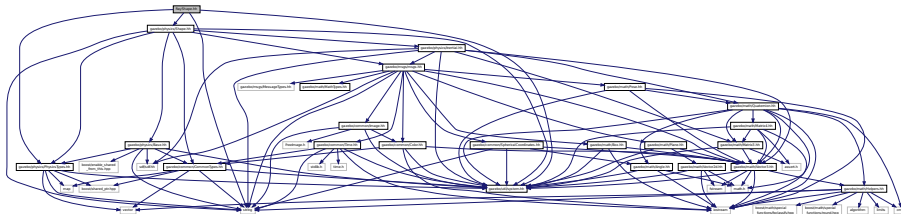
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

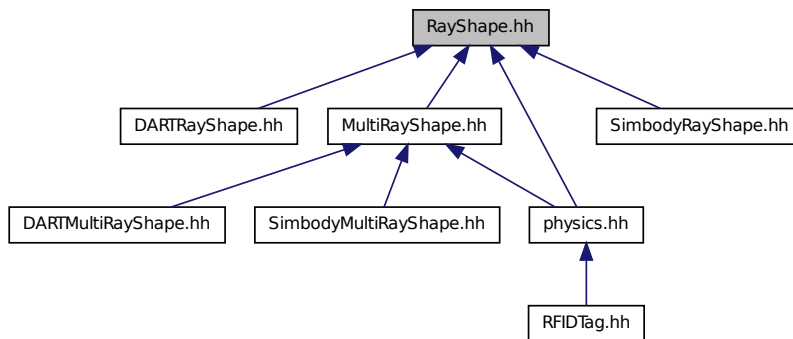
11.160 RayShape.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for RayShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

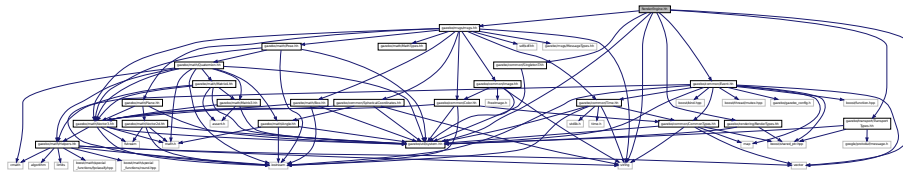
- class **gazebo::physics::RayShape**
Base (p. 168) class for Ray collision geometry.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.161 RenderEngine.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"
Include dependency graph for RenderEngine.hh:
```



Classes

- class **gazebo::rendering::RenderEngine**
Adaptor to Ogre3d.

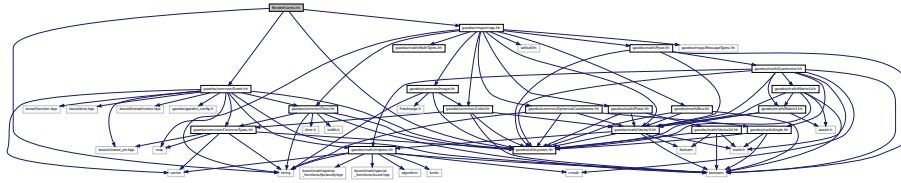
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.162 RenderEvents.hh File Reference

```
#include <string>
#include "gazebo/common/Event.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for RenderEvents.hh:



Classes

- class **gazebo::rendering::Events**
Base class for rendering events.

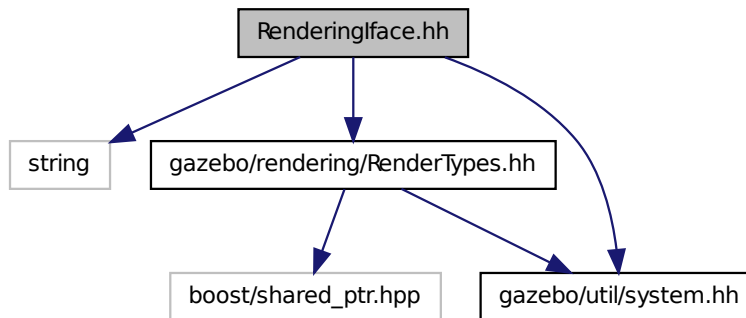
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.163 Renderingface.hh File Reference

```
#include <string>
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Renderingface.hh:



Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

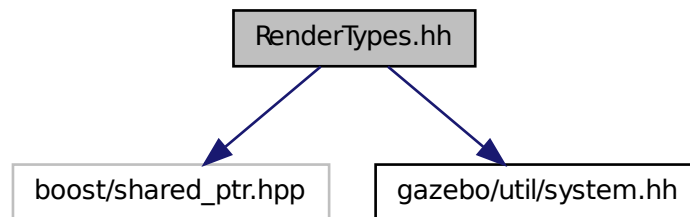
Rendering namespace.

Functions

- **GAZEBO_VISIBLE** rendering::ScenePtr **gazebo::rendering::create_scene** (const std::string &_name, bool _enableVisualizations, bool _isServer=false)
create **rendering::Scene** (p. 879) by name.
- **GAZEBO_VISIBLE** bool **gazebo::rendering::fini** ()
teardown rendering engine.
- **GAZEBO_VISIBLE** rendering::ScenePtr **gazebo::rendering::get_scene** (const std::string &_name="")
get pointer to **rendering::Scene** (p. 879) by name.
- **GAZEBO_VISIBLE** bool **gazebo::rendering::init** ()
init rendering engine.
- **GAZEBO_VISIBLE** bool **gazebo::rendering::load** ()
load rendering engine.
- **GAZEBO_VISIBLE** void **gazebo::rendering::remove_scene** (const std::string &_name)
remove a **rendering::Scene** (p. 879) by name

11.164 RenderTypes.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include "gazebo/util/system.hh"
Include dependency graph for RenderTypes.hh:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

Macros

- #define **GZ_VISIBILITY_ALL** 0x0FFFFFFF
Render everything visibility mask.
- #define **GZ_VISIBILITY_GUI** 0x00000001
Render GUI visuals mask.
- #define **GZ_VISIBILITY_SELECTABLE** 0x00000002
Render visuals that are selectable mask.
- #define **GZ_VISIBILITY_SELECTION** 0x10000000
Renders only objects that can be selected.

Typedefs

- typedef boost::shared_ptr
< ArrowVisual > **gazebo::rendering::ArrowVisualPtr**
- typedef boost::shared_ptr
< AxisVisual > **gazebo::rendering::AxisVisualPtr**
- typedef boost::shared_ptr< Camera > **gazebo::rendering::CameraPtr**
- typedef boost::shared_ptr
< CameraVisual > **gazebo::rendering::CameraVisualPtr**
- typedef boost::shared_ptr
< COMVisual > **gazebo::rendering::COMVisualPtr**
- typedef boost::shared_ptr
< ContactVisual > **gazebo::rendering::ContactVisualPtr**
- typedef boost::shared_ptr
< DepthCamera > **gazebo::rendering::DepthCameraPtr**
- typedef boost::shared_ptr
< DynamicLines > **gazebo::rendering::DynamicLinesPtr**
- typedef boost::shared_ptr
< GpuLaser > **gazebo::rendering::GpuLaserPtr**
- typedef boost::shared_ptr
< JointVisual > **gazebo::rendering::JointVisualPtr**
- typedef boost::shared_ptr
< LaserVisual > **gazebo::rendering::LaserVisualPtr**
- typedef boost::shared_ptr< Light > **gazebo::rendering::LightPtr**
- typedef boost::shared_ptr
< RFIDTagVisual > **gazebo::rendering::RFIDTagVisualPtr**
- typedef boost::shared_ptr
< RFIDVisual > **gazebo::rendering::RFIDVisualPtr**
- typedef boost::shared_ptr< Scene > **gazebo::rendering::ScenePtr**
- typedef boost::shared_ptr
< SelectionObj > **gazebo::rendering::SelectionObjPtr**

- typedef boost::shared_ptr
< SonarVisual > **gazebo::rendering::SonarVisualPtr**
- typedef boost::shared_ptr
< UserCamera > **gazebo::rendering::UserCameraPtr**
- typedef boost::shared_ptr< Visual > **gazebo::rendering::VisualPtr**
- typedef boost::shared_ptr
< WindowManager > **gazebo::rendering::WindowManagerPtr**
- typedef boost::shared_ptr
< WrenchVisual > **gazebo::rendering::WrenchVisualPtr**

Enumerations

- enum **gazebo::rendering::RenderOpType** {
gazebo::rendering::RENDERING_POINT_LIST = 0, **gazebo::rendering::RENDERING_LINE_LIST** = 1,
gazebo::rendering::RENDERING_LINE_STRIP = 2, **gazebo::rendering::RENDERING_TRIANGLE_LIST**
= 3,
gazebo::rendering::RENDERING_TRIANGLE_STRIP = 4, **gazebo::rendering::RENDERING_TRIANGLE_F-**
AN = 5, **gazebo::rendering::RENDERING_MESH_RESOURCE** = 6 }

Type of render operation for a drawable.

11.164.1 Macro Definition Documentation

11.164.1.1 **#define GZ_VISIBILITY_ALL 0xFFFFFFFF**

Render everything visibility mask.

11.164.1.2 **#define GZ_VISIBILITY_GUI 0x00000001**

Render GUI visuals mask.

11.164.1.3 **#define GZ_VISIBILITY_SELECTABLE 0x00000002**

Render visuals that are selectable mask.

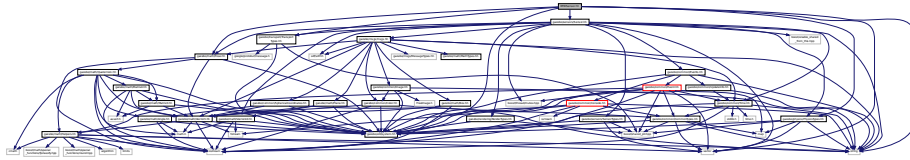
11.164.1.4 **#define GZ_VISIBILITY_SELECTION 0x10000000**

Renders only objects that can be selected.

11.165 RFIDSensor.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for RFIDSensor.hh:



Classes

- class **gazebo::sensors::RFIDSensor**
Sensor (p. 907) class for RFID type of sensor.

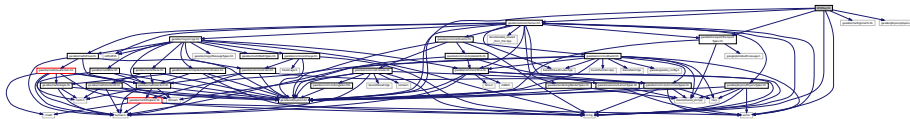
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.166 RFIDTag.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/math/gzmath.hh"
#include "gazebo/physics/physics.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for RFIDTag.hh:



Classes

- class **gazebo::sensors::RFIDTag**
RFIDTag (p. 861) to interact with RFIDTagSensors.

Namespaces

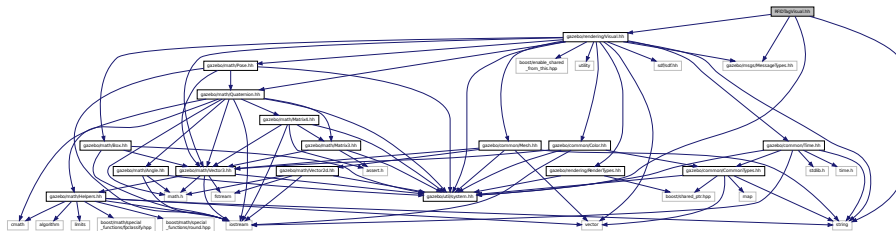
- namespace **gazebo**
Forward declarations for the common classes.

- namespace **gazebo::sensors**

Sensors namespace.

11.167 RFIDTagVisual.hh File Reference

```
#include <string>
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/util/system.hh"
Include dependency graph for RFIDTagVisual.hh:
```



Classes

- class **gazebo::rendering::RFIDTagVisual**

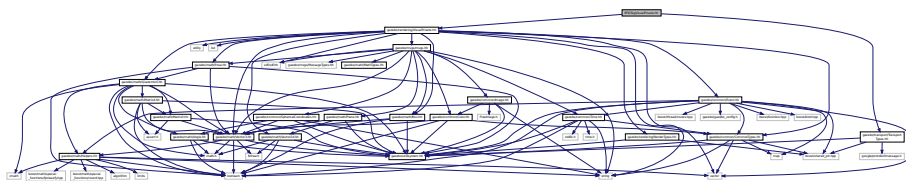
Visualization for RFID tags sensor.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.168 RFIDTagVisualPrivate.hh File Reference

```
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/VisualPrivate.hh"
Include dependency graph for RFIDTagVisualPrivate.hh:
```



Classes

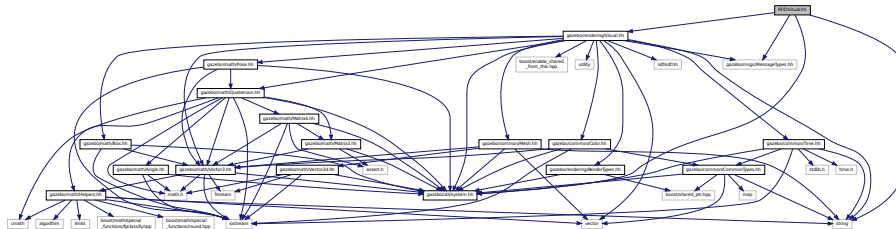
- class **gazebo::rendering::RFIDTagVisualPrivate**
*Private data for the RFID Tag **Visual** (p. 1196) class.*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.169 RFIDVisual.hh File Reference

```
#include <string>
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/util/system.hh"
Include dependency graph for RFIDVisual.hh:
```



Classes

- class **gazebo::rendering::RFIDVisual**
Visualization for RFID sensor.

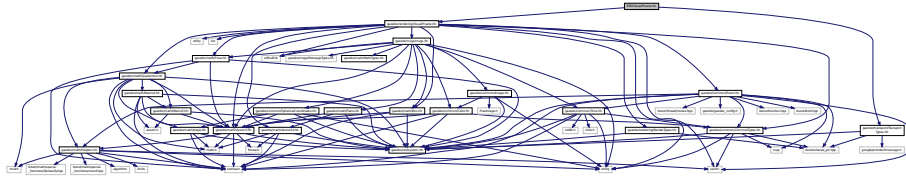
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.170 RFIDVisualPrivate.hh File Reference

```
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/VisualPrivate.hh"
```

Include dependency graph for RFIDVisualPrivate.hh:



Classes

- class **gazebo::rendering::RFIDVisualPrivate**

Private data for the RFID Visual (p. 1196) class.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

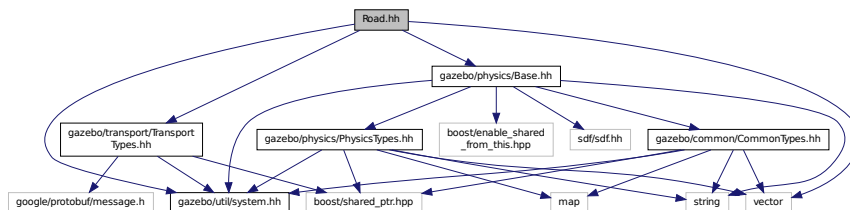
- namespace **gazebo::rendering**

Rendering namespace.

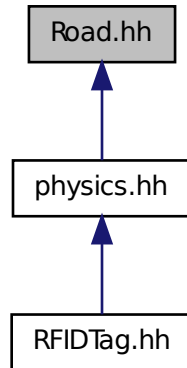
11.171 Road.hh File Reference

```
#include <vector>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/Base.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Road.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Road**
*for building a **Road** (p. 869) from SDF*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.172 Road2d.hh File Reference

```
#include <string>
#include <vector>
#include <list>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Spline.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Road2d.hh:



Classes

- class `gazebo::rendering::Road2d`
- class `gazebo::rendering::Road2d::Segment`

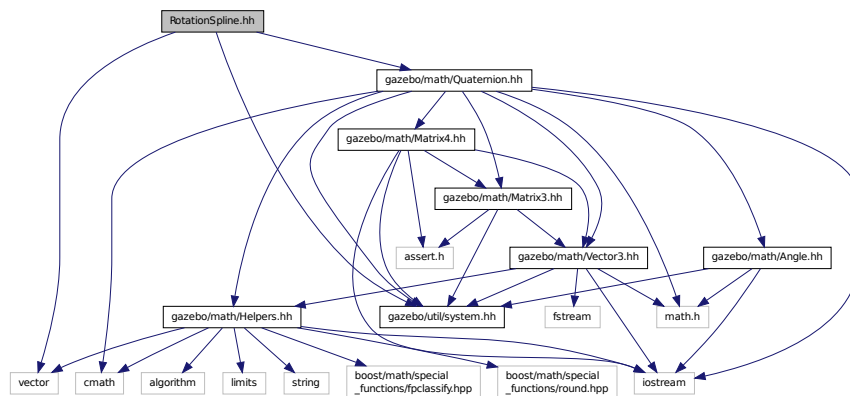
A road segment.

Namespaces

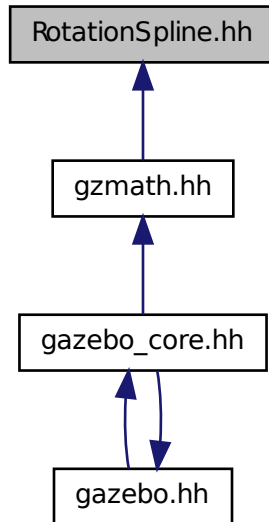
- namespace `gazebo`
Forward declarations for the common classes.
- namespace `gazebo::rendering`
Rendering namespace.

11.173 RotationSpline.hh File Reference

```
#include <vector>
#include "gazebo/math/Quaternion.hh"
#include "gazebo/util/system.hh"
Include dependency graph for RotationSpline.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::RotationSpline**
Spline (p. 1063) for rotations.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

11.174 RTShaderSystem.hh File Reference

```
#include <list>
#include <string>
#include <vector>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/gazebo_config.h"
#include "gazebo/rendering/Camera.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for RTShaderSystem.hh:



Classes

- class **gazebo::rendering::RTShaderSystem**

Implements **Ogre** (p. 137)'s Run-Time Shader system.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

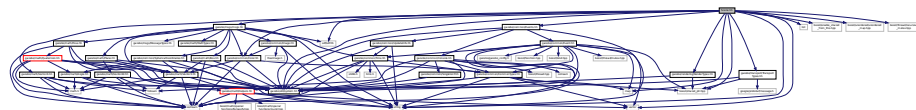
- namespace **gazebo::rendering**

Rendering namespace.

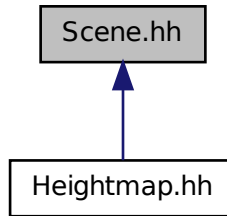
11.175 Scene.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include <list>
#include <boost/enable_shared_from_this.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/unordered/unordered_map.hpp>
#include <boost/thread/recursive_mutex.hpp>
#include <sdf/sdf.hh>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/common/Color.hh"
#include "gazebo/math/Vector2i.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Scene.hh:



This graph shows which files directly or indirectly include this file:



Classes

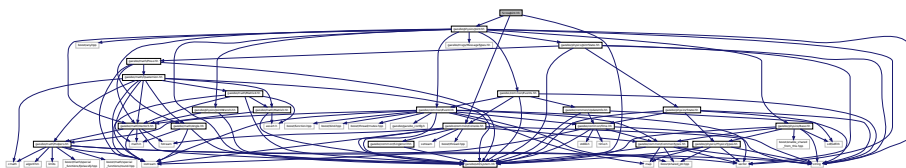
- class **gazebo::rendering::Scene**
Representation of an entire scene graph.

Namespaces

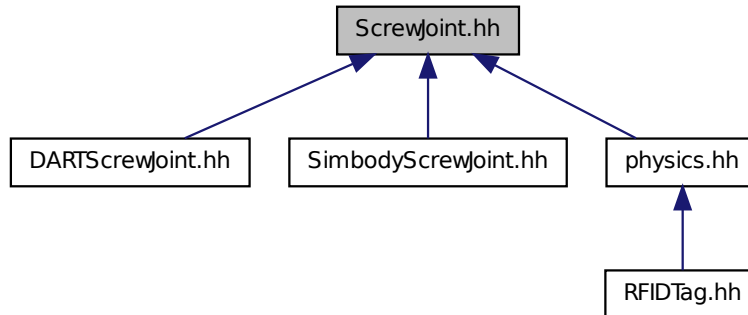
- namespace **boost**
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**
- namespace **SkyX**

11.176 ScrewJoint.hh File Reference

```
#include "gazebo/physics/Joint.hh"
#include "gazebo/common/Console.hh"
#include "gazebo/util/system.hh"
Include dependency graph for ScrewJoint.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ScrewJoint**< T >

A screw joint, which has both prismatic and rotational DOFs.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

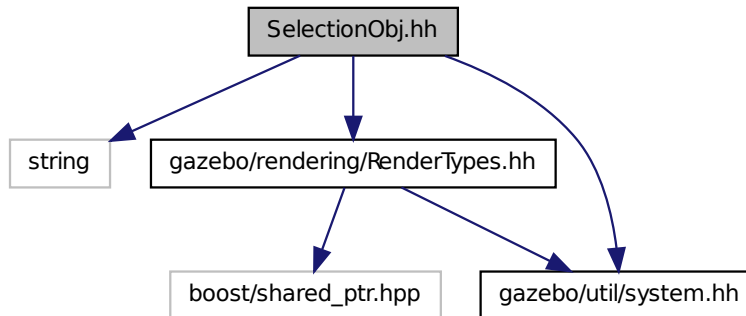
- namespace **gazebo::physics**

namespace for physics

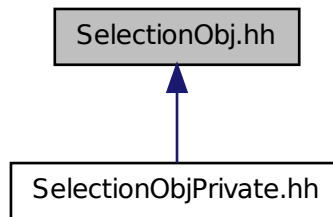
11.177 SelectionObj.hh File Reference

```
#include <string>
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SelectionObj.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::SelectionObj**
Interactive selection object for models and links.

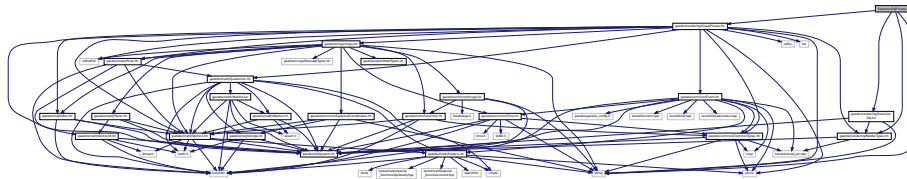
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.178 SelectionObjPrivate.hh File Reference

```
#include <string>
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/rendering/VisualPrivate.hh"
#include "gazebo/rendering/SelectionObj.hh"
```

Include dependency graph for SelectionObjPrivate.hh:



Classes

- class **gazebo::rendering::SelectionObjPrivate**

Private data for the Selection Obj class.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

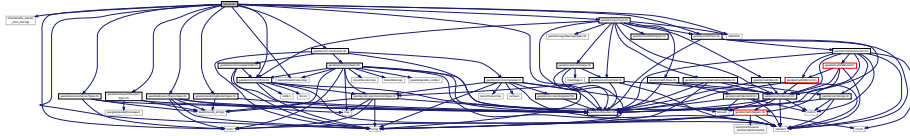
- namespace **gazebo::rendering**

Rendering namespace.

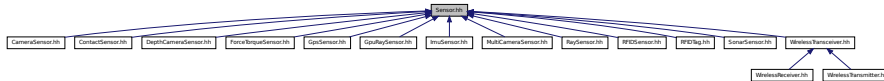
11.179 Sensor.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <boost/thread/mutex.hpp>
#include <vector>
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/sensors/SensorTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Sensor.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::sensors::Sensor**

Base class for sensors.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::sensors**

Sensors namespace.

Enumerations

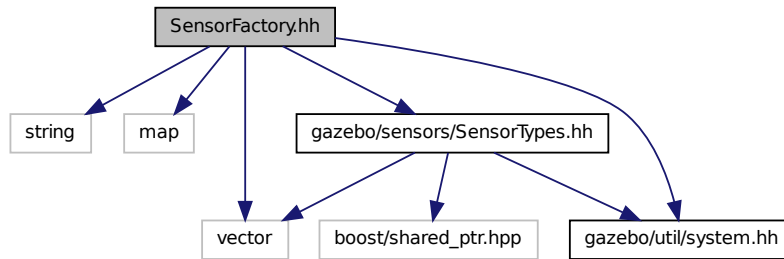
- enum **gazebo::sensors::SensorCategory** { **gazebo::sensors::IMAGE** = 0, **gazebo::sensors::RAY** = 1, **gazebo::sensors::OTHER** = 2, **gazebo::sensors::CATEGORY_COUNT** = 3 }

SensorClass is used to categorize sensors.

11.180 SensorFactory.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "gazebo/sensors/SensorTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SensorFactory.hh:



Classes

- class **gazebo::sensors::SensorFactory**

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

Macros

- #define **GZ_REGISTER_STATIC_SENSOR**(name, classname)
Static sensor registration macro.

Typedefs

- typedef Sensor *(* **gazebo::sensors::SensorFactoryFn**)()

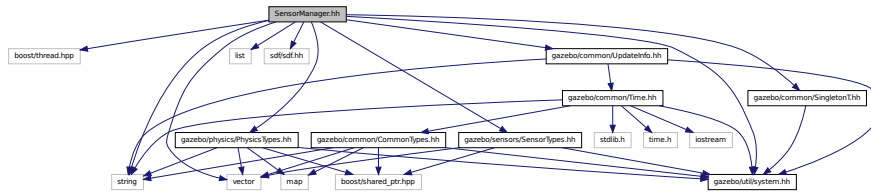
11.181 SensorManager.hh File Reference

```

#include <boost/thread.hpp>
#include <string>
#include <vector>
#include <list>
#include <sdf/sdf.hh>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/sensors/SensorTypes.hh"
#include "gazebo/util/system.hh"

```


Include dependency graph for SensorManager.hh:



Classes

- class **gazebo::sensors::SensorManager::ImageSensorContainer**
- class **gazebo::sensors::SensorManager::SensorContainer**
- class **gazebo::sensors::SensorManager**

Class to manage and update all sensors.
- class **gazebo::sensors::SimTimeEvent**
- class **gazebo::sensors::SimTimeEventHandler**

Monitors simulation time, and notifies conditions when a specified time has been reached.

Namespaces

- namespace **gazebo**

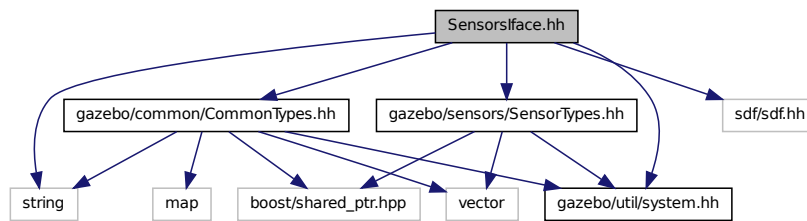
Forward declarations for the common classes.
- namespace **gazebo::sensors**

Sensors namespace.

11.182 Sensorsface.hh File Reference

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/sensors/SensorTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SensorsIface.hh:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

Functions

- **GAZEBO_VISIBLE** `std::string gazebo::sensors::create_sensor (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName)` **GAZEBO_DEPRECATED(2.0)**
Deprecated.
- **GAZEBO_VISIBLE** `std::string gazebo::sensors::create_sensor (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName, uint32_t _parentId)`
Create a sensor using SDF.
- **GAZEBO_VISIBLE** `void gazebo::sensors::disable ()`
Disable sensors.
- **GAZEBO_VISIBLE** `void gazebo::sensors::enable ()`
Enable sensors.
- **GAZEBO_VISIBLE** `bool gazebo::sensors::fini ()`
shutdown the sensor generation loop.
- **GAZEBO_VISIBLE** `SensorPtr gazebo::sensors::get_sensor (const std::string &_name)`
Get a sensor using by name.
- **GAZEBO_VISIBLE** `bool gazebo::sensors::init ()`
initialize the sensor generation loop.
- **GAZEBO_VISIBLE** `bool gazebo::sensors::load ()`
Load the sensor library.
- **GAZEBO_VISIBLE** `void gazebo::sensors::remove_sensor (const std::string &_sensorName)`
Remove a sensor by name.
- **GAZEBO_VISIBLE** `bool gazebo::sensors::remove_sensors ()`
Remove all sensors.
- **GAZEBO_VISIBLE** `void gazebo::sensors::run_once (bool _force=false)`
Run the sensor generation one step.
- **GAZEBO_VISIBLE** `void gazebo::sensors::run_threads ()`

Run sensors in a threads. This is a non-blocking call.

- **GAZEBO_VISIBLE** void `gazebo::sensors::stop ()`

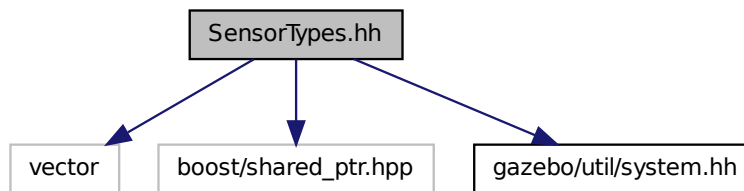
Stop the sensor generation loop.

11.183 SensorTypes.hh File Reference

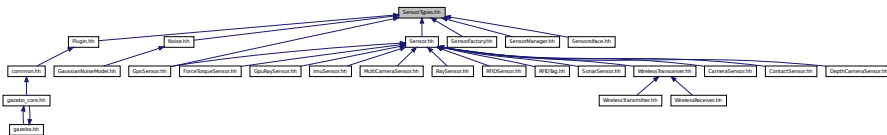
Forward declarations and typedefs for sensors.

```
#include <vector>
#include <boost/shared_ptr.hpp>
#include "gazebo/util/system.hh"
```

Include dependency graph for SensorTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

Typedefs

- typedef `std::vector`
< CameraSensorPtr > **gazebo::sensors::CameraSensor_V**
- typedef `boost::shared_ptr`
< CameraSensor > **gazebo::sensors::CameraSensorPtr**

- typedef std::vector
 < ContactSensorPtr > **gazebo::sensors::ContactSensor_V**
- typedef boost::shared_ptr
 < ContactSensor > **gazebo::sensors::ContactSensorPtr**
- typedef std::vector
 < DepthCameraSensorPtr > **gazebo::sensors::DepthCameraSensor_V**
- typedef boost::shared_ptr
 < DepthCameraSensor > **gazebo::sensors::DepthCameraSensorPtr**
- typedef boost::shared_ptr
 < ForceTorqueSensor > **gazebo::sensors::ForceTorqueSensorPtr**
- typedef boost::shared_ptr
 < GaussianNoiseModel > **gazebo::sensors::GaussianNoiseModelPtr**
- typedef boost::shared_ptr
 < GpsSensor > **gazebo::sensors::GpsSensorPtr**
- typedef std::vector
 < GpuRaySensorPtr > **gazebo::sensors::GpuRaySensor_V**
- typedef boost::shared_ptr
 < GpuRaySensor > **gazebo::sensors::GpuRaySensorPtr**
- typedef boost::shared_ptr
 < ImageGaussianNoiseModel > **gazebo::sensors::ImageGaussianNoiseModelPtr**
 *Shared pointer to **Noise** (p. 748).*
- typedef std::vector< ImuSensorPtr > **gazebo::sensors::ImuSensor_V**
- typedef boost::shared_ptr
 < ImuSensor > **gazebo::sensors::ImuSensorPtr**
- typedef std::vector
 < MultiCameraSensorPtr > **gazebo::sensors::MultiCameraSensor_V**
- typedef boost::shared_ptr
 < MultiCameraSensor > **gazebo::sensors::MultiCameraSensorPtr**
- typedef boost::shared_ptr< Noise > **gazebo::sensors::NoisePtr**
- typedef std::vector< RaySensorPtr > **gazebo::sensors::RaySensor_V**
- typedef boost::shared_ptr
 < RaySensor > **gazebo::sensors::RaySensorPtr**
- typedef std::vector< RFIDSensor > **gazebo::sensors::RFIDSensor_V**
- typedef boost::shared_ptr
 < RFIDSensor > **gazebo::sensors::RFIDSensorPtr**
- typedef std::vector< RFIDTag > **gazebo::sensors::RFIDTag_V**
- typedef boost::shared_ptr
 < RFIDTag > **gazebo::sensors::RFIDTagPtr**
- typedef std::vector< SensorPtr > **gazebo::sensors::Sensor_V**
- typedef boost::shared_ptr< Sensor > **gazebo::sensors::SensorPtr**
- typedef boost::shared_ptr
 < SonarSensor > **gazebo::sensors::SonarSensorPtr**
- typedef std::vector
 < WirelessReceiver > **gazebo::sensors::WirelessReceiver_V**
- typedef boost::shared_ptr
 < WirelessReceiver > **gazebo::sensors::WirelessReceiverPtr**
- typedef std::vector
 < WirelessTransceiver > **gazebo::sensors::WirelessTransceiver_V**
- typedef boost::shared_ptr
 < WirelessTransceiver > **gazebo::sensors::WirelessTransceiverPtr**
- typedef std::vector
 < WirelessTransmitter > **gazebo::sensors::WirelessTransmitter_V**
- typedef boost::shared_ptr
 < WirelessTransmitter > **gazebo::sensors::WirelessTransmitterPtr**

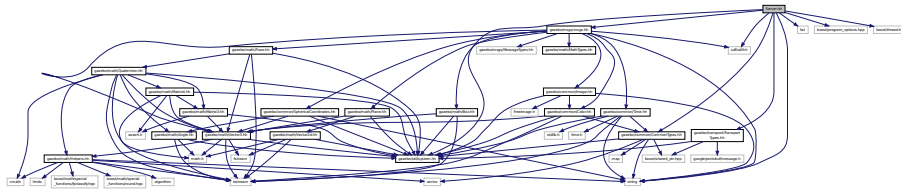
11.183.1 Detailed Description

Forward declarations and typedefs for sensors.

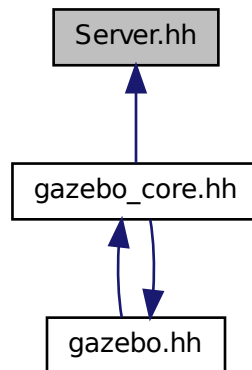
11.184 Server.hh File Reference

```
#include <string>
#include <list>
#include <boost/program_options.hpp>
#include <boost/thread.hpp>
#include <sdf/sdf.hh>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Server.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::Server**

Namespaces

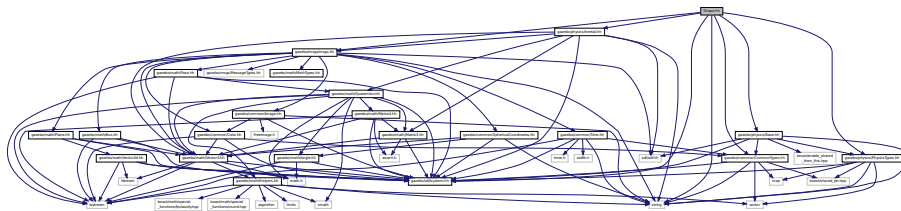
- namespace **boost**
- namespace **gazebo**

Forward declarations for the common classes.

11.185 Shape.hh File Reference

```
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Inertial.hh"
#include "gazebo/physics/Base.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Shape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Shape**
Base (p. 168) class for all shapes.

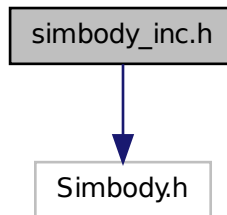
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

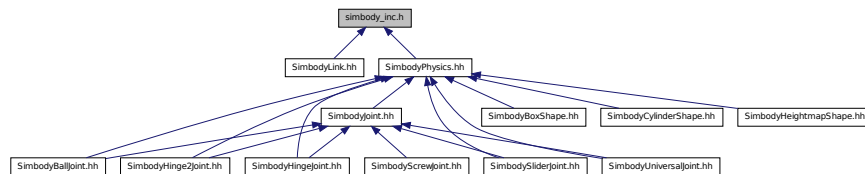
11.186 simbody_inc.h File Reference

```
#include <Simbody.h>
```

Include dependency graph for simbody_inc.h:



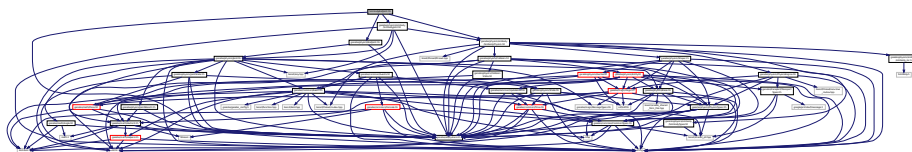
This graph shows which files directly or indirectly include this file:



11.187 SimbodyBallJoint.hh File Reference

```
#include "gazebo/physics/BallJoint.hh"
#include "gazebo/physics/simbody/SimbodyJoint.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SimbodyBallJoint.hh:



Classes

- class **gazebo::physics::SimbodyBallJoint**
SimbodyBallJoint (p. 936) class models a ball joint in Simbody.

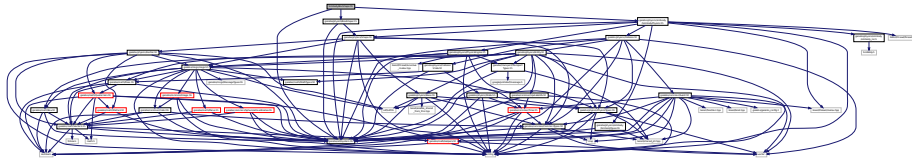
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.188 SimbodyBoxShape.hh File Reference

```
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/physics/BoxShape.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SimbodyBoxShape.hh:



Classes

- class **gazebo::physics::SimbodyBoxShape**
Simbody box collision.

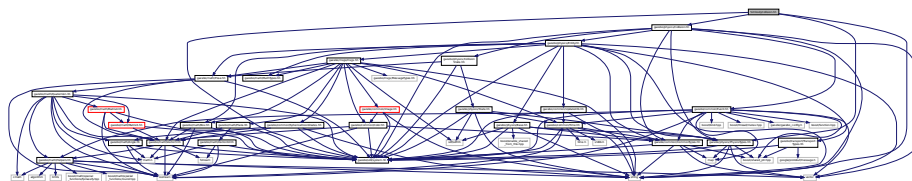
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.189 SimbodyCollision.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SimbodyCollision.hh:



Classes

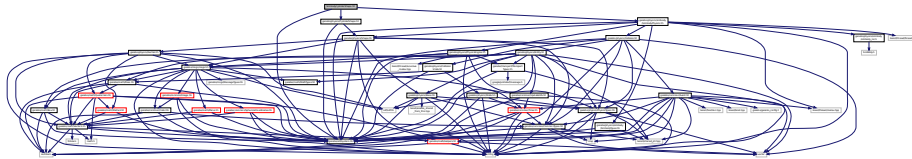
- class **gazebo::physics::SimbodyCollision**
Simbody collisions.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics
- namespace **SimTK**

11.190 SimbodyCylinderShape.hh File Reference

```
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/physics/CylinderShape.hh"
#include "gazebo/util/system.hh"
Include dependency graph for SimbodyCylinderShape.hh:
```



Classes

- class **gazebo::physics::SimbodyCylinderShape**
Cylinder collision.

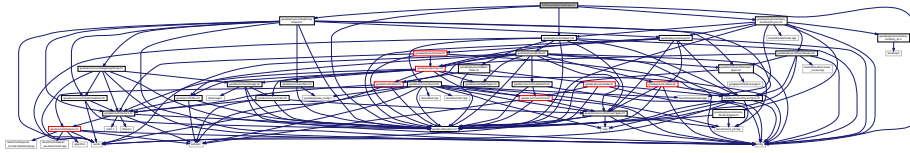
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.191 SimbodyHeightmapShape.hh File Reference

```
#include <string>
#include "gazebo/physics/HeightmapShape.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SimbodyHeightmapShape.hh:



Classes

- class **gazebo::physics::SimbodyHeightmapShape**
Height map collision.

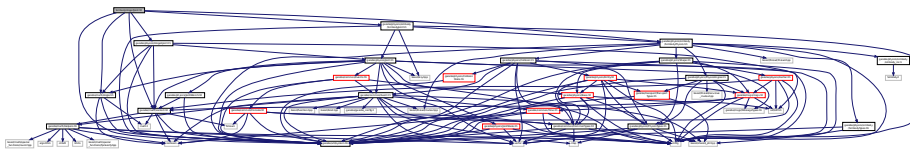
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.192 SimbodyHinge2Joint.hh File Reference

```
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/Hinge2Joint.hh"
#include "gazebo/physics/simbody/SimbodyJoint.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SimbodyHinge2Joint.hh:



Classes

- class **gazebo::physics::SimbodyHinge2Joint**
A two axis hinge joint.

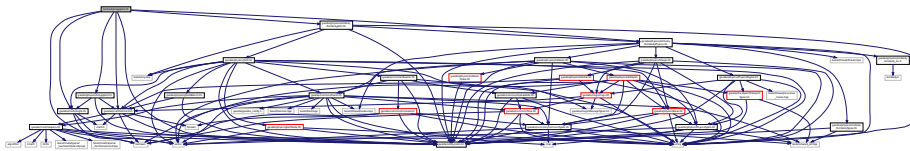
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.193 SimbodyHingeJoint.hh File Reference

```
#include <vector>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/HingeJoint.hh"
#include "gazebo/physics/simbody/SimbodyJoint.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SimbodyHingeJoint.hh:



Classes

- class **gazebo::physics::SimbodyHingeJoint**

A single axis hinge joint.

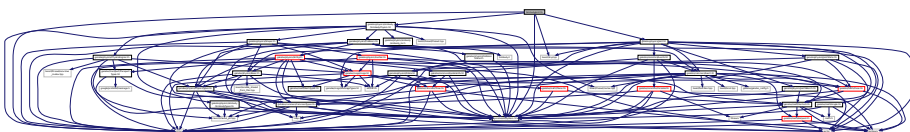
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

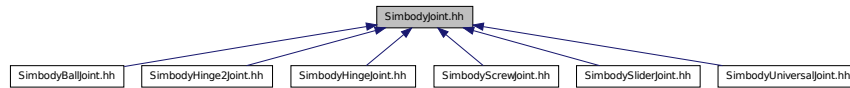
11.194 SimbodyJoint.hh File Reference

```
#include <boost/any.hpp>
#include <string>
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/physics/Joint.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SimbodyJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::SimbodyJoint**
Base (p. 168) class for all joints.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

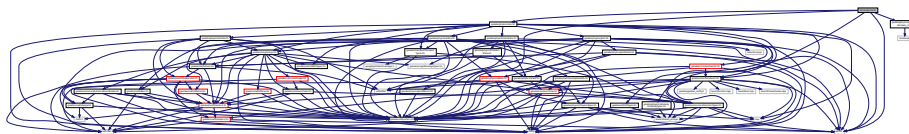
11.195 SimbodyLink.hh File Reference

```

#include <vector>
#include "gazebo/physics/simbody/SimbodyTypes.hh"
#include "gazebo/physics/Link.hh"
#include "gazebo/physics/simbody/simbody_inc.h"
#include "gazebo/util/system.hh"

```

Include dependency graph for SimbodyLink.hh:



Classes

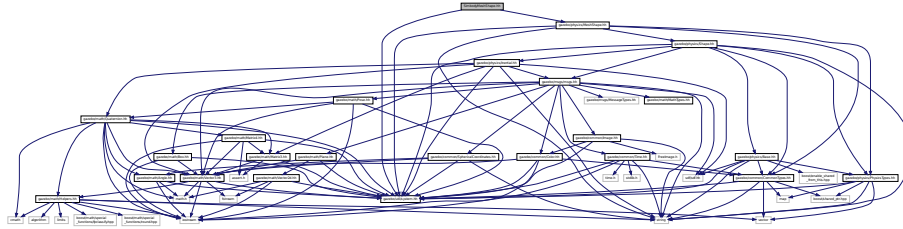
- class **gazebo::physics::SimbodyLink**
*Simbody **Link** (p. 595) class.*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.196 SimbodyMeshShape.hh File Reference

```
#include "gazebo/physics/MeshShape.hh"
#include "gazebo/util/system.hh"
Include dependency graph for SimbodyMeshShape.hh:
```



Classes

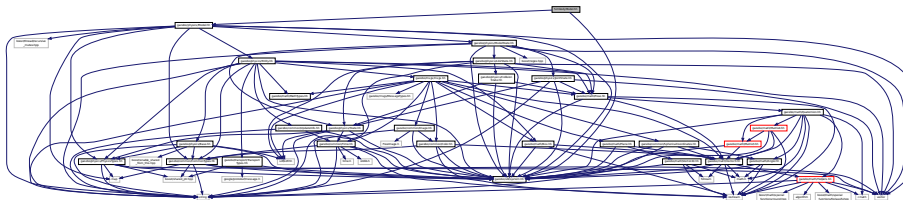
- class **gazebo::physics::SimbodyMeshShape**
Triangle mesh collision.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.197 SimbodyModel.hh File Reference

```
#include "gazebo/physics/Model.hh"
#include "gazebo/util/system.hh"
Include dependency graph for SimbodyModel.hh:
```



Classes

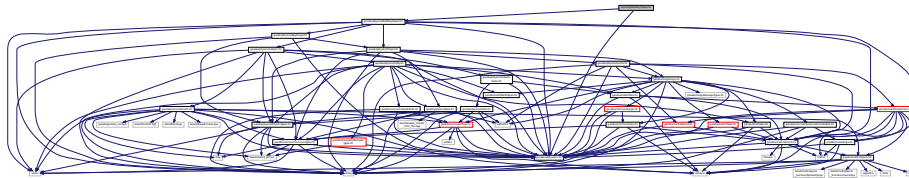
- class **gazebo::physics::SimbodyModel**
A model is a collection of links, joints, and plugins.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.198 SimbodyMultiRayShape.hh File Reference

```
#include "gazebo/physics/MultiRayShape.hh"
#include "gazebo/util/system.hh"
Include dependency graph for SimbodyMultiRayShape.hh:
```



Classes

- class **gazebo::physics::SimbodyMultiRayShape**
*Simbody specific version of **MultiRayShape** (p. 723).*

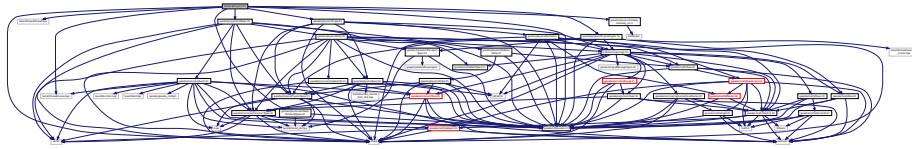
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

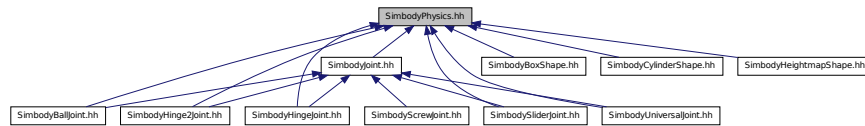
11.199 SimbodyPhysics.hh File Reference

```
#include <string>
#include <boost/thread/thread.hpp>
#include <boost/thread/mutex.hpp>
#include "gazebo/physics/PhysicsEngine.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/physics/simbody/SimbodyTypes.hh"
#include "gazebo/physics/simbody/simbody_inc.h"
#include "gazebo/util/system.hh"
```

Include dependency graph for SimbodyPhysics.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::SimbodyPhysics**
Simbody physics engine.

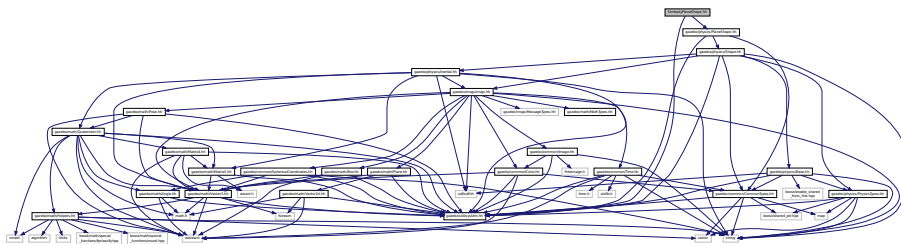
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.200 SimbodyPlaneShape.hh File Reference

```
#include "gazebo/physics/PlaneShape.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SimbodyPlaneShape.hh:



Classes

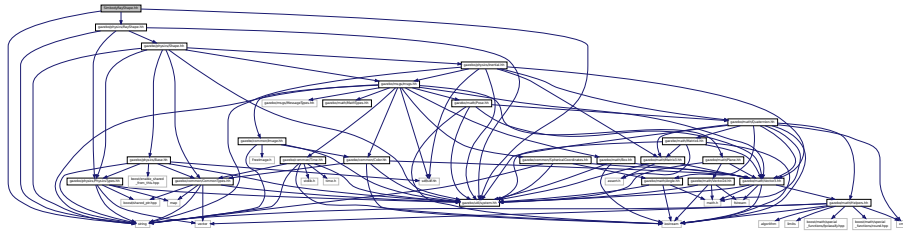
- class **gazebo::physics::SimbodyPlaneShape**
Simbody collision for an infinite plane.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.201 SimbodyRayShape.hh File Reference

```
#include <string>
#include "gazebo/physics/RayShape.hh"
#include "gazebo/util/system.hh"
Include dependency graph for SimbodyRayShape.hh:
```



Classes

- class **gazebo::physics::SimbodyRayShape**
Ray shape for simbody.

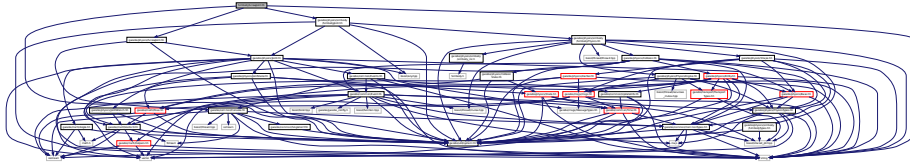
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.202 SimbodyScrewJoint.hh File Reference

```
#include <string>
#include "gazebo/physics/simbody/SimbodyJoint.hh"
#include "gazebo/physics/ScrewJoint.hh"
#include "gazebo/util/system.hh"
```


Include dependency graph for SimbodyScrewJoint.hh:



Classes

- class **gazebo::physics::SimbodyScrewJoint**
A screw joint.

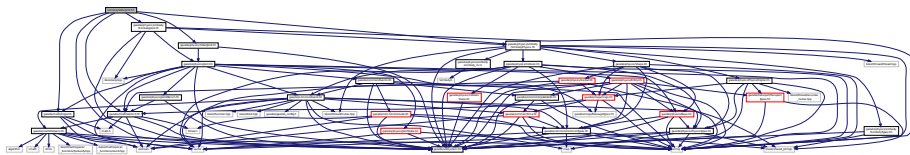
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.203 SimbodySliderJoint.hh File Reference

```
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/simbody/SimbodyJoint.hh"
#include "gazebo/physics/SliderJoint.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SimbodySliderJoint.hh:



Classes

- class **gazebo::physics::SimbodySliderJoint**
A slider joint.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.

- namespace **gazebo::physics**

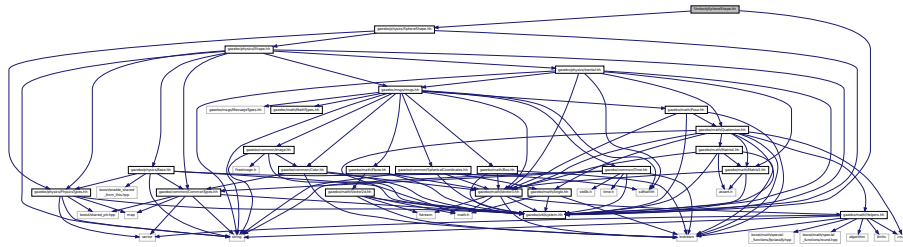
namespace for physics

11.204 SimbodySphereShape.hh File Reference

```
#include "gazebo/physics/SphereShape.hh"
```

```
#include "gazebo/util/system.hh"
```

Include dependency graph for SimbodySphereShape.hh:



Classes

- class **gazebo::physics::SimbodySphereShape**

Simbody sphere collision.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

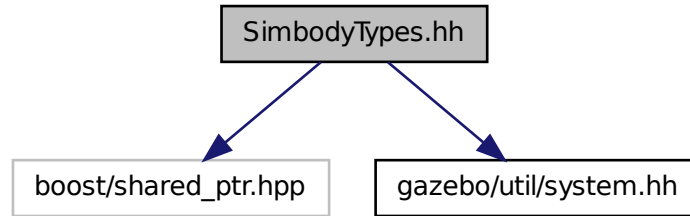
11.205 SimbodyTypes.hh File Reference

Simbody wrapper forward declarations and typedefs.

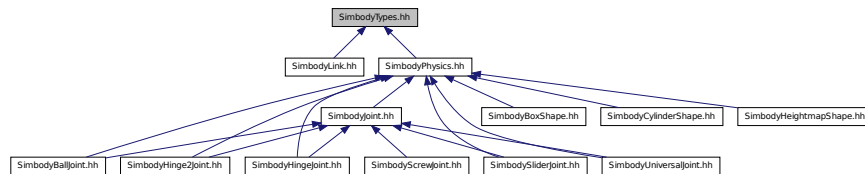
```
#include <boost/shared_ptr.hpp>
```

```
#include "gazebo/util/system.hh"
```

Include dependency graph for SimbodyTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Typedefs

- typedef boost::shared_ptr
< SimbodyCollision > **gazebo::physics::SimbodyCollisionPtr**
- typedef boost::shared_ptr
< SimbodyLink > **gazebo::physics::SimbodyLinkPtr**
- typedef boost::shared_ptr
< SimbodyModel > **gazebo::physics::SimbodyModelPtr**
- typedef boost::shared_ptr
< SimbodyPhysics > **gazebo::physics::SimbodyPhysicsPtr**
- typedef boost::shared_ptr
< SimbodyRayShape > **gazebo::physics::SimbodyRayShapePtr**

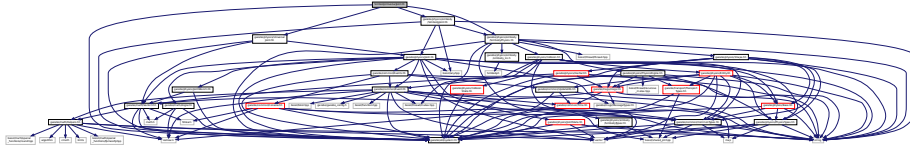
11.205.1 Detailed Description

Simbody wrapper forward declarations and typedefs.

11.206 SimbodyUniversalJoint.hh File Reference

```
#include "gazebo/physics/UniversalJoint.hh"
#include "gazebo/physics/simbody/SimbodyJoint.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SimbodyUniversalJoint.hh:



Classes

- class **gazebo::physics::SimbodyUniversalJoint**
A simbody universal joint class.

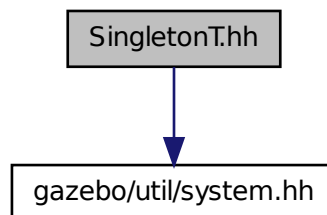
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

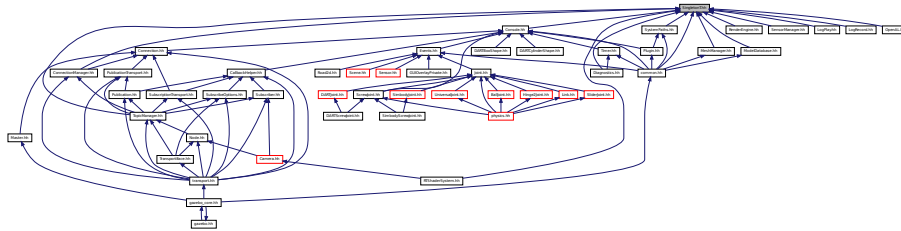
11.207 SingletonT.hh File Reference

```
#include "gazebo/util/system.hh"
```

Include dependency graph for SingletonT.hh:



This graph shows which files directly or indirectly include this file:



Classes

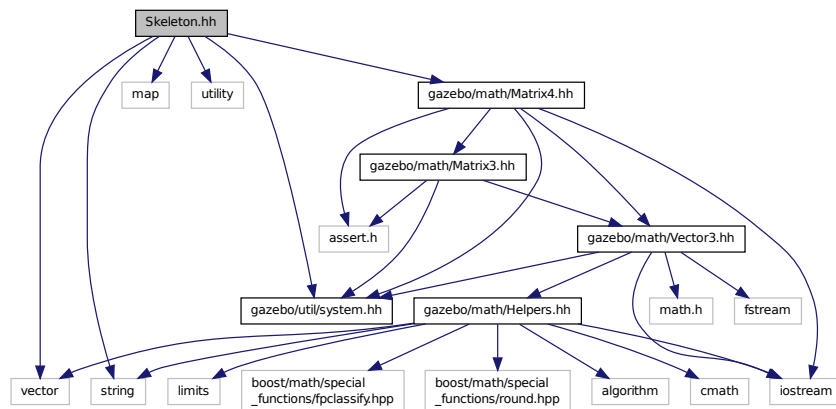
- class **SingletonT**< T >

Singleton template class.

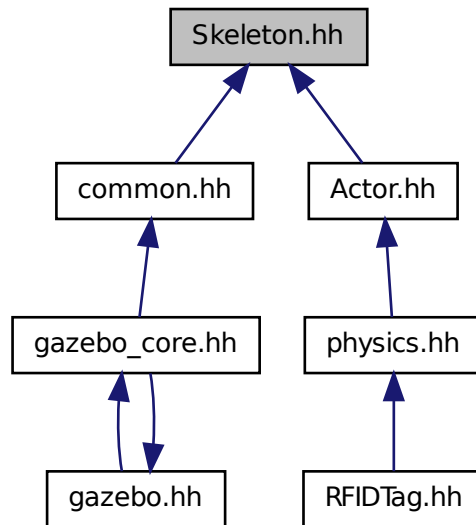
11.208 Skeleton.hh File Reference

```
#include <vector>
#include <string>
#include <map>
#include <utility>
#include "gazebo/math/Matrix4.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Skeleton.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::NodeTransform**
NodeTransform (p. 742) *Skeleton.hh* (p. 1465) *common/common.hh*
- class **gazebo::common::Skeleton**
A skeleton.
- class **gazebo::common::SkeletonNode**
A skeleton node.

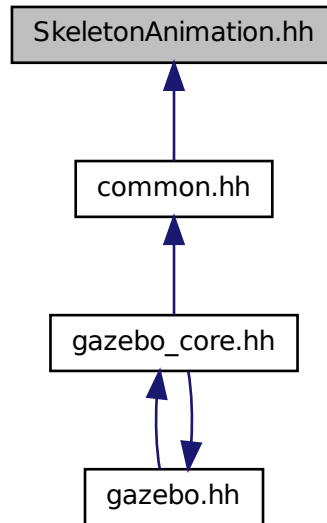
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

Typedefs

- typedef `std::map< unsigned int, SkeletonNode * >` **gazebo::common::NodeMap**
- typedef `std::map< unsigned int, SkeletonNode * >::iterator` **gazebo::common::NodeMapIter**

This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::NodeAnimation**
Node animation.
- class **gazebo::common::SkeletonAnimation**
***Skeleton** (p. 1024) animation.*

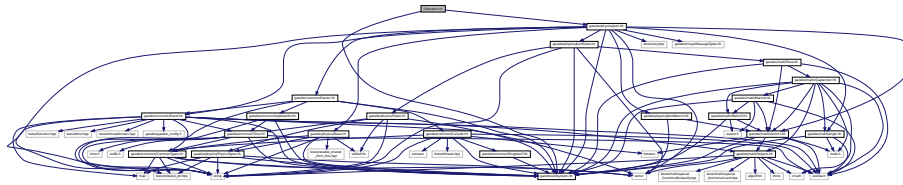
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

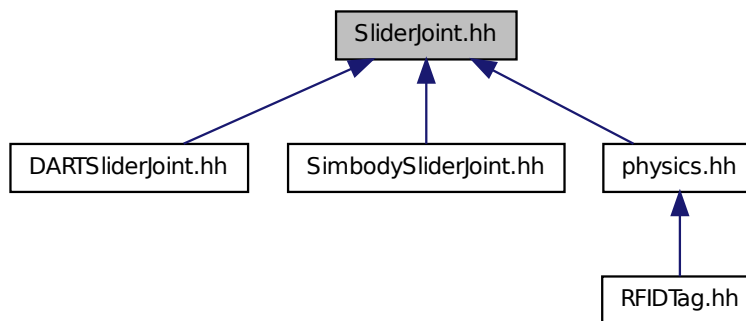
11.210 SliderJoint.hh File Reference

```
#include "gazebo/physics/Joint.hh"  
#include "gazebo/util/system.hh"
```


Include dependency graph for SliderJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::SliderJoint**< T >
A slider joint.

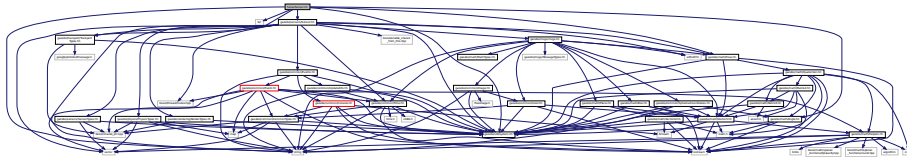
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.211 SonarSensor.hh File Reference

```
#include <string>
```

```
#include <list>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/util/system.hh"
Include dependency graph for SonarSensor.hh:
```



Classes

- class **gazebo::sensors::SonarSensor**

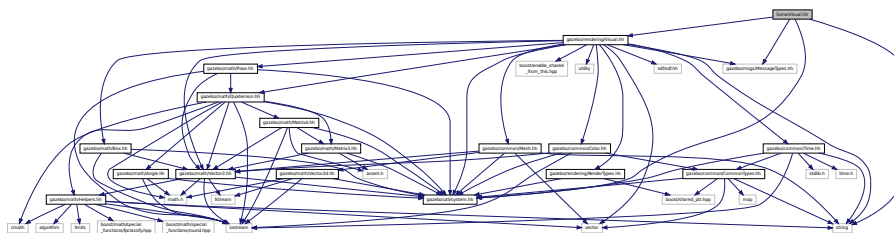
Sensor (p. 907) with sonar cone.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.212 SonarVisual.hh File Reference

```
#include <string>
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/util/system.hh"
Include dependency graph for SonarVisual.hh:
```



Classes

- class **gazebo::rendering::SonarVisual**

Visualization for sonar data.

Namespaces

- namespace **gazebo**

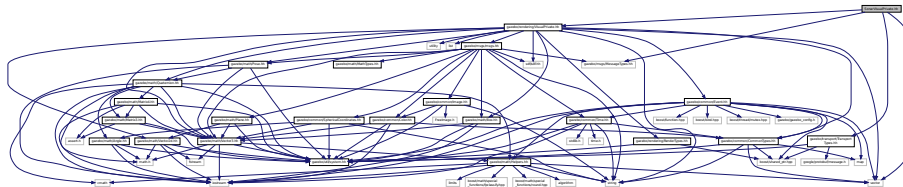
Forward declarations for the common classes.

- namespace **gazebo::rendering**

Rendering namespace.

11.213 SonarVisualPrivate.hh File Reference

```
#include <vector>
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/VisualPrivate.hh"
Include dependency graph for SonarVisualPrivate.hh:
```



Classes

- class **gazebo::rendering::SonarVisualPrivate**

*Private data for the Sonar **Visual** (p. 1196) class.*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

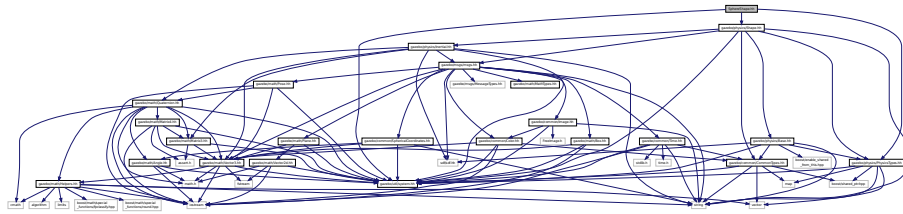
- namespace **gazebo::rendering**

Rendering namespace.

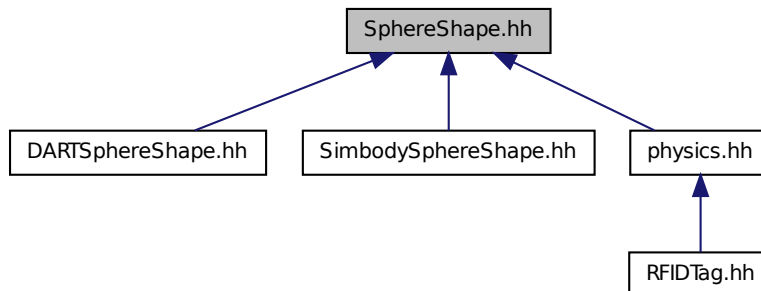
11.214 SphereShape.hh File Reference

```
#include "gazebo/physics/Shape.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SphereShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::SphereShape**
Sphere collision shape.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

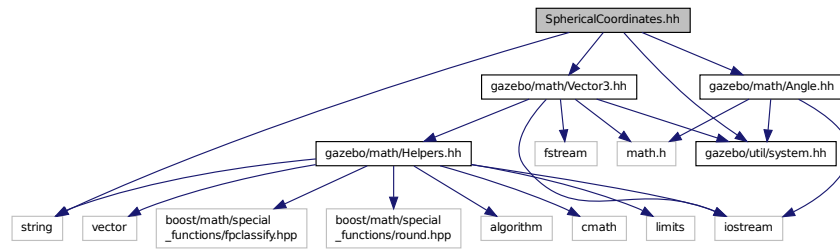
11.215 SphericalCoordinates.hh File Reference

```

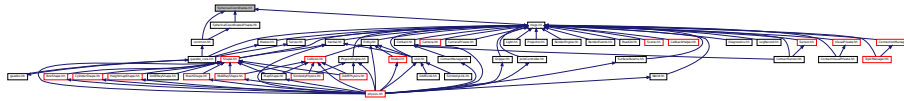
#include <string>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for SphericalCoordinates.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::SphericalCoordinates**

Convert spherical coordinates for planetary surfaces.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

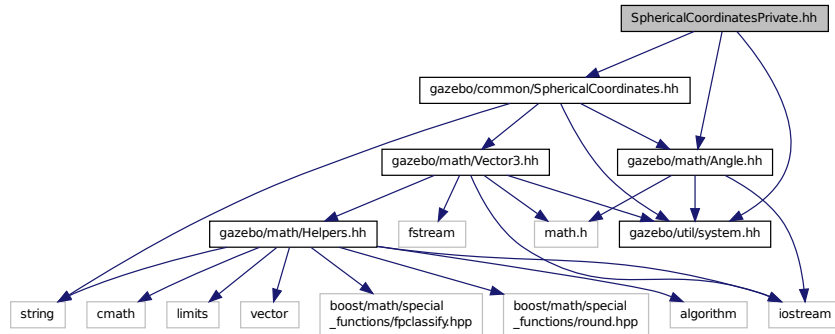
Common namespace.

11.216 SphericalCoordinatesPrivate.hh File Reference

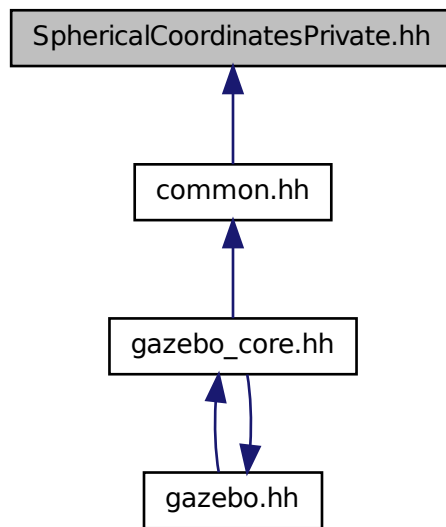
```

#include "gazebo/common/SphericalCoordinates.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/util/system.hh"
  
```

Include dependency graph for SphericalCoordinatesPrivate.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::SphericalCoordinatesPrivate**
common/common.hh

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

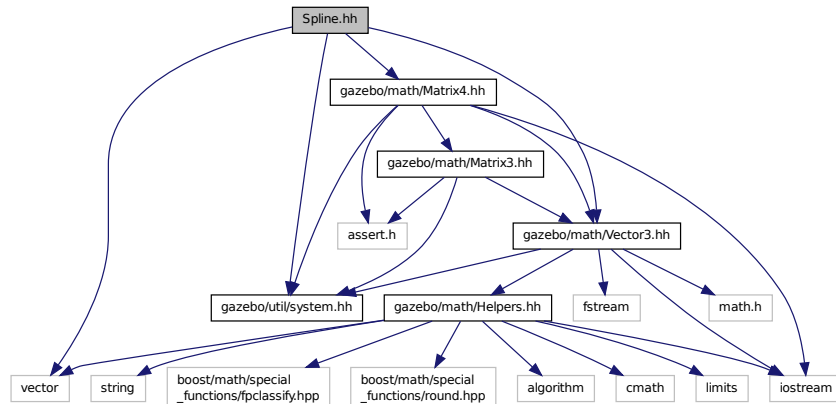
- namespace **gazebo::common**

Common namespace.

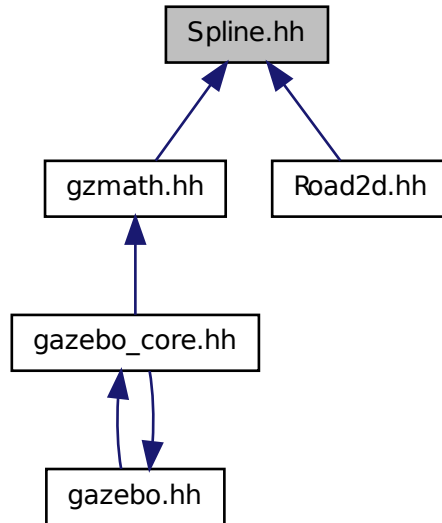
11.217 Spline.hh File Reference

```
#include <vector>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Matrix4.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Spline.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Spline**

Splines.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

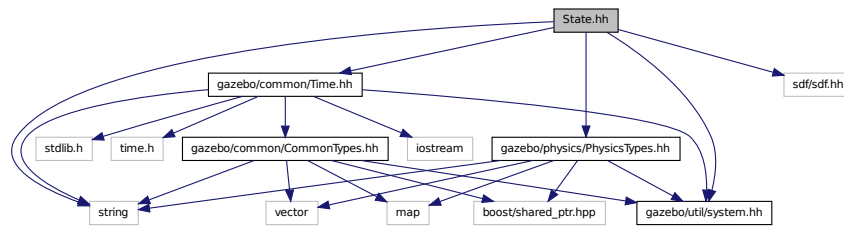
- namespace **gazebo::math**

Math namespace.

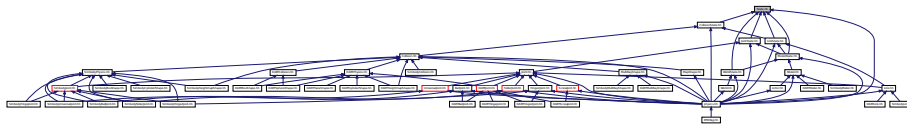
11.218 State.hh File Reference

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/util/system.hh"
```


Include dependency graph for State.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::State**

State (p. 1068) of an entity.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

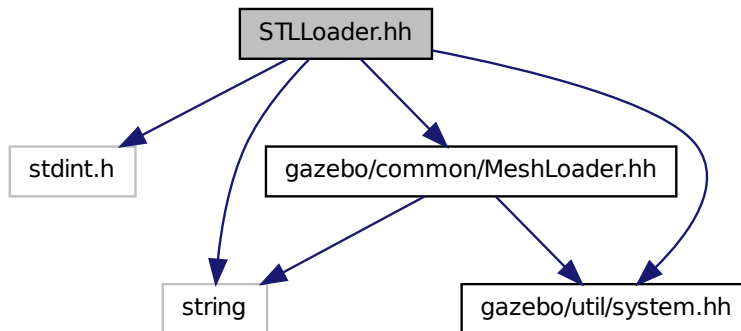
- namespace **gazebo::physics**

namespace for physics

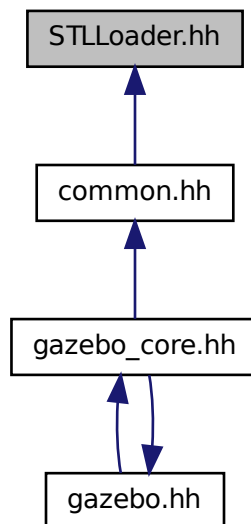
11.219 STLLoader.hh File Reference

```
#include <stdint.h>
#include <string>
#include "gazebo/common/MeshLoader.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for STLLoader.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::STLLoader**

Class used to load STL mesh files.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

Common namespace.

Macros

- `#define COR3_MAX 200000`
- `#define FACE_MAX 200000`
- `#define LINE_MAX_LEN 256`
- `#define ORDER_MAX 10`

11.219.1 Macro Definition Documentation

11.219.1.1 `#define COR3_MAX 200000`

11.219.1.2 `#define FACE_MAX 200000`

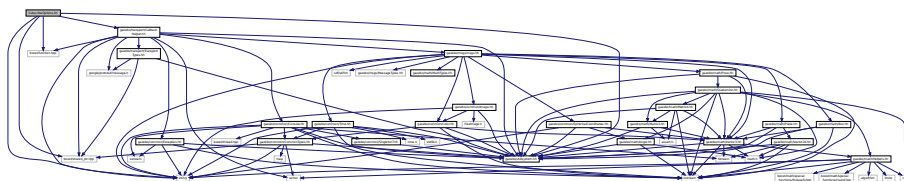
11.219.1.3 `#define LINE_MAX_LEN 256`

11.219.1.4 `#define ORDER_MAX 10`

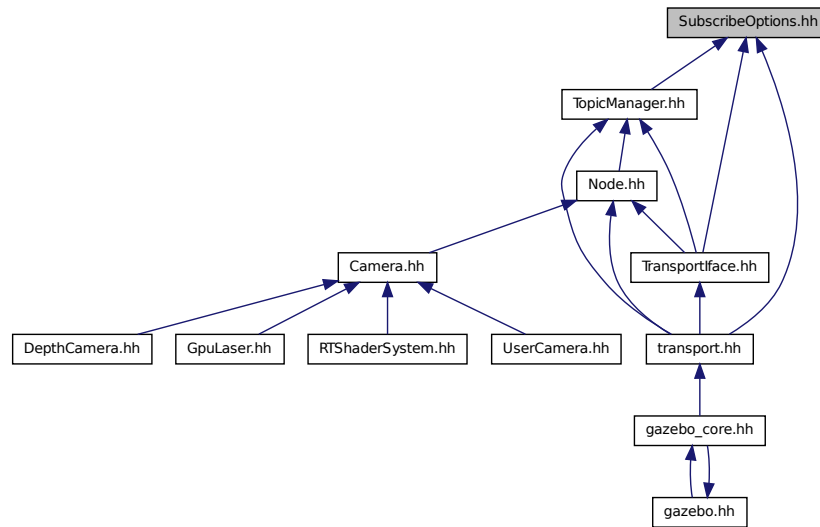
11.220 SubscribeOptions.hh File Reference

```
#include <boost/function.hpp>
#include <boost/shared_ptr.hpp>
#include <string>
#include "gazebo/transport/CallbackHelper.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SubscribeOptions.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::SubscribeOptions**
Options for a subscription.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

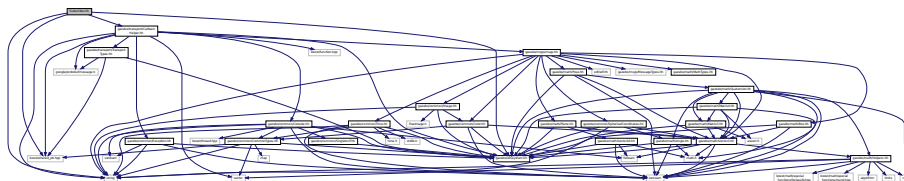
11.221 Subscriber.hh File Reference

```

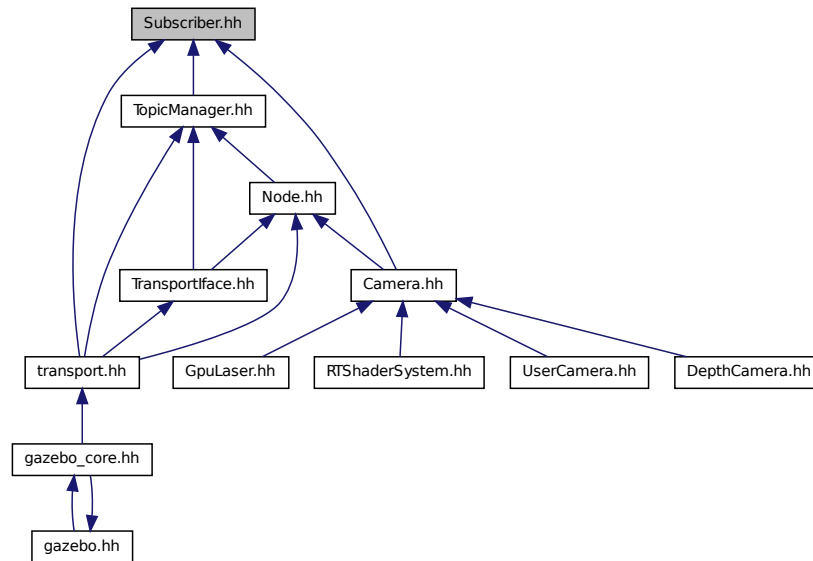
#include <string>
#include <boost/shared_ptr.hpp>
#include "gazebo/transport/CallbackHelper.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for Subscriber.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Subscriber**

A subscriber to a topic.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::transport**

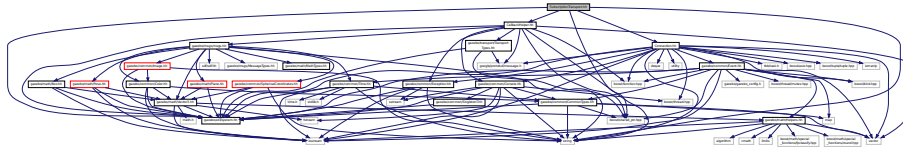
11.222 SubscriptionTransport.hh File Reference

```

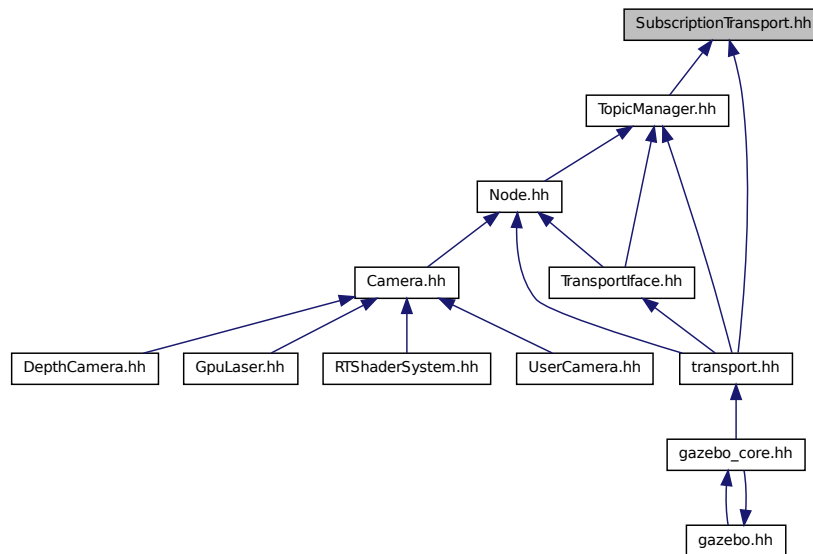
#include <boost/shared_ptr.hpp>
#include <string>
#include "Connection.hh"
#include "CallbackHelper.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for SubscriptionTransport.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::SubscriptionTransport**
transport/transport.hh

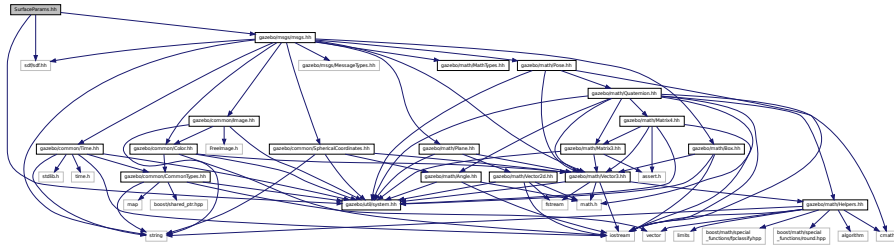
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

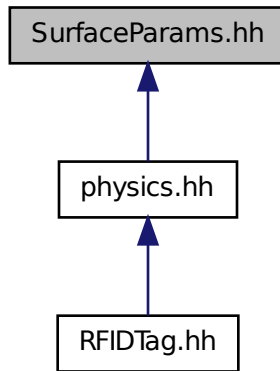
11.223 SurfaceParams.hh File Reference

```
#include <sdf/sdf.hh>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for SurfaceParams.hh:



This graph shows which files directly or indirectly include this file:



Classes

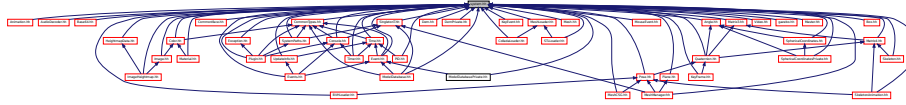
- class **gazebo::physics::FrictionPyramid**
Parameters used for friction pyramid model.
- class **gazebo::physics::SurfaceParams**
***SurfaceParams** (p. 1090) defines various Surface contact parameters.*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.224 system.hh File Reference

This graph shows which files directly or indirectly include this file:



Macros

- `#define GAZEBO_HIDDEN`
Use to represent "symbol hidden" if supported.
- `#define GAZEBO_VISIBLE`
Use to represent "symbol visible" if supported.

11.224.1 Macro Definition Documentation

11.224.1.1 `#define GAZEBO_HIDDEN`

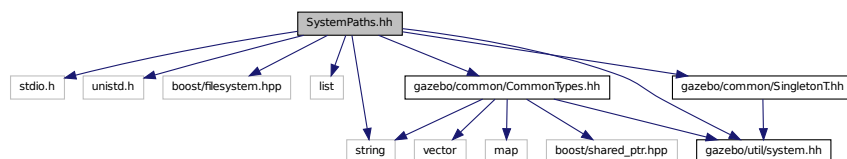
Use to represent "symbol hidden" if supported.

11.224.1.2 `#define GAZEBO_VISIBLE`

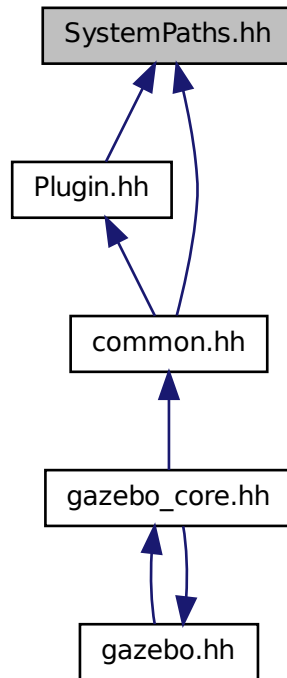
Use to represent "symbol visible" if supported.

11.225 SystemPaths.hh File Reference

```
#include <stdio.h>
#include <unistd.h>
#include <boost/filesystem.hpp>
#include <list>
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/util/system.hh"
Include dependency graph for SystemPaths.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::SystemPaths**

Functions to handle getting system paths, keeps track of:

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

Macros

- #define **GetCurrentDir** getcwd
- #define **LINUX**

11.225.1 Macro Definition Documentation

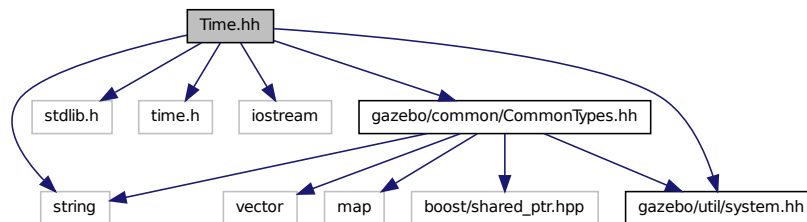
11.225.1.1 `#define GetCurrentDir getcwd`

11.225.1.2 `#define LINUX`

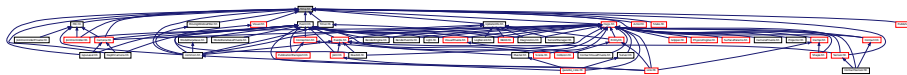
11.226 Time.hh File Reference

```
#include <string>
#include <stdlib.h>
#include <time.h>
#include <iostream>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Time.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Time**
A **Time** (p. 1099) class, can be used to hold wall- or sim-time.

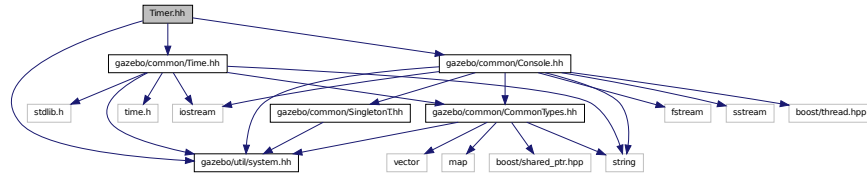
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

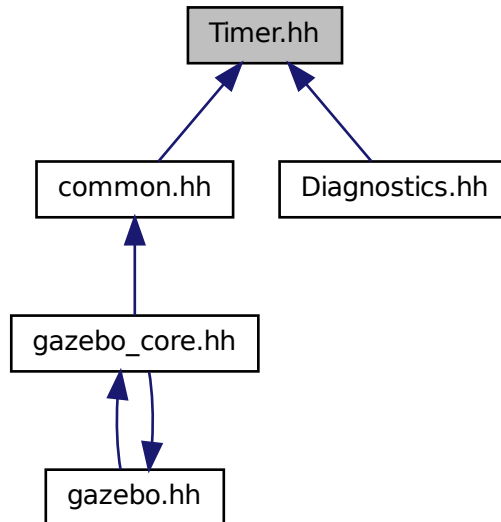
11.227 Timer.hh File Reference

```
#include "gazebo/common/Console.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Timer.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Timer**
A timer class, used to time things in real world walltime.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

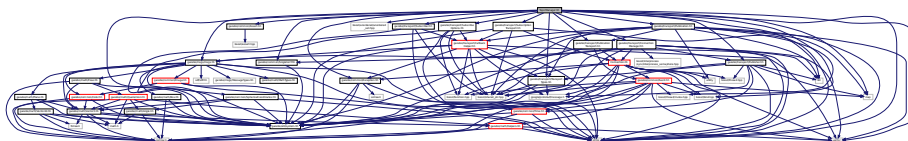
- namespace **gazebo::common**

Common namespace.

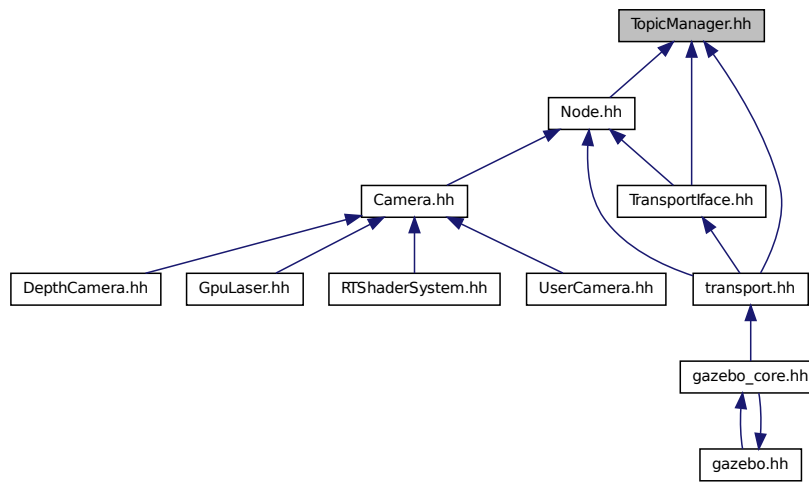
11.228 TopicManager.hh File Reference

```
#include <boost/bind.hpp>
#include <map>
#include <list>
#include <string>
#include <vector>
#include <boost/unordered/unordered_set.hpp>
#include "gazebo/common/Assert.hh"
#include "gazebo/common/Exception.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/transport/SubscribeOptions.hh"
#include "gazebo/transport/SubscriptionTransport.hh"
#include "gazebo/transport/PublicationTransport.hh"
#include "gazebo/transport/ConnectionManager.hh"
#include "gazebo/transport/Publisher.hh"
#include "gazebo/transport/Publication.hh"
#include "gazebo/transport/Subscriber.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for TopicManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::TopicManager**
Manages topics and their subscriptions.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

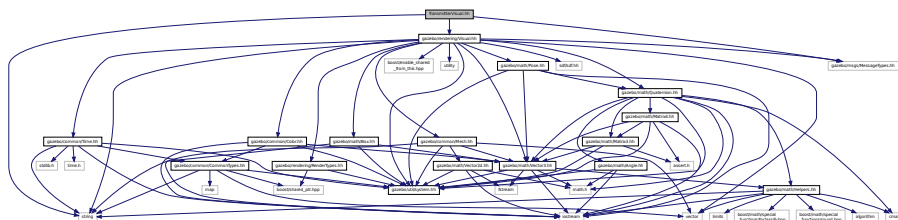
11.229 TransmitterVisual.hh File Reference

```

#include <string>
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/Visual.hh"

```

Include dependency graph for TransmitterVisual.hh:



Classes

- class **gazebo::rendering::TransmitterVisual**
Visualization for the wireless propagation data.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.230 TransmitterVisualPrivate.hh File Reference

```
#include <vector>
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/VisualPrivate.hh"
```

Include dependency graph for TransmitterVisualPrivate.hh:



Classes

- class **gazebo::rendering::TransmitterVisualPrivate**
*Private data for the Transmitter **Visual** (p. 1196) class.*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.231 Transportface.hh File Reference

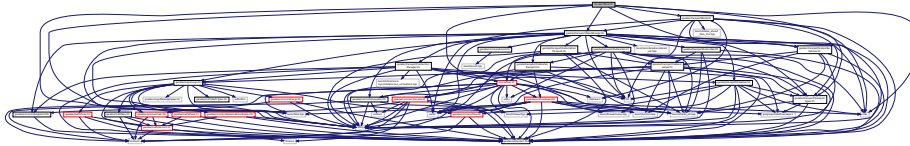
```
#include <boost/bind.hpp>
```

```

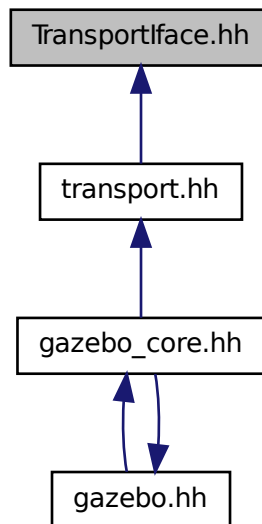
#include <string>
#include <list>
#include <map>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/transport/SubscribeOptions.hh"
#include "gazebo/transport/Node.hh"
#include "gazebo/transport/TopicManager.hh"

```

Include dependency graph for Transportface.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

Functions

- **GAZEBO_VISIBLE** void **gazebo::transport::clear_buffers** ()

Clear any remaining communication buffers.

- **GAZEBO_VISIBLE**
 transport::ConnectionPtr **gazebo::transport::connectToMaster** ()
Create a connection to master.
- **GAZEBO_VISIBLE** void **gazebo::transport::fini** ()
Cleanup the transport component.
- **GAZEBO_VISIBLE** bool **gazebo::transport::get_master_uri** (std::string &_master_host, unsigned int &_master_port)
Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.
- **GAZEBO_VISIBLE** void **gazebo::transport::get_topic_namespaces** (std::list< std::string > &_namespaces)
Return all the namespace (world names) on the master.
- **GAZEBO_VISIBLE** std::map
 < std::string, std::list
 < std::string > > **gazebo::transport::getAdvertisedTopics** ()
Get a list of all the topics and their message types.
- **GAZEBO_VISIBLE** std::list
 < std::string > **gazebo::transport::getAdvertisedTopics** (const std::string &_msgType)
Get a list of all the unique advertised topic names.
- **GAZEBO_VISIBLE** bool **gazebo::transport::getMinimalComms** ()
Get whether minimal comms has been enabled.
- **GAZEBO_VISIBLE** std::string **gazebo::transport::getTopicMsgType** (const std::string &_topicName)
Get the message typename that is published on the given topic.
- **GAZEBO_VISIBLE** bool **gazebo::transport::init** (const std::string &_masterHost="", unsigned int _masterPort=0, uint32_t _timeoutIterations=30)
Initialize the transport system.
- bool **gazebo::transport::is_stopped** ()
Is the transport system stopped?
- **GAZEBO_VISIBLE** void **gazebo::transport::pause_incoming** (bool _pause)
Pause or unpause incoming messages.
- template<typename M >
GAZEBO_VISIBLE void **gazebo::transport::publish** (const std::string &_topic, const google::protobuf::Message &_message)
A convenience function for a one-time publication of a message.
- **GAZEBO_VISIBLE**
 boost::shared_ptr
 < msgs::Response > **gazebo::transport::request** (const std::string &_worldName, const std::string &_request, const std::string &_data="")
Send a request and receive a response.
- **GAZEBO_VISIBLE** void **gazebo::transport::requestNoReply** (const std::string &_worldName, const std::string &_request, const std::string &_data="")
Send a request and don't wait for a response.
- **GAZEBO_VISIBLE** void **gazebo::transport::requestNoReply** (NodePtr _node, const std::string &_request, const std::string &_data="")
Send a request and don't wait for a response.
- **GAZEBO_VISIBLE** void **gazebo::transport::run** ()
Run the transport component.
- **GAZEBO_VISIBLE** void **gazebo::transport::setMinimalComms** (bool _enabled)
Set whether minimal comms should be used.
- **GAZEBO_VISIBLE** void **gazebo::transport::stop** ()

Stop the transport component from running.

- **GAZEBO_VISIBLE** bool **gazebo::transport::waitForNamespaces** (const **gazebo::common::Time** &_maxWait)

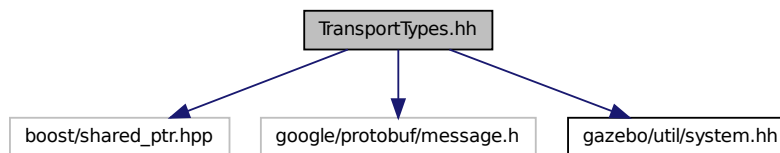
*Blocks while waiting for topic namespaces from the **Master** (p. 637).*

11.232 TransportTypes.hh File Reference

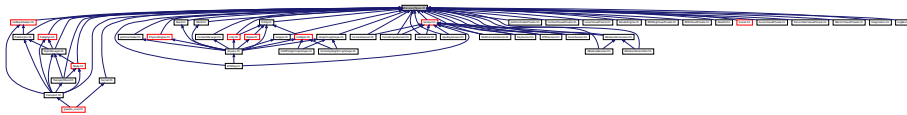
Forward declarations for transport.

```
#include <boost/shared_ptr.hpp>
#include <google/protobuf/message.h>
#include "gazebo/util/system.hh"
```

Include dependency graph for TransportTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

Typedefs

- typedef boost::shared_ptr
< google::protobuf::Message > **gazebo::transport::MessagePtr**
- typedef boost::shared_ptr< Node > **gazebo::transport::NodePtr**
- typedef boost::shared_ptr
< Publication > **gazebo::transport::PublicationPtr**
- typedef boost::shared_ptr
< PublicationTransport > **gazebo::transport::PublicationTransportPtr**
- typedef boost::shared_ptr
< Publisher > **gazebo::transport::PublisherPtr**

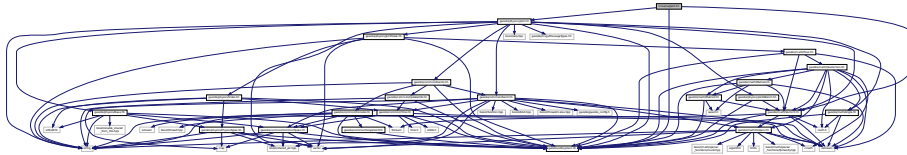
- typedef boost::shared_ptr
< Subscriber > **gazebo::transport::SubscriberPtr**
- typedef boost::shared_ptr
< SubscriptionTransport > **gazebo::transport::SubscriptionTransportPtr**

11.232.1 Detailed Description

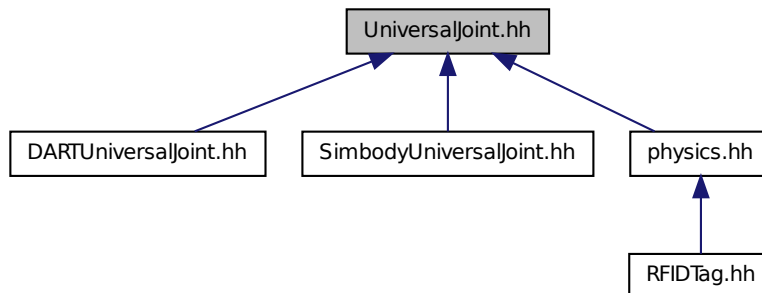
Forward declarations for transport.

11.233 UniversalJoint.hh File Reference

```
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/Joint.hh"
#include "gazebo/util/system.hh"
Include dependency graph for UniversalJoint.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::UniversalJoint**< T >
A universal joint.

Namespaces

- namespace **gazebo**

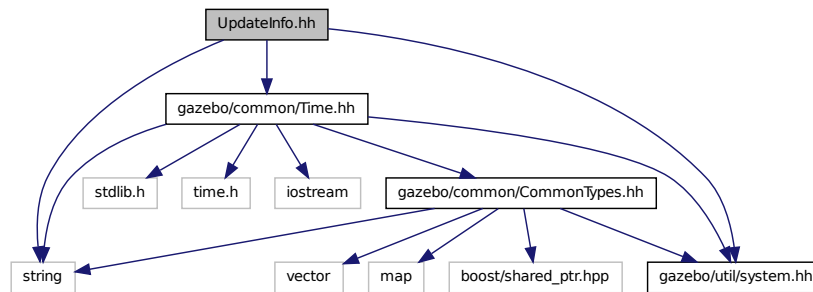
Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

11.234 UpdateInfo.hh File Reference

```
#include <string>
#include "gazebo/common/Time.hh"
#include "gazebo/util/system.hh"
Include dependency graph for UpdateInfo.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::UpdateInfo**

Information for use in an update event.

Namespaces

- namespace **gazebo**

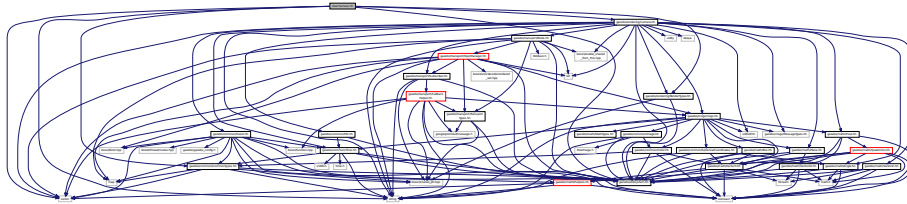
Forward declarations for the common classes.

- namespace **gazebo::common**

Common namespace.

11.235 UserCamera.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Camera.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/util/system.hh"
Include dependency graph for UserCamera.hh:
```



Classes

- class **gazebo::rendering::UserCamera**
A camera used for user visualization of a scene.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.236 UserCameraPrivate.hh File Reference

Classes

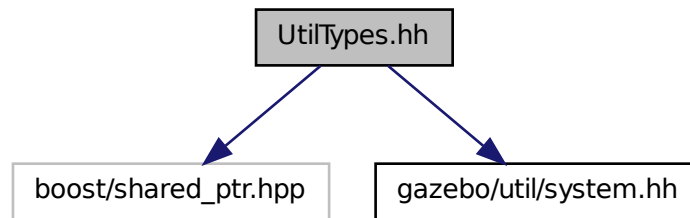
- class **gazebo::rendering::UserCameraPrivate**
*Private data for the **UserCamera** (p. 1137) class.*

Namespaces

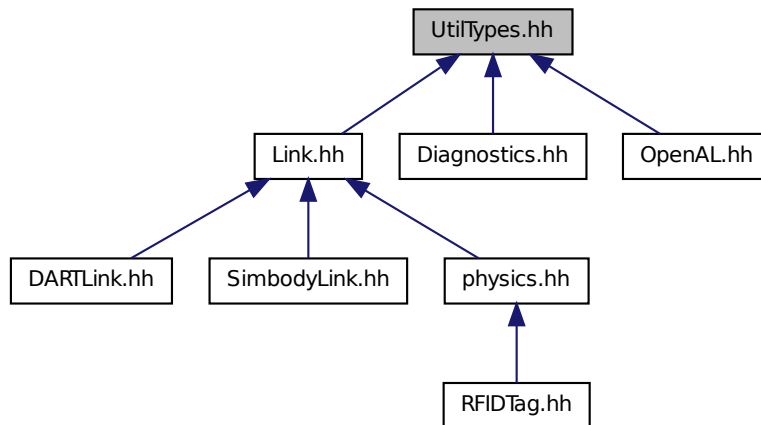
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.237 UtilTypes.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include "gazebo/util/system.hh"
Include dependency graph for UtilTypes.hh:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::util**

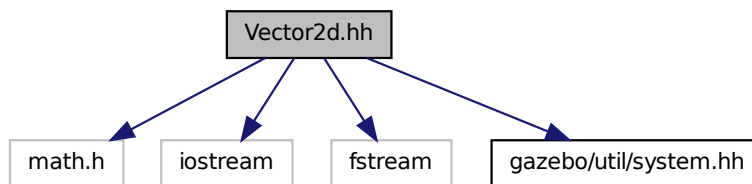
Typedefs

- typedef boost::shared_ptr
< DiagnosticTimer > **gazebo::util::DiagnosticTimerPtr**
- typedef boost::shared_ptr
< OpenALSink > **gazebo::util::OpenALSinkPtr**
- typedef boost::shared_ptr
< OpenALSource > **gazebo::util::OpenALSourcePtr**

11.237.1 Detailed Description

11.238 Vector2d.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
#include "gazebo/util/system.hh"
Include dependency graph for Vector2d.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Vector2d**
Generic double x, y vector.

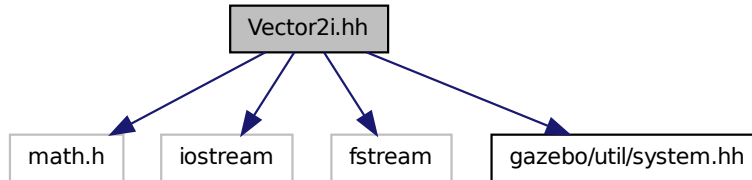
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

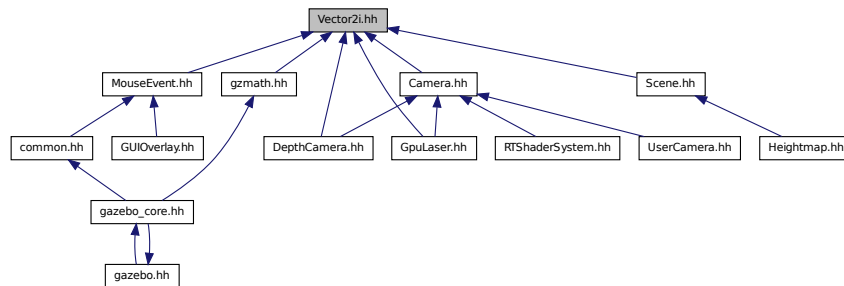
11.239 Vector2i.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
#include "gazebo/util/system.hh"
```

Include dependency graph for Vector2i.hh:



This graph shows which files directly or indirectly include this file:



Classes

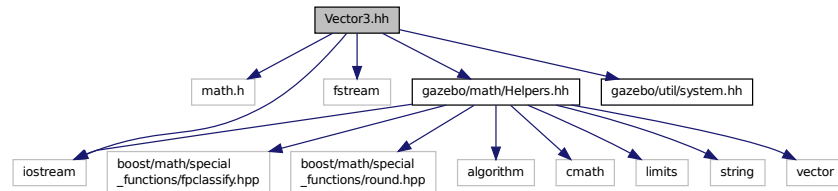
- class **gazebo::math::Vector2i**
Generic integer x, y vector.

Namespaces

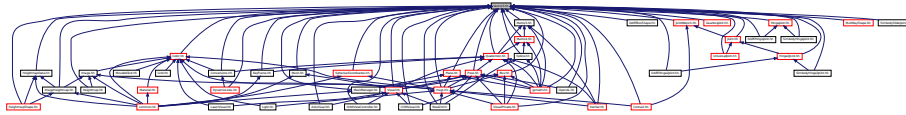
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

11.240 Vector3.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
#include "gazebo/math/Helpers.hh"
#include "gazebo/util/system.hh"
Include dependency graph for Vector3.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Vector3**

The **Vector3** (p. 1165) class represents the generic vector containing 3 elements.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

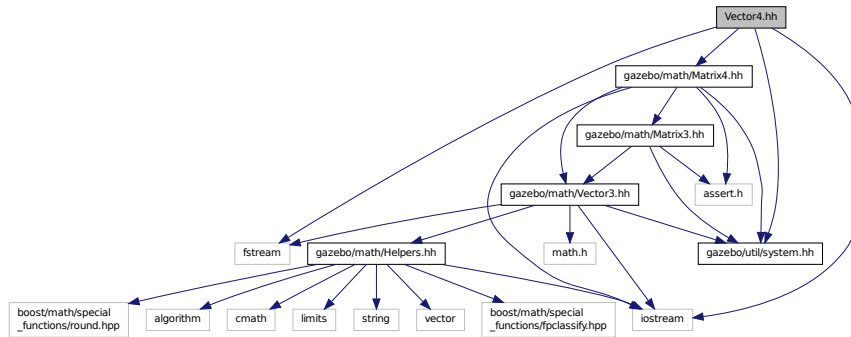
- namespace **gazebo::math**

Math namespace.

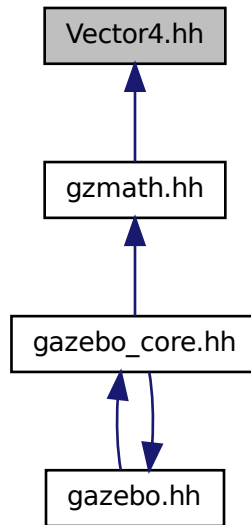
11.241 Vector4.hh File Reference

```
#include <iostream>
#include <fstream>
#include "gazebo/math/Matrix4.hh"
#include "gazebo/util/system.hh"
```


Include dependency graph for Vector4.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Vector4**
double Generic x, y, z, w vector

Namespaces

- namespace **gazebo**

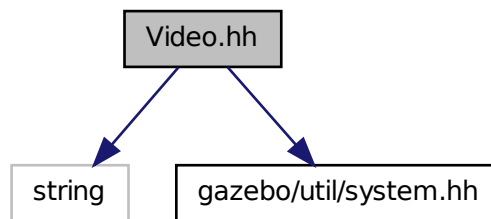
Forward declarations for the common classes.

- namespace **gazebo::math**

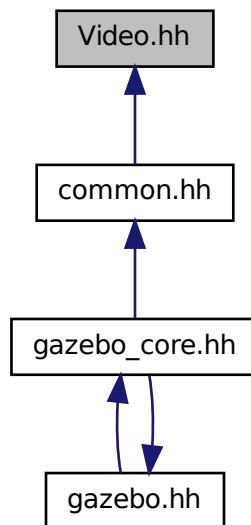
Math namespace.

11.242 Video.hh File Reference

```
#include <string>
#include "gazebo/util/system.hh"
Include dependency graph for Video.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

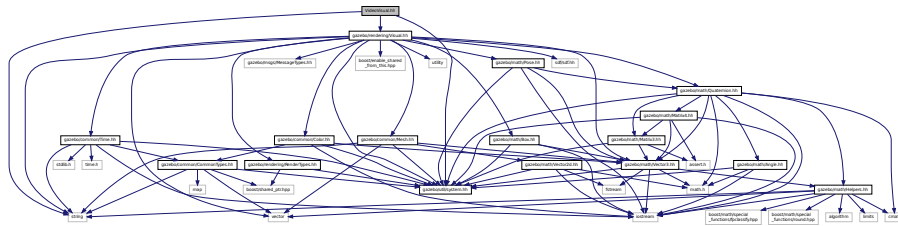
- class **gazebo::common::Video**
Handle video encoding and decoding using libavcodec.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.243 VideoVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/util/system.hh"
Include dependency graph for VideoVisual.hh:
```



Classes

- class **gazebo::rendering::VideoVisual**
A visual element that displays a video as a texture.

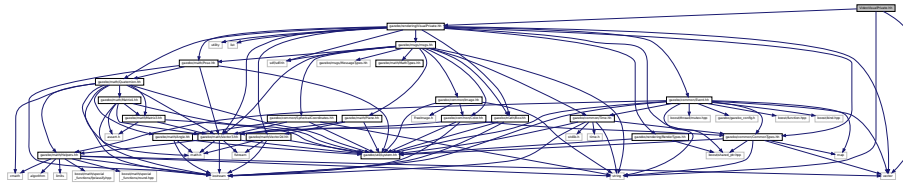
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.244 VideoVisualPrivate.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/VisualPrivate.hh"
```

Include dependency graph for VideoVisualPrivate.hh:



Classes

- class **gazebo::rendering::VideoVisualPrivate**
*Private data for the Video **Visual** (p. 1196) class.*

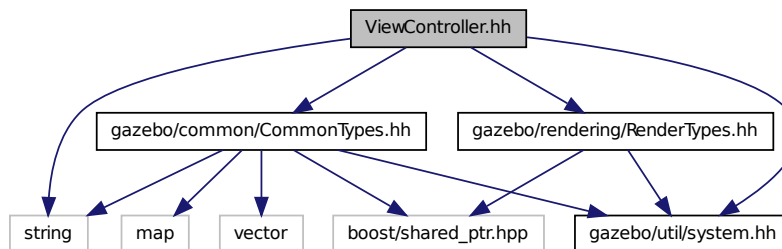
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::rendering**
Rendering namespace.

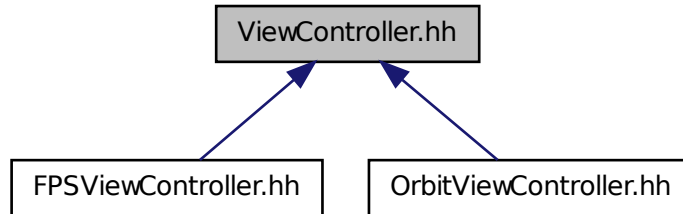
11.245 ViewController.hh File Reference

```
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for ViewController.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::ViewController**

Base class for view controllers.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

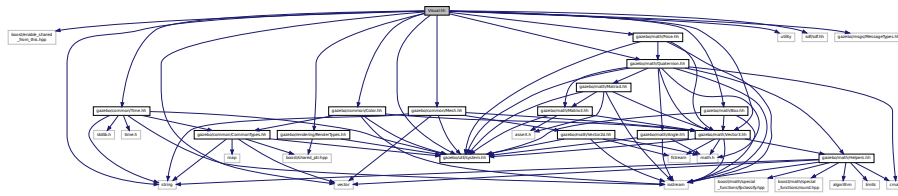
- namespace **gazebo::rendering**

Rendering namespace.

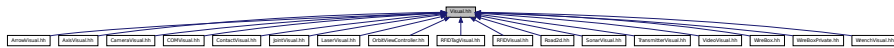
11.246 Visual.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include <utility>
#include <vector>
#include <sdf/sdf.hh>
#include "gazebo/common/Color.hh"
#include "gazebo/common/Mesh.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/messages/MessageTypes.hh"
#include "gazebo/math/Box.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Quaternion.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for Visual.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::Visual**

A renderable object.

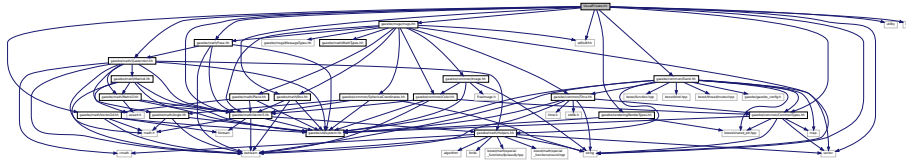
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

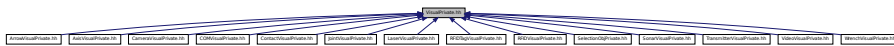
11.247 VisualPrivate.hh File Reference

```
#include <string>
#include <utility>
#include <list>
#include <vector>
#include <sdf/sdf.hh>
#include "gazebo_msgs_msgs.hh"
#include "gazebo_common/Event.hh"
#include "gazebo_math/Box.hh"
#include "gazebo_math/Pose.hh"
#include "gazebo_math/Quaternion.hh"
#include "gazebo_math/Vector3.hh"
#include "gazebo_math/Vector2d.hh"
#include "gazebo_rendering/RenderTypes.hh"
#include "gazebo_common/CommonTypes.hh"
```

Include dependency graph for VisualPrivate.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::VisualPrivate**

*Private data for the **Visual** (p. 1196) class.*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

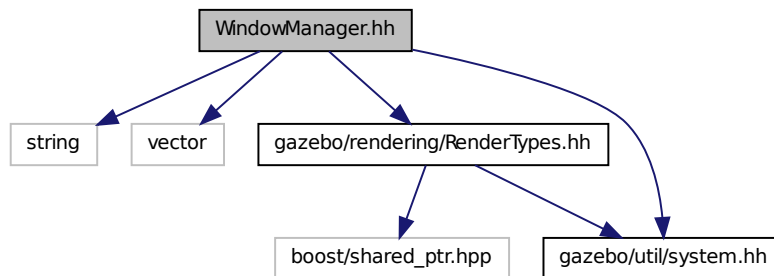
Rendering namespace.

- namespace **Ogre**

11.248 WindowManager.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for WindowManager.hh:



Classes

- class **gazebo::rendering::WindowManager**
Class to manage render windows.

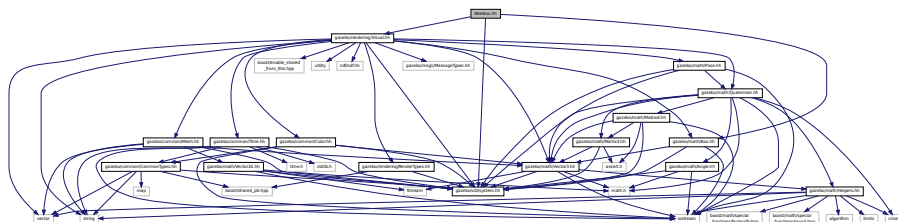
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.249 WireBox.hh File Reference

```
#include "gazebo/math/Box.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for WireBox.hh:



Classes

- class **gazebo::rendering::WireBox**

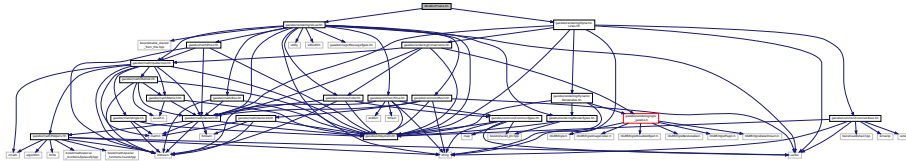
Draws a wireframe box.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.250 WireBoxPrivate.hh File Reference

```
#include "gazebo/rendering/Visual.hh"
#include "gazebo/rendering/DynamicLines.hh"
Include dependency graph for WireBoxPrivate.hh:
```



Classes

- class **gazebo::rendering::WireBoxPrivate**

*Private data for the **WireBox** (p. 1227) class.*

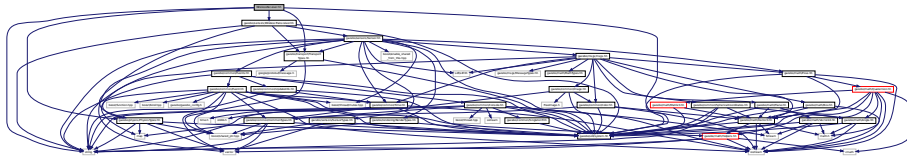
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.251 WirelessReceiver.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/WirelessTransceiver.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for WirelessReceiver.hh:



Classes

- class **gazebo::sensors::WirelessReceiver**

Sensor (p. 907) class for receiving wireless signals.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

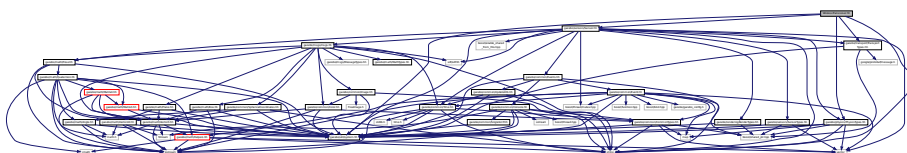
- namespace **gazebo::sensors**

Sensors namespace.

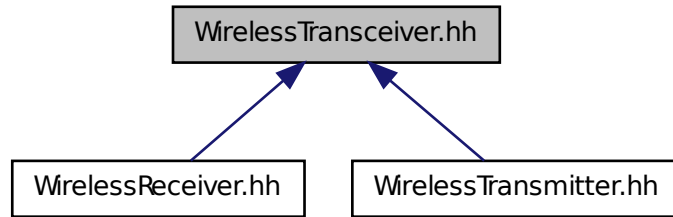
11.252 WirelessTransceiver.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for WirelessTransceiver.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::sensors::WirelessTransceiver**
Sensor (p. 907) class for receiving wireless signals.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

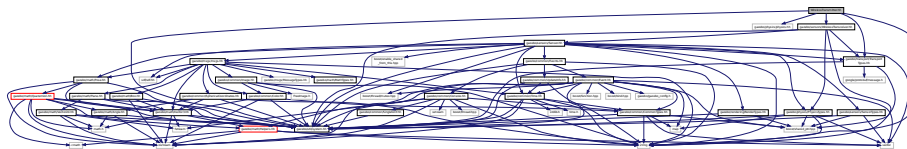
11.253 WirelessTransmitter.hh File Reference

```

#include <string>
#include "gazebo/physics/physics.hh"
#include "gazebo/sensors/WirelessTransceiver.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/util/system.hh"

```

Include dependency graph for WirelessTransmitter.hh:



Classes

- class **gazebo::sensors::WirelessTransmitter**
Transmitter to send wireless signals.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::sensors**

Sensors namespace.

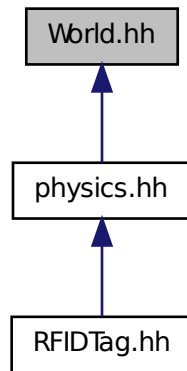
11.254 World.hh File Reference

```
#include <vector>
#include <list>
#include <set>
#include <deque>
#include <string>
#include <boost/thread.hpp>
#include <boost/enable_shared_from_this.hpp>
#include <boost/shared_ptr.hpp>
#include <sdf/sdf.hh>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/physics/Base.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/WorldState.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for World.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::World**

The world provides access to all other object within a simulated environment.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

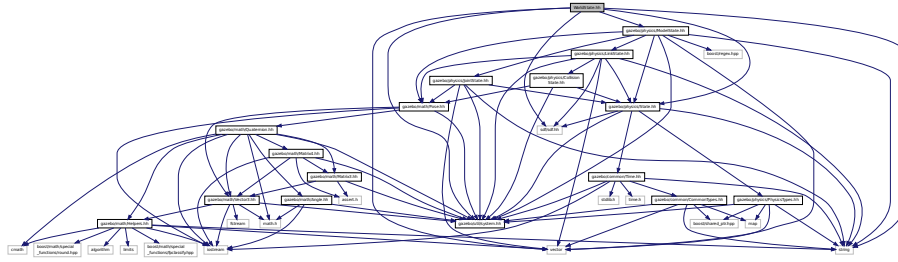
- namespace **gazebo::physics**

namespace for physics

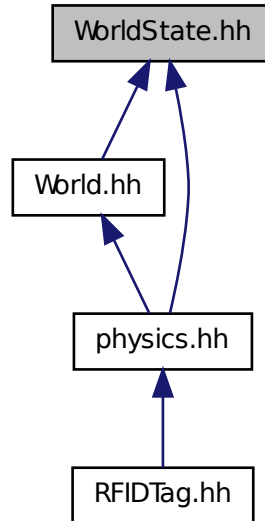
11.255 WorldState.hh File Reference

```
#include <string>
#include <vector>
#include <sdf/sdf.hh>
#include "gazebo/physics/State.hh"
#include "gazebo/physics/ModelState.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for WorldState.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::WorldState**
*Store state information of a **physics::World** (p. 1239) object.*

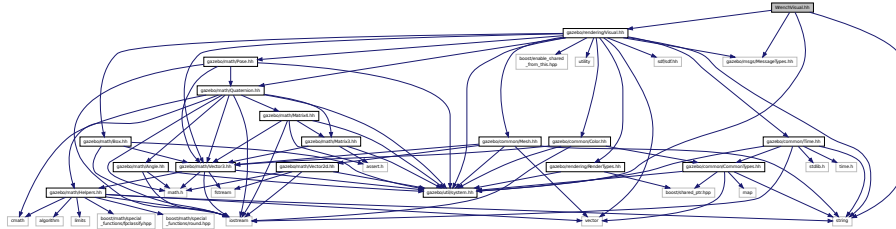
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.256 WrenchVisual.hh File Reference

```
#include <string>
#include "gazebo_msgs/MessageTypes.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/util/system.hh"
```

Include dependency graph for WrenchVisual.hh:



Classes

- class **gazebo::rendering::WrenchVisual**
Visualization for sonar data.

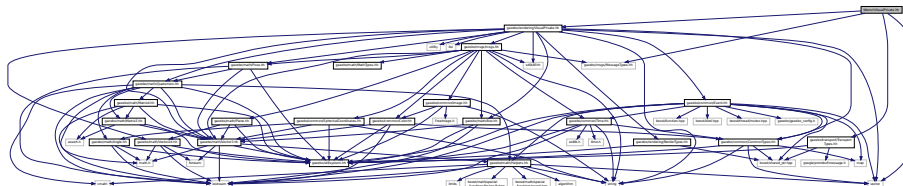
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.257 WrenchVisualPrivate.hh File Reference

```
#include <vector>
#include "gazebo_msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/VisualPrivate.hh"
```

Include dependency graph for WrenchVisualPrivate.hh:



Classes

- class **gazebo::rendering::WrenchVisualPrivate**

*Private data for the Wrench **Visual** (p. 1196) class.*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

Index

- ~Actor
 - gazebo::physics::Actor, 141
- ~Angle
 - gazebo::math::Angle, 147
- ~Animation
 - gazebo::common::Animation, 155
- ~ArrowVisual
 - gazebo::rendering::ArrowVisual, 158
- ~AssertionInternalError
 - gazebo::common::AssertionInternalError, 161
- ~AudioDecoder
 - gazebo::common::AudioDecoder, 162
- ~AxisVisual
 - gazebo::rendering::AxisVisual, 164
- ~BVHLoader
 - gazebo::common::BVHLoader, 191
- ~BallJoint
 - gazebo::physics::BallJoint, 167
- ~Base
 - gazebo::physics::Base, 173
- ~Box
 - gazebo::math::Box, 182
- ~BoxShape
 - gazebo::physics::BoxShape, 187
- ~Buffer
 - gazebo::common::FileLogger::Buffer, 190
 - gazebo::common::Logger::Buffer, 189
- ~COMVisual
 - gazebo::rendering::COMVisual, 261
- ~CallbackHelper
 - gazebo::transport::CallbackHelper, 193
- ~Camera
 - gazebo::rendering::Camera, 204
- ~CameraSensor
 - gazebo::sensors::CameraSensor, 229
- ~CameraVisual
 - gazebo::rendering::CameraVisual, 232
- ~ColladaLoader
 - gazebo::common::ColladaLoader, 235
- ~Collision
 - gazebo::physics::Collision, 238
- ~CollisionState
 - gazebo::physics::CollisionState, 247
- ~Color
 - gazebo::common::Color, 252
- ~Connection
 - gazebo::event::Connection, 264
 - gazebo::transport::Connection, 266
- ~Contact
 - gazebo::physics::Contact, 280
- ~ContactManager
 - gazebo::physics::ContactManager, 283
- ~ContactSensor
 - gazebo::sensors::ContactSensor, 289
- ~ContactVisual
 - gazebo::rendering::ContactVisual, 293
- ~CylinderShape
 - gazebo::physics::CylinderShape, 299
- ~DARTBallJoint
 - gazebo::physics::DARTBallJoint, 303
- ~DARTBoxShape
 - gazebo::physics::DARTBoxShape, 309
- ~DARTCollision
 - gazebo::physics::DARTCollision, 311
- ~DARTCylinderShape
 - gazebo::physics::DARTCylinderShape, 315
- ~DARTHeightmapShape
 - gazebo::physics::DARTHeightmapShape, 317
- ~DARTHinge2Joint
 - gazebo::physics::DARTHinge2Joint, 319
- ~DARTHingeJoint
 - gazebo::physics::DARTHingeJoint, 324
- ~DARTJoint
 - gazebo::physics::DARTJoint, 330
- ~DARTLink
 - gazebo::physics::DARTLink, 340
- ~DARTMeshShape
 - gazebo::physics::DARTMeshShape, 349
- ~DARTModel
 - gazebo::physics::DARTModel, 351
- ~DARTMultiRayShape
 - Bullet Physics, 79
- ~DARTPhysics
 - gazebo::physics::DARTPhysics, 356
- ~DARTPlaneShape
 - gazebo::physics::DARTPlaneShape, 362
- ~DARTRayShape
 - DART Physics, 77
- ~DARTScrewJoint
 - gazebo::physics::DARTScrewJoint, 366

- ~DARTSliderJoint
 - gazebo::physics::DARTSliderJoint, 373
- ~DARTSphereShape
 - gazebo::physics::DARTSphereShape, 377
- ~DARTUniversalJoint
 - gazebo::physics::DARTUniversalJoint, 380
- ~DepthCamera
 - gazebo::rendering::DepthCamera, 384
- ~DepthCameraSensor
 - gazebo::sensors::DepthCameraSensor, 388
- ~DiagnosticTimer
 - gazebo::util::DiagnosticTimer, 395
- ~DynamicLines
 - gazebo::rendering::DynamicLines, 398
- ~DynamicRenderable
 - gazebo::rendering::DynamicRenderable, 401
- ~Entity
 - gazebo::physics::Entity, 407
- ~Event
 - gazebo::event::Event, 417
- ~EventT
 - Events, 47
- ~Exception
 - gazebo::common::Exception, 446
- ~FPSViewController
 - gazebo::rendering::FPSViewController, 454
- ~FileLogger
 - gazebo::common::FileLogger, 448
- ~ForceTorqueSensor
 - gazebo::sensors::ForceTorqueSensor, 450
- ~FrictionPyramid
 - gazebo::physics::FrictionPyramid, 456
- ~GUIOverlay
 - gazebo::rendering::GUIOverlay, 495
- ~GaussianNoiseModel
 - gazebo::sensors::GaussianNoiseModel, 459
- ~GazeboGenerator
 - google::protobuf::compiler::cpp::GazeboGenerator, 461
- ~GearboxJoint
 - gazebo::physics::GearboxJoint, 463
- ~GpsSensor
 - gazebo::sensors::GpsSensor, 465
- ~GpuLaser
 - gazebo::rendering::GpuLaser, 470
- ~GpuRaySensor
 - gazebo::sensors::GpuRaySensor, 479
- ~Grid
 - gazebo::rendering::Grid, 489
- ~Gripper
 - gazebo::physics::Gripper, 493
- ~GzTerrainMatGen
 - gazebo::rendering::GzTerrainMatGen, 499
- ~Heightmap
 - gazebo::rendering::Heightmap, 501
- ~HeightmapData
 - gazebo::common::HeightmapData, 505
- ~HeightmapShape
 - gazebo::physics::HeightmapShape, 508
- ~Hinge2Joint
 - gazebo::physics::Hinge2Joint, 512
- ~HingeJoint
 - gazebo::physics::HingeJoint, 514
- ~IOManager
 - gazebo::transport::IOManager, 540
- ~Image
 - gazebo::common::Image, 517
- ~ImageGaussianNoiseModel
 - gazebo::sensors::ImageGaussianNoiseModel, 522
- ~ImuSensor
 - gazebo::sensors::ImuSensor, 527
- ~Inertial
 - gazebo::physics::Inertial, 531
- ~InternalError
 - gazebo::common::InternalError, 540
- ~Joint
 - gazebo::physics::Joint, 547
- ~JointController
 - gazebo::physics::JointController, 569
- ~JointState
 - gazebo::physics::JointState, 576
- ~JointVisual
 - gazebo::rendering::JointVisual, 579
- ~KeyFrame
 - gazebo::common::KeyFrame, 585
- ~LaserVisual
 - gazebo::rendering::LaserVisual, 587
- ~Light
 - gazebo::rendering::Light, 591
- ~Link
 - gazebo::physics::Link, 601
- ~LinkState
 - gazebo::physics::LinkState, 620
- ~Logger
 - gazebo::common::Logger, 627
- ~Master
 - gazebo::Master, 638
- ~Material
 - gazebo::common::Material, 642
- ~Matrix3
 - gazebo::math::Matrix3, 650
- ~Matrix4
 - gazebo::math::Matrix4, 656
- ~Mesh
 - gazebo::common::Mesh, 662
- ~MeshCSG
 - gazebo::common::MeshCSG, 668
- ~MeshLoader

- gazebo::common::MeshLoader, 669
- ~MeshShape
 - gazebo::physics::MeshShape, 676
- ~Model
 - gazebo::physics::Model, 682
- ~ModelPlugin
 - gazebo::ModelPlugin, 697
- ~ModelState
 - gazebo::physics::ModelState, 700
- ~MovableText
 - gazebo::rendering::MovableText, 711
- ~MovingWindowFilter
 - Common, 41
- ~MultiCameraSensor
 - gazebo::sensors::MultiCameraSensor, 720
- ~MultiRayShape
 - gazebo::physics::MultiRayShape, 726
- ~Node
 - gazebo::transport::Node, 733
- ~NodeAnimation
 - gazebo::common::NodeAnimation, 739
- ~NodeTransform
 - gazebo::common::NodeTransform, 745
- ~Noise
 - gazebo::sensors::Noise, 749
- ~NumericAnimation
 - gazebo::common::NumericAnimation, 753
- ~NumericKeyFrame
 - gazebo::common::NumericKeyFrame, 755
- ~OpenALSink
 - gazebo::util::OpenALSink, 758
- ~OpenALSource
 - gazebo::util::OpenALSource, 759
- ~OrbitViewController
 - gazebo::rendering::OrbitViewController, 764
- ~PID
 - gazebo::common::PID, 784
- ~PhysicsEngine
 - gazebo::physics::PhysicsEngine, 770
- ~Plane
 - gazebo::math::Plane, 789
- ~PlaneShape
 - gazebo::physics::PlaneShape, 792
- ~PluginT
 - gazebo::PluginT, 795
- ~Pose
 - gazebo::math::Pose, 799
- ~PoseAnimation
 - gazebo::common::PoseAnimation, 807
- ~PoseKeyFrame
 - gazebo::common::PoseKeyFrame, 809
- ~Projector
 - gazebo::rendering::Projector, 811
- ~Publication
 - gazebo::transport::Publication, 814
- ~PublicationTransport
 - gazebo::transport::PublicationTransport, 818
- ~Publisher
 - gazebo::transport::Publisher, 821
- ~Quaternion
 - gazebo::math::Quaternion, 828
- ~RFIDSensor
 - gazebo::sensors::RFIDSensor, 860
- ~RFIDTag
 - gazebo::sensors::RFIDTag, 863
- ~RFIDTagVisual
 - gazebo::rendering::RFIDTagVisual, 865
- ~RFIDVisual
 - gazebo::rendering::RFIDVisual, 867
- ~RaySensor
 - gazebo::sensors::RaySensor, 844
- ~RayShape
 - gazebo::physics::RayShape, 851
- ~Road
 - gazebo::physics::Road, 870
- ~Road2d
 - gazebo::rendering::Road2d, 871
- ~RotationSpline
 - gazebo::math::RotationSpline, 872
- ~SM2Profile
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 1046
- ~STLLoader
 - gazebo::common::STLLoader, 1073
- ~Scene
 - gazebo::rendering::Scene, 883
- ~ScrewJoint
 - gazebo::physics::ScrewJoint, 897
- ~SelectionObj
 - gazebo::rendering::SelectionObj, 901
- ~Sensor
 - gazebo::sensors::Sensor, 911
- ~SensorPlugin
 - gazebo::SensorPlugin, 925
- ~Server
 - gazebo::Server, 926
- ~Shape
 - gazebo::physics::Shape, 934
- ~SimTimeEventHandler
 - gazebo::sensors::SimTimeEventHandler, 1021
- ~SimbodyBallJoint
 - gazebo::physics::SimbodyBallJoint, 937
- ~SimbodyBoxShape
 - gazebo::physics::SimbodyBoxShape, 942
- ~SimbodyCollision
 - gazebo::physics::SimbodyCollision, 945
- ~SimbodyCylinderShape
 - gazebo::physics::SimbodyCylinderShape, 947

- ~SimbodyHeightmapShape
 - gazebo::physics::SimbodyHeightmapShape, 949
- ~SimbodyHinge2Joint
 - gazebo::physics::SimbodyHinge2Joint, 952
- ~SimbodyHingeJoint
 - gazebo::physics::SimbodyHingeJoint, 957
- ~SimbodyJoint
 - gazebo::physics::SimbodyJoint, 962
- ~SimbodyLink
 - gazebo::physics::SimbodyLink, 974
- ~SimbodyMeshShape
 - gazebo::physics::SimbodyMeshShape, 982
- ~SimbodyModel
 - gazebo::physics::SimbodyModel, 984
- ~SimbodyMultiRayShape
 - gazebo::physics::SimbodyMultiRayShape, 986
- ~SimbodyPhysics
 - gazebo::physics::SimbodyPhysics, 989
- ~SimbodyPlaneShape
 - gazebo::physics::SimbodyPlaneShape, 998
- ~SimbodyRayShape
 - gazebo::physics::SimbodyRayShape, 1001
- ~SimbodyScrewJoint
 - gazebo::physics::SimbodyScrewJoint, 1004
- ~SimbodySliderJoint
 - gazebo::physics::SimbodySliderJoint, 1011
- ~SimbodySphereShape
 - gazebo::physics::SimbodySphereShape, 1015
- ~SimbodyUniversalJoint
 - gazebo::physics::SimbodyUniversalJoint, 1017
- ~SingletonT
 - SingletonT, 1024
- ~Skeleton
 - gazebo::common::Skeleton, 1026
- ~SkeletonAnimation
 - gazebo::common::SkeletonAnimation, 1032
- ~SkeletonNode
 - gazebo::common::SkeletonNode, 1038
- ~SliderJoint
 - gazebo::physics::SliderJoint, 1045
- ~SonarSensor
 - gazebo::sensors::SonarSensor, 1048
- ~SonarVisual
 - gazebo::rendering::SonarVisual, 1052
- ~SphereShape
 - gazebo::physics::SphereShape, 1056
- ~SphericalCoordinates
 - gazebo::common::SphericalCoordinates, 1059
- ~Spline
 - gazebo::math::Spline, 1064
- ~State
 - gazebo::physics::State, 1069
- ~SubMesh
 - gazebo::common::SubMesh, 1077
- ~Subscriber
 - gazebo::transport::Subscriber, 1087
- ~SubscriptionTransport
 - gazebo::transport::SubscriptionTransport, 1089
- ~SurfaceParams
 - gazebo::physics::SurfaceParams, 1091
- ~SystemPlugin
 - gazebo::SystemPlugin, 1099
- ~Time
 - gazebo::common::Time, 1104
- ~Timer
 - gazebo::common::Timer, 1121
- ~TransmitterVisual
 - gazebo::rendering::TransmitterVisual, 1132
- ~UniversalJoint
 - gazebo::physics::UniversalJoint, 1136
- ~UserCamera
 - gazebo::rendering::UserCamera, 1140
- ~Vector2d
 - gazebo::math::Vector2d, 1150
- ~Vector2i
 - gazebo::math::Vector2i, 1158
- ~Vector3
 - gazebo::math::Vector3, 1168
- ~Vector4
 - gazebo::math::Vector4, 1181
- ~Video
 - gazebo::common::Video, 1188
- ~VideoVisual
 - gazebo::rendering::VideoVisual, 1190
- ~ViewController
 - gazebo::rendering::ViewController, 1194
- ~Visual
 - gazebo::rendering::Visual, 1202
- ~WindowManager
 - gazebo::rendering::WindowManager, 1225
- ~WireBox
 - gazebo::rendering::WireBox, 1228
- ~WirelessReceiver
 - gazebo::sensors::WirelessReceiver, 1231
- ~WirelessTransceiver
 - gazebo::sensors::WirelessTransceiver, 1233
- ~WirelessTransmitter
 - gazebo::sensors::WirelessTransmitter, 1237
- ~World
 - gazebo::physics::World, 1242
- ~WorldPlugin
 - gazebo::WorldPlugin, 1252
- ~WorldState
 - gazebo::physics::WorldState, 1255
- ~WrenchVisual
 - gazebo::rendering::WrenchVisual, 1260
- _setupGeometry
 - gazebo::rendering::MovableText, 711

- `_updateColors`
 - `gazebo::rendering::MovableText`, 711
- a
 - `gazebo::common::Color`, 260
- ABGR
 - `gazebo::common::Color`, 252
- ACTOR
 - `gazebo::physics::Base`, 172
- ADD
 - `gazebo::common::Material`, 641
- ARGB
 - `gazebo::common::Color`, 252
- AXIS_CHILD
 - `gazebo::physics::UniversalJoint`, 1135
- AXIS_PARENT
 - `gazebo::physics::UniversalJoint`, 1135
- AcceptCallback
 - `gazebo::transport::Connection`, 266
- active
 - `gazebo::physics::Actor`, 143
 - `gazebo::sensors::Sensor`, 917
- Actor
 - `gazebo::physics::Actor`, 141
- Actor.hh, 1265
- Actor_V
 - `gazebo::physics`, 122
- ActorPtr
 - `gazebo::physics`, 122
- Add
 - `gazebo::util::LogRecord`, 633
- add_plugin
 - `gazebo`, 103
- add_search_path_suffix
 - Common, 41
- AddAnimation
 - `gazebo::common::Skeleton`, 1026
- AddCallback
 - `gazebo::transport::PublicationTransport`, 819
- AddChild
 - `gazebo::common::SkeletonNode`, 1038
 - `gazebo::physics::Base`, 173
- AddChildJoint
 - `gazebo::physics::Link`, 601
- AddContact
 - `gazebo::physics::Collision`, 238
- AddDARTChildJoint
 - `gazebo::physics::DARTLink`, 341
- addEntity
 - `gazebo::event::Events`, 431
- AddForce
 - `gazebo::physics::DARTLink`, 341
 - `gazebo::physics::Link`, 601
 - `gazebo::physics::SimbodyLink`, 974
- AddForceAtRelativePosition
 - `gazebo::physics::DARTLink`, 341
 - `gazebo::physics::Link`, 601
 - `gazebo::physics::SimbodyLink`, 974
- AddForceAtWorldPosition
 - `gazebo::physics::DARTLink`, 341
 - `gazebo::physics::Link`, 601
 - `gazebo::physics::SimbodyLink`, 975
- AddGazeboPaths
 - `gazebo::common::SystemPaths`, 1094
- AddIndex
 - `gazebo::common::SubMesh`, 1077
- AddJoint
 - `gazebo::physics::JointController`, 569
- AddKeyFrame
 - `gazebo::common::NodeAnimation`, 739
 - `gazebo::common::SkeletonAnimation`, 1032
- AddMaterial
 - `gazebo::common::Mesh`, 662
- AddMesh
 - `gazebo::common::MeshManager`, 671
- AddModelPaths
 - `gazebo::common::SystemPaths`, 1094
- AddNode
 - `gazebo::transport::TopicManager`, 1124
- AddNodeAssignment
 - `gazebo::common::SubMesh`, 1077
- AddNodeToProcess
 - `gazebo::transport::TopicManager`, 1125
- AddNormal
 - `gazebo::common::SubMesh`, 1077
- AddOgrePaths
 - `gazebo::common::SystemPaths`, 1094
- AddParentJoint
 - `gazebo::physics::Link`, 602
- addPlugin
 - `gazebo`, 103
- AddPluginPaths
 - `gazebo::common::SystemPaths`, 1094
- AddPoint
 - `gazebo::math::RotationSpline`, 873
 - `gazebo::math::Spline`, 1065
 - `gazebo::rendering::DynamicLines`, 398
- AddPublisher
 - `gazebo::transport::Publication`, 814
- AddRawTransform
 - `gazebo::common::SkeletonNode`, 1038
- AddRay
 - Bullet Physics, 79
 - `gazebo::physics::MultiRayShape`, 726
 - `gazebo::physics::SimbodyMultiRayShape`, 986
- AddRelativeEvent
 - `gazebo::sensors::SimTimeEventHandler`, 1022
- AddRelativeForce

- gazebo::physics::DARTLink, 341
- gazebo::physics::Link, 602
- gazebo::physics::SimbodyLink, 975
- AddRelativeTorque
 - gazebo::physics::DARTLink, 342
 - gazebo::physics::Link, 602
 - gazebo::physics::SimbodyLink, 975
- AddResourcePath
 - gazebo::rendering::RenderEngine, 857
- AddScene
 - gazebo::rendering::RTShaderSystem, 877
- AddSearchPathSuffix
 - gazebo::common::SystemPaths, 1094
- AddSubMesh
 - gazebo::common::Mesh, 662
- AddSubscription
 - gazebo::transport::Publication, 814
- AddTag
 - gazebo::sensors::RFIDSensor, 860
- addTechnique
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 1046
- AddTexCoord
 - gazebo::common::SubMesh, 1077
- AddTime
 - gazebo::common::Animation, 155
- AddTorque
 - gazebo::physics::DARTLink, 342
 - gazebo::physics::Link, 602
 - gazebo::physics::SimbodyLink, 975
- AddTransport
 - gazebo::transport::Publication, 814
- AddType
 - gazebo::physics::Base, 173
- AddVertNodeWeight
 - gazebo::common::Skeleton, 1026
- AddVertex
 - gazebo::common::SubMesh, 1078
- AddVisual
 - gazebo::rendering::Scene, 883
- Advertise
 - gazebo::transport::ConnectionManager, 272
 - gazebo::transport::Node, 733
 - gazebo::transport::TopicManager, 1125
- alt
 - gazebo::common::MouseEvent, 708
- ambient
 - gazebo::common::Material, 647
- anchorLink
 - gazebo::physics::Joint, 566
- anchorPos
 - gazebo::physics::Joint, 566
- anchorPose
 - gazebo::physics::Joint, 566
- Angle
 - gazebo::math::Angle, 147
- Angle.hh, 1266
 - GZ_DTOR, 1268
 - GZ_NORMALIZE, 1268
 - GZ_RTOD, 1268
- angularAccel
 - gazebo::physics::Link, 617
- animState
 - gazebo::rendering::Camera, 222
 - gazebo::rendering::VisualPrivate, 1222
- Animation
 - gazebo::common::Animation, 155
- animation
 - gazebo::physics::Entity, 414
- Animation.hh, 1269
- AnimationComplete
 - gazebo::rendering::Camera, 204
 - gazebo::rendering::UserCamera, 1140
- animationConnection
 - gazebo::physics::Entity, 414
- AnimationPtr
 - gazebo::common, 110
- animationStartPose
 - gazebo::physics::Entity, 414
- animations
 - gazebo::common::SkeletonAnimation, 1034
- anims
 - gazebo::common::Skeleton, 1030
- Apply
 - gazebo::sensors::Noise, 749
- ApplyDamping
 - gazebo::physics::DARTJoint, 330
 - gazebo::physics::Joint, 547
- applyDamping
 - gazebo::physics::Joint, 566
- ApplyImpl
 - gazebo::sensors::GaussianNoiseModel, 459
 - gazebo::sensors::Noise, 750
- ApplyShadows
 - gazebo::rendering::RTShaderSystem, 877
- ApplyStiffnessDamping
 - gazebo::physics::Joint, 547
- AreConnected
 - gazebo::physics::DARTJoint, 330
 - gazebo::physics::Joint, 548
 - gazebo::physics::SimbodyJoint, 963
- ArrowVisual
 - gazebo::rendering::ArrowVisual, 158
- ArrowVisual.hh, 1270
- ArrowVisualPrivate.hh, 1271
- ArrowVisualPtr
 - gazebo::rendering, 127
- Assert.hh, 1271

- GZ_ASSERT, 1272
- AssertionInternalError
 - gazebo::common::AssertionInternalError, 161
- AsyncRead
 - gazebo::transport::Connection, 266
- Attach
 - gazebo::physics::DARTJoint, 330
 - gazebo::physics::Joint, 548
 - gazebo::rendering::SelectionObj, 902
- AttachAxes
 - gazebo::rendering::Visual, 1202
- AttachCameraToImage
 - gazebo::rendering::GUIOverlay, 495
- AttachEntity
 - gazebo::rendering::RTShaderSystem, 877
- AttachLineVertex
 - gazebo::rendering::Visual, 1203
- AttachMesh
 - gazebo::rendering::Visual, 1203
- AttachObject
 - gazebo::rendering::Visual, 1203
- AttachStaticModel
 - gazebo::physics::Link, 602
 - gazebo::physics::Model, 682
- AttachToVisual
 - gazebo::rendering::Camera, 204, 205
- AttachToVisualImpl
 - gazebo::rendering::Camera, 205
 - gazebo::rendering::UserCamera, 1140
- AttachViewport
 - gazebo::rendering::RTShaderSystem, 877
- AttachVisual
 - gazebo::rendering::Visual, 1203
- attachedModels
 - gazebo::physics::Model, 693
- attachedModelsOffset
 - gazebo::physics::Link, 617
 - gazebo::physics::Model, 693
- Attribute
 - gazebo::physics::Joint, 547
- AudioDecoder
 - gazebo::common::AudioDecoder, 162
- AudioDecoder.hh, 1272
- autoCalc
 - gazebo::math::RotationSpline, 875
 - gazebo::math::Spline, 1067
- autoStart
 - gazebo::physics::Actor, 143
- AxisIndex
 - gazebo::physics::UniversalJoint, 1135
- axisParentModelFrame
 - gazebo::physics::Joint, 566
- AxisVisual
 - gazebo::rendering::AxisVisual, 164
- axisVisual
 - gazebo::rendering::JointVisualPrivate, 581
- AxisVisual.hh, 1274
- AxisVisualPrivate.hh, 1274
- AxisVisualPtr
 - gazebo::rendering, 127
- b
 - gazebo::common::Color, 260
- BALL_JOINT
 - gazebo::physics::Base, 172
- BASE
 - gazebo::physics::Base, 172
- BAYER_GBRG8
 - gazebo::common::Image, 516
- BAYER_GRBG8
 - gazebo::common::Image, 516
- BAYER_RRGB8
 - gazebo::common::Image, 516
- BAYER_RGGR8
 - gazebo::common::Image, 516
- BGR_INT16
 - gazebo::common::Image, 516
- BGR_INT32
 - gazebo::common::Image, 516
- BGR_INT8
 - gazebo::common::Image, 516
- BGRA
 - gazebo::common::Color, 252
- BGRA_INT8
 - gazebo::common::Image, 516
- BLEND_COUNT
 - gazebo::common::Material, 641
- BLINN
 - gazebo::common::Material, 642
- BOX_SHAPE
 - gazebo::physics::Base, 172
- BVHLoader
 - gazebo::common::BVHLoader, 191
- BVHLoader.hh, 1280
 - X_POSITION, 1282
 - X_ROTATION, 1282
 - Y_POSITION, 1282
 - Y_ROTATION, 1282
 - Z_POSITION, 1282
 - Z_ROTATION, 1282
- BackupState
 - gazebo::physics::DARTModel, 351
- BallJoint
 - gazebo::physics::BallJoint, 167
- BallJoint.hh, 1275
- Base
 - gazebo::physics::Base, 173
- Base.hh, 1276

- Base64.hh, 1277
 - Base64Decode, 1278
 - Base64Encode, 1278
- Base64Decode
 - Base64.hh, 1278
- Base64Encode
 - Base64.hh, 1278
- Base_V
 - gazebo::physics, 122
- BasePtr
 - gazebo::physics, 122
- bayerFrameBuffer
 - gazebo::rendering::Camera, 222
- bias
 - gazebo::sensors::GaussianNoiseModel, 460
- bindShapeTransform
 - gazebo::common::Skeleton, 1030
- Black
 - gazebo::common::Color, 260
- BlendMode
 - gazebo::common::Material, 641
- blendMode
 - gazebo::common::Material, 647
- BlendModeStr
 - gazebo::common::Material, 647
- Blue
 - gazebo::common::Color, 260
- body1Force
 - gazebo::physics::JointWrench, 582
- body1Torque
 - gazebo::physics::JointWrench, 582
- body2Force
 - gazebo::physics::JointWrench, 583
- body2Torque
 - gazebo::physics::JointWrench, 583
- bonePosePub
 - gazebo::physics::Actor, 143
- BooleanOperation
 - gazebo::common::MeshCSG, 668
- boost, 101
- boundingBox
 - gazebo::rendering::VisualPrivate, 1222
- Box
 - gazebo::math::Box, 182
- Box.hh, 1278
- boxNode
 - gazebo::rendering::COMVisualPrivate, 263
- BoxShape
 - gazebo::physics::BoxShape, 187
- BoxShape.hh, 1279
- BoxShapePtr
 - gazebo::physics, 122
- Buffer
 - gazebo::common::FileLogger::Buffer, 190
 - gazebo::common::Logger::Buffer, 189
- build
 - gazebo::common::Animation, 157
- BuildInterpolationSplines
 - gazebo::common::PoseAnimation, 807
- BuildNodeMap
 - gazebo::common::Skeleton, 1027
- Bullet Physics, 79
 - ~DARTMultiRayShape, 79
 - AddRay, 79
 - DARTMultiRayShape, 79
 - UpdateRays, 80
- button
 - gazebo::common::MouseEvent, 708
- ButtonCallback
 - gazebo::rendering::GUIOverlay, 495
- Buttons
 - gazebo::common::MouseEvent, 707
- buttons
 - gazebo::common::MouseEvent, 708
- CATEGORY_COUNT
 - gazebo::sensors, 133
- CFM
 - gazebo::physics::Joint, 547
- COLLISION
 - gazebo::physics::Base, 172
- COMVisual
 - gazebo::rendering::COMVisual, 261
- COMVisual.hh, 1296
- COMVisualPrivate.hh, 1296
- COMVisualPtr
 - gazebo::rendering, 128
- COR3_MAX
 - STLLoader.hh, 1479
- CUSTOM
 - gazebo::sensors::Noise, 749
- CYLINDER_SHAPE
 - gazebo::physics::Base, 172
- CacheForceTorque
 - gazebo::physics::Joint, 548
 - gazebo::physics::SimbodyJoint, 963
- CallbackFunc
 - gazebo::common::ModelDatabasePrivate, 695
- CallbackHelper
 - gazebo::transport::CallbackHelper, 193
- CallbackHelper.hh, 1282
- CallbackHelperPtr
 - Transport, 94
- CallbackHelperT
 - gazebo::transport::CallbackHelperT, 196
- callbacks
 - gazebo::rendering::GUIOverlay, 498
- callbacksMutex

- gazebo::common::ModelDatabasePrivate, 695
- Camera
 - gazebo::rendering::Camera, 204
- camera
 - gazebo::rendering::Camera, 222
 - gazebo::rendering::CameraVisualPrivate, 234
 - gazebo::rendering::ViewController, 1195
- Camera.hh, 1283
- CameraCmdMsgs_L
 - gazebo::rendering::CameraPrivate, 226
- cameraCount
 - gazebo::rendering::GpuLaser, 475
- cameraCounter
 - gazebo::rendering::CameraPrivate, 226
- cameraElem
 - gazebo::sensors::GpuRaySensor, 487
- CameraPrivate.hh, 1285
- CameraPtr
 - gazebo::rendering, 127
- CameraSensor
 - gazebo::sensors::CameraSensor, 229
- CameraSensor.hh, 1285
- CameraSensor_V
 - gazebo::sensors, 132
- CameraSensorPtr
 - gazebo::sensors, 132
- CameraVisual
 - gazebo::rendering::CameraVisual, 232
- CameraVisual.hh, 1286
- CameraVisualPrivate.hh, 1287
- CameraVisualPtr
 - gazebo::rendering, 128
- Cancel
 - gazebo::transport::Connection, 266
- captureData
 - gazebo::rendering::Camera, 222
- captureDataOnce
 - gazebo::rendering::Camera, 222
- cegui.h, 1287
- Center
 - gazebo::common::Mesh, 663
 - gazebo::common::SubMesh, 1078
- cgVisuals
 - gazebo::physics::Link, 617
- CheckAndTruncateForce
 - gazebo::physics::Joint, 548
- chfov
 - gazebo::rendering::GpuLaser, 475
- childLink
 - gazebo::physics::Joint, 566
- children
 - gazebo::common::SkeletonNode, 1043
 - gazebo::physics::Base, 180
 - gazebo::rendering::VisualPrivate, 1222
- clamp
 - Math, 52
- Classes for physics and dynamics, 68
 - create_world, 72
 - EntityTypename, 75
 - fini, 72
 - GZ_REGISTER_PHYSICS_ENGINE, 72
 - get_world, 72
 - getUniqueld, 73
 - init_world, 73
 - init_worlds, 73
 - load, 73
 - load_world, 73
 - load_worlds, 73
 - pause_world, 73
 - pause_worlds, 74
 - PhysicsFactoryFn, 72
 - remove_worlds, 74
 - run_world, 74
 - run_worlds, 74
 - stop_world, 74
 - stop_worlds, 74
 - worlds_running, 75
- Clear
 - gazebo::math::RotationSpline, 873
 - gazebo::math::Spline, 1065
 - gazebo::physics::ContactManager, 283
 - gazebo::physics::World, 1242
 - gazebo::rendering::DynamicLines, 398
 - gazebo::rendering::RTShaderSystem, 878
 - gazebo::rendering::Scene, 883
- clear_buffers
 - Transport, 94
- ClearBuffers
 - gazebo::transport::TopicManager, 1125
- ClearGazeboPaths
 - gazebo::common::SystemPaths, 1095
- ClearModelPaths
 - gazebo::common::SystemPaths, 1095
- ClearModels
 - gazebo::physics::World, 1242
- ClearOgrePaths
 - gazebo::common::SystemPaths, 1095
- ClearParent
 - gazebo::rendering::Visual, 1203
- ClearPluginPaths
 - gazebo::common::SystemPaths, 1095
- Clone
 - gazebo::rendering::Visual, 1203
- CloneVisual
 - gazebo::rendering::Scene, 883
- cmdSub
 - gazebo::rendering::CameraPrivate, 226
- coeffs

- gazebo::math::Spline, 1067
- ColladaLoader
 - gazebo::common::ColladaLoader, 235
- ColladaLoader.hh, 1288
- collideWithoutContact
 - gazebo::physics::SurfaceParams, 1092
- collideWithoutContactBitmask
 - gazebo::physics::SurfaceParams, 1092
- Collision
 - gazebo::physics::Collision, 238
- Collision.hh, 1289
- collision1
 - gazebo::physics::Contact, 281
- collision2
 - gazebo::physics::Contact, 281
- Collision_V
 - gazebo::physics, 122
- collisionNames
 - gazebo::physics::ContactPublisher, 287
- collisionParent
 - gazebo::physics::Shape, 935
- CollisionPtr
 - gazebo::physics, 122
- CollisionState
 - gazebo::physics::CollisionState, 247
- CollisionState.hh, 1290
- collisions
 - gazebo::physics::ContactPublisher, 287
- Color
 - gazebo::common::Color, 252
- color
 - gazebo::common::Logger, 628
 - gazebo::common::Logger::Buffer, 189
- Color.hh, 1291
- commandMsgs
 - gazebo::rendering::CameraPrivate, 226
- Common, 35
 - ~MovingWindowFilter, 41
 - add_search_path_suffix, 41
 - DownloadDependencies, 41
 - find_file, 41, 42
 - find_file_path, 42
 - Fini, 42
 - Get, 42
 - get_sha1, 42
 - GetDBCConfig, 43
 - GetModelConfig, 43
 - GetModelFile, 43
 - GetModelName, 43
 - GetModelPath, 43
 - GetModels, 44
 - GetURI, 44
 - GetWindowFilled, 44
 - GetWindowSize, 45
 - gzLogInit, 40
 - gzdbg, 39
 - gzerr, 39
 - gzlog, 40
 - gzmsg, 40
 - gzthrow, 40
 - gzwarn, 40
 - HasModel, 45
 - load, 45
 - MODEL_PLUGIN, 40
 - MovingWindowFilter, 41
 - MovingWindowFilterPrivate, 41
 - PixelFormatNames, 46
 - PluginType, 40
 - SENSOR_PLUGIN, 41
 - SYSTEM_PLUGIN, 41
 - SetWindowSize, 45
 - Start, 45
 - Update, 45
 - VISUAL_PLUGIN, 41
 - WORLD_PLUGIN, 40
- Commonface.hh, 1292
- CommonTypes.hh, 1294
 - GAZEBO_DEPRECATED, 1295
 - GAZEBO_FORCEINLINE, 1295
 - NULL, 1295
- ComputeScopedName
 - gazebo::physics::Base, 173
- condition
 - gazebo::sensors::SimTimeEvent, 1021
- coneNode
 - gazebo::rendering::SonarVisualPrivate, 1054
- coneXNode
 - gazebo::rendering::WrenchVisualPrivate, 1262
- coneYNode
 - gazebo::rendering::WrenchVisualPrivate, 1262
- coneZNode
 - gazebo::rendering::WrenchVisualPrivate, 1262
- Connect
 - Events, 48
 - gazebo::transport::Connection, 267
- ConnectAddEntity
 - gazebo::event::Events, 424
- ConnectCreateEntity
 - gazebo::event::Events, 424
- ConnectCreateScene
 - gazebo::rendering::Events, 433
- ConnectDeleteEntity
 - gazebo::event::Events, 424
- ConnectDiagTimerStart
 - gazebo::event::Events, 424
- ConnectDiagTimerStop
 - gazebo::event::Events, 425
- ConnectEnabled

- gazebo::physics::Link, 603
- ConnectJointUpdate
 - gazebo::physics::Joint, 548
- ConnectNewDepthFrame
 - gazebo::rendering::DepthCamera, 385
- ConnectNewImageFrame
 - gazebo::rendering::Camera, 206
- ConnectNewLaserFrame
 - gazebo::rendering::GpuLaser, 470
 - gazebo::sensors::GpuRaySensor, 480
- ConnectNewLaserScans
 - gazebo::physics::MultiRayShape, 726
- ConnectNewRGBPointCloud
 - gazebo::rendering::DepthCamera, 385
- ConnectPause
 - gazebo::event::Events, 425
- ConnectPostRender
 - gazebo::event::Events, 425
- ConnectPreRender
 - gazebo::event::Events, 426
- ConnectPubToSub
 - gazebo::transport::TopicManager, 1125
- ConnectRemoveScene
 - gazebo::rendering::Events, 433
- ConnectRender
 - gazebo::event::Events, 426
- ConnectSetSelectedEntity
 - gazebo::event::Events, 426
- ConnectSigInt
 - gazebo::event::Events, 426
- ConnectStep
 - gazebo::event::Events, 427
- ConnectStop
 - gazebo::event::Events, 427
- ConnectSubToPub
 - gazebo::transport::TopicManager, 1126
- ConnectSubscribers
 - gazebo::transport::TopicManager, 1125
- connectToMaster
 - Transport, 94
- ConnectToRemoteHost
 - gazebo::transport::ConnectionManager, 272
- ConnectToShutdown
 - gazebo::transport::Connection, 267
- ConnectUpdate
 - gazebo::sensors::ForceTorqueSensor, 451
 - gazebo::sensors::SonarSensor, 1048
- ConnectUpdated
 - gazebo::sensors::Sensor, 911
- ConnectWorldCreated
 - gazebo::event::Events, 427
- ConnectWorldUpdateBegin
 - gazebo::event::Events, 427
- ConnectWorldUpdateEnd
 - gazebo::event::Events, 428
- Connection
 - gazebo::event::Connection, 264
 - gazebo::transport::Connection, 266
- connection
 - gazebo::rendering::LaserVisualPrivate, 588
- Connection.hh, 1297
- HEADER_LENGTH, 1299
- Connection_V
 - gazebo::event, 111
- ConnectionCount
 - Events, 48
- ConnectionManager.hh, 1299
- ConnectionPrivate
 - gazebo::event::ConnectionPrivate, 276
- ConnectionPtr
 - gazebo::event, 111
 - gazebo::transport, 136
- ConnectionReadTask
 - gazebo::transport::ConnectionReadTask, 277
- connections
 - gazebo::event::EventTPrivate, 444
 - gazebo::physics::Entity, 415
 - gazebo::rendering::Camera, 222
 - gazebo::rendering::CameraVisualPrivate, 234
 - gazebo::rendering::ContactVisualPrivate, 295
 - gazebo::rendering::GUIOverlayPrivate, 498
 - gazebo::rendering::SonarVisualPrivate, 1054
 - gazebo::rendering::TransmitterVisualPrivate, 1134
 - gazebo::rendering::VideoVisualPrivate, 1191
 - gazebo::rendering::WrenchVisualPrivate, 1262
 - gazebo::sensors::Sensor, 917
- connectionsEraseMutex
 - gazebo::event::EventTPrivate, 444
- connectionsToErase
 - gazebo::event::EventTPrivate, 444
- Console.hh, 1300
- constraint
 - gazebo::physics::SimbodyJoint, 970
- Contact
 - gazebo::physics::Contact, 280
- contact
 - gazebo::physics::SimbodyPhysics, 996
- Contact.hh, 1301
- MAX_COLLIDE_RETURNS, 1302
- MAX_CONTACT_JOINTS, 1302
- contactFiducial
 - gazebo::physics::RayShape, 854
- contactLen
 - gazebo::physics::RayShape, 854
- ContactManager
 - gazebo::physics::ContactManager, 283
- contactManager
 - gazebo::physics::PhysicsEngine, 780

- ContactManager.hh, 1303
- ContactPtr
 - gazebo::physics, 122
- contactRetro
 - gazebo::physics::RayShape, 854
- ContactSensor
 - gazebo::sensors::ContactSensor, 289
- ContactSensor.hh, 1304
- ContactSensor_V
 - gazebo::sensors, 132
- ContactSensorPtr
 - gazebo::sensors, 132
- ContactVisual
 - gazebo::rendering::ContactVisual, 293
- ContactVisual.hh, 1304
- ContactVisualPrivate.hh, 1305
- ContactVisualPtr
 - gazebo::rendering, 128
- contacts
 - gazebo::physics::ContactPublisher, 287
- contactsMsg
 - gazebo::rendering::ContactVisualPrivate, 295
- contactsSub
 - gazebo::rendering::ContactVisualPrivate, 295
- control
 - gazebo::common::MouseEvent, 708
- ConvPose
 - DART Physics, 77
- ConvQuat
 - DART Physics, 77, 78
- ConvVec3
 - DART Physics, 78
- Conversions.hh, 1306
- Convert
 - gazebo::common::SphericalCoordinates, 1059
 - gazebo::rendering::Conversions, 296, 297
 - Messages, 59–62
- ConvertPixelFormat
 - gazebo::common::Image, 517
- CoordPoseSolve
 - gazebo::math::Pose, 799
- CoordPositionAdd
 - gazebo::math::Pose, 800
- CoordPositionSub
 - gazebo::math::Pose, 800
- CoordRotationAdd
 - gazebo::math::Pose, 800
- CoordRotationSub
 - gazebo::math::Pose, 800
- CopyNormals
 - gazebo::common::SubMesh, 1078
- CopyVertices
 - gazebo::common::SubMesh, 1078
- Correct
 - gazebo::math::Pose, 801
 - gazebo::math::Quaternion, 828
 - gazebo::math::Vector3, 1168
- count
 - gazebo::physics::Contact, 281
- Create
 - gazebo::PluginT, 796
- create_scene
 - Rendering, 85
- create_sensor
 - Sensors, 89
- create_world
 - Classes for physics and dynamics, 72
- CreateBoolean
 - gazebo::common::MeshCSG, 668
- CreateBox
 - gazebo::common::MeshManager, 671
- CreateCamera
 - gazebo::common::MeshManager, 672
 - gazebo::rendering::Scene, 884
- CreateCollision
 - gazebo::physics::DARTPhysics, 356
 - gazebo::physics::PhysicsEngine, 770
 - gazebo::physics::SimbodyPhysics, 989
- CreateCone
 - gazebo::common::MeshManager, 672
- CreateCylinder
 - gazebo::common::MeshManager, 672
- CreateDepthCamera
 - gazebo::rendering::Scene, 884
- CreateDepthTexture
 - gazebo::rendering::DepthCamera, 385
- CreateDynamicLine
 - gazebo::rendering::Visual, 1204
- CreateFilter
 - gazebo::physics::ContactManager, 283, 284
- CreateGpuLaser
 - gazebo::rendering::Scene, 884
- CreateGrid
 - gazebo::rendering::Scene, 885
- CreateJoint
 - gazebo::physics::DARTPhysics, 356
 - gazebo::physics::PhysicsEngine, 771
 - gazebo::physics::SimbodyPhysics, 989
- CreateKeyFrame
 - gazebo::common::NumericAnimation, 753
 - gazebo::common::PoseAnimation, 807
- CreateLaserTexture
 - gazebo::rendering::GpuLaser, 470
- CreateLink
 - gazebo::physics::DARTPhysics, 356
 - gazebo::physics::PhysicsEngine, 771
 - gazebo::physics::SimbodyPhysics, 990
- CreateModel

- gazebo::physics::DARTPhysics, 357
- gazebo::physics::PhysicsEngine, 771
- gazebo::physics::SimbodyPhysics, 990
- CreatePlane
 - gazebo::common::MeshManager, 672, 673
 - gazebo::physics::DARTPlaneShape, 362
 - gazebo::physics::PlaneShape, 792
 - gazebo::physics::SimbodyPlaneShape, 999
- CreateRenderTexture
 - gazebo::rendering::Camera, 206
- CreateRequest
 - Messages, 62
- CreateScene
 - gazebo::rendering::RenderEngine, 857
- createScene
 - gazebo::rendering::Events, 434
- CreateSensor
 - gazebo::sensors::SensorManager, 922
- CreateShape
 - gazebo::physics::DARTPhysics, 357
 - gazebo::physics::PhysicsEngine, 771
 - gazebo::physics::SimbodyPhysics, 990
- CreateSink
 - gazebo::util::OpenAL, 756
- CreateSource
 - gazebo::util::OpenAL, 756
- CreateSphere
 - gazebo::common::MeshManager, 673
- CreateTube
 - gazebo::common::MeshManager, 673
- CreateUserCamera
 - gazebo::rendering::Scene, 885
- CreateVertexDeclaration
 - gazebo::rendering::DynamicRenderable, 402
- CreateWindow
 - gazebo::rendering::GUIOverlay, 496
 - gazebo::rendering::WindowManager, 1226
- creationTime
 - gazebo::event::ConnectionPrivate, 276
- Cross
 - gazebo::math::Vector2d, 1150
 - gazebo::math::Vector2i, 1158
 - gazebo::math::Vector3, 1168
- crossLines
 - gazebo::rendering::COMVisualPrivate, 263
- cvfov
 - gazebo::rendering::GpuLaser, 476
- CylinderShape
 - gazebo::physics::CylinderShape, 299
- CylinderShape.hh, 1307
- CylinderShapePtr
 - gazebo::physics, 122
- d
 - gazebo::math::Plane, 790
 - DART Physics, 76
 - ~DARTRayShape, 77
 - ConvPose, 77
 - ConvQuat, 77, 78
 - ConvVec3, 78
 - DARTRayShape, 77
 - GetIntersection, 78
 - SetPoints, 78
 - Update, 78
 - DARTBallJoint
 - gazebo::physics::DARTBallJoint, 303
 - DARTBallJoint.hh, 1308
 - DARTBoxShape
 - gazebo::physics::DARTBoxShape, 309
 - DARTBoxShape.hh, 1309
 - DARTCollision
 - gazebo::physics::DARTCollision, 311
 - DARTCollision.hh, 1310
 - DARTCollisionPtr
 - gazebo::physics, 122
 - DARTCylinderShape
 - gazebo::physics::DARTCylinderShape, 314
 - DARTCylinderShape.hh, 1311
 - DARTHeightmapShape
 - gazebo::physics::DARTHeightmapShape, 316
 - DARTHeightmapShape.hh, 1311
 - DARTHinge2Joint
 - gazebo::physics::DARTHinge2Joint, 319
 - DARTHinge2Joint.hh, 1312
 - DARTHingeJoint
 - gazebo::physics::DARTHingeJoint, 324
 - DARTHingeJoint.hh, 1312
 - DARTJoint
 - gazebo::physics::DARTJoint, 330
 - DARTJoint.hh, 1313
 - DARTJointPtr
 - gazebo::physics, 122
 - DARTLink
 - gazebo::physics::DARTLink, 340
 - DARTLink.hh, 1313
 - DARTLinkPtr
 - gazebo::physics, 122
 - DARTMeshShape
 - gazebo::physics::DARTMeshShape, 349
 - DARTMeshShape.hh, 1314
 - DARTModel
 - gazebo::physics::DARTModel, 351
 - DARTModel.hh, 1315
 - DARTModelPtr
 - gazebo::physics, 122
 - DARTMultiRayShape
 - Bullet Physics, 79
 - DARTMultiRayShape.hh, 1315

- DARTParam
 - gazebo::physics::DARTPhysics, 356
- DARTPhysics
 - gazebo::physics::DARTPhysics, 356
- DARTPhysics.hh, 1316
- DARTPhysicsPtr
 - gazebo::physics, 122
- DARTPlaneShape
 - gazebo::physics::DARTPlaneShape, 362
- DARTPlaneShape.hh, 1316
- DARTRayShape
 - DART Physics, 77
- DARTRayShape.hh, 1317
- DARTRayShapePtr
 - gazebo::physics, 122
- DARTScrewJoint
 - gazebo::physics::DARTScrewJoint, 366
- DARTScrewJoint.hh, 1318
- DARTSliderJoint
 - gazebo::physics::DARTSliderJoint, 373
- DARTSliderJoint.hh, 1318
- DARTSphereShape
 - gazebo::physics::DARTSphereShape, 377
- DARTSphereShape.hh, 1319
- DARTTypes.hh, 1319
- DARTUniversalJoint
 - gazebo::physics::DARTUniversalJoint, 380
- DARTUniversalJoint.hh, 1320
- DEFERRED
 - gazebo::rendering::RenderEngine, 856
- DIAG_TIMER_LAP
 - Utility, 99
- DIAG_TIMER_START
 - Utility, 99
- DIAG_TIMER_STOP
 - Utility, 99
- DIFFERENCE
 - gazebo::common::MeshCSG, 668
- damper
 - gazebo::physics::SimbodyJoint, 970
- dampingCoefficient
 - gazebo::physics::Joint, 567
- dart_inc.h, 1308
- dartPhysicsEngine
 - gazebo::physics::DARTJoint, 337
- dartScrewJoint
 - gazebo::physics::DARTScrewJoint, 371
- dataPtr
 - gazebo::common::MovingWindowFilter, 716
 - gazebo::event::Event, 418
 - gazebo::rendering::Visual, 1217
- dbg
 - gazebo::common::Console, 278
- DebugPrint
 - gazebo::physics::DARTPhysics, 357
 - gazebo::physics::PhysicsEngine, 771
 - gazebo::physics::SimbodyPhysics, 990
- DebugString
 - gazebo::physics::Contact, 280
- DecCount
 - gazebo::transport::IOManager, 541
- Decode
 - gazebo::common::AudioDecoder, 162
- DecodeTopicName
 - gazebo::transport::Node, 733
- defaultVpParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperCg, 929
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperGLSL, 931
- defxAB
 - gazebo::physics::SimbodyJoint, 970
- Degree
 - gazebo::math::Angle, 148
- DeleteDynamicLine
 - gazebo::rendering::Visual, 1204
- deleteEntity
 - gazebo::event::Events, 431
- Dem.hh, 1321
- DemPrivate.hh, 1322
- deprecatedCallbacks
 - gazebo::common::ModelDatabasePrivate, 695
- depth
 - gazebo::rendering::ContactVisualPrivate::Contact-
Point, 286
- DepthCamera
 - gazebo::rendering::DepthCamera, 384
- DepthCamera.hh, 1323
- DepthCameraPtr
 - gazebo::rendering, 128
- DepthCameraSensor
 - gazebo::sensors::DepthCameraSensor, 388
- DepthCameraSensor.hh, 1324
- DepthCameraSensor_V
 - gazebo::sensors, 132
- DepthCameraSensorPtr
 - gazebo::sensors, 132
- depthTarget
 - gazebo::rendering::DepthCamera, 387
- depthTexture
 - gazebo::rendering::DepthCamera, 387
- depthViewport
 - gazebo::rendering::DepthCamera, 387
- depths
 - gazebo::physics::Contact, 281
- Detach
 - gazebo::physics::DARTJoint, 331
 - gazebo::physics::Joint, 549

- gazebo::physics::SimbodyJoint, 963
- gazebo::rendering::SelectionObj, 902
- DetachAllStaticModels
 - gazebo::physics::Link, 603
- DetachEntity
 - gazebo::rendering::RTShaderSystem, 878
- DetachObjects
 - gazebo::rendering::Visual, 1204
- DetachStaticModel
 - gazebo::physics::Link, 603
 - gazebo::physics::Model, 682
- DetachViewport
 - gazebo::rendering::RTShaderSystem, 878
- DetachVisual
 - gazebo::rendering::Visual, 1204
- diagTimerStart
 - gazebo::event::Events, 431
- diagTimerStop
 - gazebo::event::Events, 431
- DiagnosticTimer
 - gazebo::util::DiagnosticTimer, 394
- DiagnosticTimerPtr
 - gazebo::common, 110
 - gazebo::util, 137
- Diagnostics.hh, 1324
- diffuse
 - gazebo::common::Material, 647
- direction1
 - gazebo::physics::FrictionPyramid, 457
- dirtyPose
 - gazebo::physics::Entity, 415
- dirtyPoses
 - gazebo::physics::World, 1251
- disable
 - Sensors, 89
- DisableAllModels
 - gazebo::physics::World, 1242
- DisableTrackVisual
 - gazebo::rendering::Visual, 1204
- Disconnect
 - Events, 48
 - gazebo::event::Event, 417, 418
- DisconnectAddEntity
 - gazebo::event::Events, 428
- DisconnectCreateEntity
 - gazebo::event::Events, 428
- DisconnectCreateScene
 - gazebo::rendering::Events, 434
- DisconnectDeleteEntity
 - gazebo::event::Events, 428
- DisconnectDiagTimerStart
 - gazebo::event::Events, 429
- DisconnectDiagTimerStop
 - gazebo::event::Events, 429
- DisconnectEnabled
 - gazebo::physics::Link, 603
- DisconnectJointUpdate
 - gazebo::physics::Joint, 549
- DisconnectNewDepthFrame
 - gazebo::rendering::DepthCamera, 385
- DisconnectNewImageFrame
 - gazebo::rendering::Camera, 206
- DisconnectNewLaserFrame
 - gazebo::rendering::GpuLaser, 470
 - gazebo::sensors::GpuRaySensor, 480
- DisconnectNewLaserScans
 - gazebo::physics::MultiRayShape, 727
- DisconnectNewRGBPointCloud
 - gazebo::rendering::DepthCamera, 385
- DisconnectPause
 - gazebo::event::Events, 429
- DisconnectPostRender
 - gazebo::event::Events, 429
- DisconnectPreRender
 - gazebo::event::Events, 429
- DisconnectPubFromSub
 - gazebo::transport::TopicManager, 1126
- DisconnectRemoveScene
 - gazebo::rendering::Events, 434
- DisconnectRender
 - gazebo::event::Events, 429
- DisconnectSetSelectedEntity
 - gazebo::event::Events, 430
- DisconnectShutdown
 - gazebo::transport::Connection, 267
- DisconnectSigInt
 - gazebo::event::Events, 430
- DisconnectStep
 - gazebo::event::Events, 430
- DisconnectStop
 - gazebo::event::Events, 430
- DisconnectSubFromPub
 - gazebo::transport::TopicManager, 1126
- DisconnectUpdate
 - gazebo::sensors::ForceTorqueSensor, 451
 - gazebo::sensors::SonarSensor, 1048
- DisconnectUpdated
 - gazebo::sensors::Sensor, 911
- DisconnectWorldCreated
 - gazebo::event::Events, 430
- DisconnectWorldUpdateBegin
 - gazebo::event::Events, 430
- DisconnectWorldUpdateEnd
 - gazebo::event::Events, 431
- discreteForces
 - gazebo::physics::SimbodyPhysics, 996
- dissipationCoefficient
 - gazebo::physics::Joint, 567

- Distance
 - gazebo::common::SphericalCoordinates, 1059
 - gazebo::math::Plane, 789
 - gazebo::math::Vector2d, 1150
 - gazebo::math::Vector2i, 1158
 - gazebo::math::Vector3, 1168
 - gazebo::math::Vector4, 1181
- dlGBufferInstance
 - gazebo::rendering::CameraPrivate, 226
- dlMergeInstance
 - gazebo::rendering::CameraPrivate, 226
- Dot
 - gazebo::math::Quaternion, 828
 - gazebo::math::Vector3, 1169
- Double
 - gazebo::common::Time, 1104
- DownloadDependencies
 - Common, 41
- dragging
 - gazebo::common::MouseEvent, 708
- DrawLine
 - gazebo::rendering::Scene, 885
- dsGBufferInstance
 - gazebo::rendering::CameraPrivate, 226
- dsMergeInstance
 - gazebo::rendering::CameraPrivate, 226
- dtBallJoint
 - gazebo::physics::DARTBallJoint, 307
- dtChildBodyNode
 - gazebo::physics::DARTJoint, 337
- dtConfig
 - gazebo::physics::DARTModel, 352
- dtJoint
 - gazebo::physics::DARTJoint, 337
- dtPrismaticJoint
 - gazebo::physics::DARTSliderJoint, 375
- dtRevoluteJoint
 - gazebo::physics::DARTHingeJoint, 327
- dtSkeleton
 - gazebo::physics::DARTModel, 352
- dtUniversalJoint
 - gazebo::physics::DARTHinge2Joint, 322
 - gazebo::physics::DARTUniversalJoint, 383
- dtVelocity
 - gazebo::physics::DARTModel, 352
- dummyContext
 - gazebo::rendering::RenderEngine, 858
- dummyDisplay
 - gazebo::rendering::RenderEngine, 858
- dummyWindowId
 - gazebo::rendering::RenderEngine, 859
- duration
 - gazebo::physics::TrajectoryInfo, 1130
- DynamicLines
 - gazebo::rendering::DynamicLines, 398
- DynamicLines.hh, 1325
- DynamicLinesPtr
 - gazebo::rendering, 128
- DynamicRenderable
 - gazebo::rendering::DynamicRenderable, 401
- DynamicRenderable.hh, 1326
- EARTH_WGS84
 - gazebo::common::SphericalCoordinates, 1058
- ENTITY
 - gazebo::physics::Base, 172
- ERP
 - gazebo::physics::Joint, 547
- effortLimit
 - gazebo::physics::Joint, 567
- elevationReference
 - gazebo::common::SphericalCoordinatesPrivate, 1063
- emissive
 - gazebo::common::Material, 647
- Enable
 - gazebo::rendering::Grid, 490
- enable
 - Sensors, 90
- EnableAllModels
 - gazebo::physics::World, 1242
- EnablePhysicsEngine
 - gazebo::physics::World, 1242
- EnableSaveFrame
 - gazebo::rendering::Camera, 206
- EnableTrackVisual
 - gazebo::rendering::Visual, 1205
- EnableViewController
 - gazebo::rendering::UserCamera, 1140
- enabled
 - gazebo::rendering::ContactVisualPrivate, 295
 - gazebo::rendering::ViewController, 1196
 - gazebo::rendering::WrenchVisualPrivate, 1262
- EncodeTopicName
 - gazebo::transport::Node, 733
- endTime
 - gazebo::physics::TrajectoryInfo, 1130
- EnqueueMsg
 - gazebo::transport::Connection, 267, 268
- Entity
 - gazebo::physics::Entity, 407
- Entity.hh, 1327
- entityCreated
 - gazebo::event::Events, 431
- EntityPtr
 - gazebo::physics, 122
- EntityType
 - gazebo::physics::Base, 172

- EntityTypename
 - Classes for physics and dynamics, 75
- Equal
 - gazebo::math::Vector3, 1169
- equal
 - Math, 52
- err
 - gazebo::common::Console, 278
- EulerToQuaternion
 - gazebo::math::Quaternion, 829
- Event
 - gazebo::event::Event, 417
- event
 - gazebo::event::ConnectionPrivate, 276
- Event.hh, 1328
- eventConnections
 - gazebo::transport::ConnectionManager, 275
- EventPrivate
 - gazebo::event::EventPrivate, 421
- EventT
 - Events, 47
- EventType
 - gazebo::common::KeyEvent, 584
 - gazebo::common::MouseEvent, 707
- Events, 47
 - ~EventT, 47
 - Connect, 48
 - ConnectionCount, 48
 - Disconnect, 48
 - EventT, 47
- Events.hh, 1329
- Exception
 - gazebo::common::Exception, 446
- Exception.hh, 1330
- execute
 - gazebo::transport::ConnectionReadTask, 277
 - gazebo::transport::PublishTask, 824

- FACE_MAX
 - STLLoader.hh, 1479
- FLAT
 - gazebo::common::Material, 642
- FMAX
 - gazebo::physics::Joint, 547
- FORWARD
 - gazebo::rendering::RenderEngine, 856
- FPSViewController
 - gazebo::rendering::FPSViewController, 454
- FPSViewController.hh, 1333
- FUDGE_FACTOR
 - gazebo::physics::Joint, 547
- far
 - gazebo::rendering::GpuLaser, 476
- ffmpeg_inc.h, 1332

- FileLogger
 - gazebo::common::FileLogger, 448
- filename
 - gazebo::PluginT, 796
- FillArrays
 - gazebo::common::Mesh, 663
 - gazebo::common::SubMesh, 1079
- FillBufferFromFile
 - gazebo::util::OpenALSource, 760
- FillBufferFromPCM
 - gazebo::util::OpenALSource, 760
- FillHardwareBuffers
 - gazebo::rendering::DynamicRenderable, 402
- FillHeightMap
 - gazebo::common::HeightmapData, 505
 - gazebo::common::ImageHeightmap, 524
- FillMsg
 - gazebo::physics::BoxShape, 187
 - gazebo::physics::Collision, 238
 - gazebo::physics::Contact, 280
 - gazebo::physics::CylinderShape, 299
 - gazebo::physics::HeightmapShape, 509
 - gazebo::physics::Joint, 549
 - gazebo::physics::Link, 603
 - gazebo::physics::MeshShape, 676
 - gazebo::physics::Model, 683
 - gazebo::physics::MultiRayShape, 727
 - gazebo::physics::PlaneShape, 792
 - gazebo::physics::RayShape, 851
 - gazebo::physics::Shape, 934
 - gazebo::physics::SphereShape, 1056
 - gazebo::physics::SurfaceParams, 1091
 - gazebo::rendering::Light, 591
 - gazebo::sensors::Sensor, 911
- FillSDF
 - gazebo::physics::CollisionState, 247
 - gazebo::physics::JointState, 576
 - gazebo::physics::LinkState, 621
 - gazebo::physics::ModelState, 701
 - gazebo::physics::WorldState, 1255
- find_file
 - Common, 41, 42
 - gazebo, 104
- find_file_path
 - Common, 42
- FindFile
 - gazebo::common::SystemPaths, 1095
- FindFileURI
 - gazebo::common::SystemPaths, 1095
- FindPublication
 - gazebo::transport::TopicManager, 1126
- Fini
 - Common, 42
 - gazebo::Master, 638

- gazebo::physics::Actor, 142
 - gazebo::physics::Base, 173
 - gazebo::physics::Collision, 239
 - gazebo::physics::DARTCollision, 311
 - gazebo::physics::DARTLink, 342
 - gazebo::physics::DARTModel, 351
 - gazebo::physics::DARTPhysics, 357
 - gazebo::physics::Entity, 407
 - gazebo::physics::Joint, 549
 - gazebo::physics::Link, 604
 - gazebo::physics::Model, 683
 - gazebo::physics::PhysicsEngine, 772
 - gazebo::physics::SimbodyLink, 975
 - gazebo::physics::SimbodyPhysics, 990
 - gazebo::physics::World, 1242
 - gazebo::rendering::Camera, 207
 - gazebo::rendering::DepthCamera, 386
 - gazebo::rendering::GpuLaser, 470
 - gazebo::rendering::RenderEngine, 857
 - gazebo::rendering::RTShaderSystem, 878
 - gazebo::rendering::UserCamera, 1141
 - gazebo::rendering::Visual, 1205
 - gazebo::rendering::WindowManager, 1226
 - gazebo::sensors::CameraSensor, 229
 - gazebo::sensors::ContactSensor, 289
 - gazebo::sensors::DepthCameraSensor, 388
 - gazebo::sensors::ForceTorqueSensor, 451
 - gazebo::sensors::GaussianNoiseModel, 459
 - gazebo::sensors::GpsSensor, 465
 - gazebo::sensors::GpuRaySensor, 480
 - gazebo::sensors::ImageGaussianNoiseModel, 522
 - gazebo::sensors::ImuSensor, 527
 - gazebo::sensors::MultiCameraSensor, 720
 - gazebo::sensors::Noise, 750
 - gazebo::sensors::RaySensor, 844
 - gazebo::sensors::RFIDSensor, 860
 - gazebo::sensors::RFIDTag, 863
 - gazebo::sensors::Sensor, 912
 - gazebo::sensors::SensorManager, 922
 - gazebo::sensors::SonarSensor, 1049
 - gazebo::sensors::WirelessReceiver, 1231
 - gazebo::sensors::WirelessTransceiver, 1234
 - gazebo::Server, 927
 - gazebo::transport::ConnectionManager, 273
 - gazebo::transport::Node, 734
 - gazebo::transport::PublicationTransport, 819
 - gazebo::transport::Publisher, 821
 - gazebo::transport::TopicManager, 1126
 - gazebo::util::LogRecord, 634
 - gazebo::util::OpenAL, 757
- fini
- Classes for physics and dynamics, 72
 - gazebo, 104
 - Rendering, 85
 - Sensors, 90
 - Transport, 94
 - fixnan
 - Math, 52
 - Flatten
 - gazebo::rendering::Heightmap, 501
 - flipY
 - gazebo::physics::HeightmapShape, 511
 - Float
 - gazebo::common::Time, 1104
 - FogFromSDF
 - Messages, 62
 - forceLine
 - gazebo::rendering::WrenchVisualPrivate, 1262
 - forceNode
 - gazebo::rendering::WrenchVisualPrivate, 1262
 - ForceTorqueSensor
 - gazebo::sensors::ForceTorqueSensor, 450
 - ForceTorqueSensor.hh, 1332
 - ForceTorqueSensorPtr
 - gazebo::sensors, 132
 - forces
 - gazebo::physics::JointControllerPrivate, 573
 - gazebo::physics::SimbodyPhysics, 996
 - fpsViewController
 - gazebo::rendering::UserCameraPrivate, 1147
 - freq
 - gazebo::sensors::WirelessTransmitter, 1238
 - FrictionPyramid
 - gazebo::physics::FrictionPyramid, 456
 - g
 - gazebo::common::Color, 260
 - GAUSSIAN
 - gazebo::sensors::Noise, 749
 - GAZEBO_DEPRECATED
 - CommonTypes.hh, 1295
 - GAZEBO_FORCEINLINE
 - CommonTypes.hh, 1295
 - GAZEBO_HIDDEN
 - system.hh, 1484
 - GAZEBO_VISIBLE
 - system.hh, 1484
 - GEARBOX_JOINT
 - gazebo::physics::Base, 172
 - GOURAUD
 - gazebo::common::Material, 642
 - GPtrArray
 - MeshCSG.hh, 1380
 - GUIFromSDF
 - Messages, 63
 - GUIOverlay
 - gazebo::rendering::GUIOverlay, 495
 - GUIOverlay.hh, 1342

- GUIOverlayPrivate.hh, 1343
- GUIPluginPtr
 - gazebo, 103
- GZ_ALL_COLLIDE
 - PhysicsTypes.hh, 1409
- GZ_ASSERT
 - Assert.hh, 1272
- GZ_DBL_MAX
 - Helpers.hh, 1349
- GZ_DBL_MIN
 - Helpers.hh, 1349
- GZ_DTOR
 - Angle.hh, 1268
- GZ_FIXED_COLLIDE
 - PhysicsTypes.hh, 1409
- GZ_FLT_MAX
 - Helpers.hh, 1349
- GZ_FLT_MIN
 - Helpers.hh, 1349
- GZ_GHOST_COLLIDE
 - PhysicsTypes.hh, 1409
- GZ_INT32_MAX
 - Helpers.hh, 1349
- GZ_INT32_MIN
 - Helpers.hh, 1349
- GZ_LOG_VERSION
 - LogRecord.hh, 1371
- GZ_MODEL_DB_MANIFEST_FILENAME
 - ModelDatabase.hh, 1387
- GZ_MODEL_MANIFEST_FILENAME
 - ModelDatabase.hh, 1387
- GZ_NONE_COLLIDE
 - PhysicsTypes.hh, 1409
- GZ_NORMALIZE
 - Angle.hh, 1268
- GZ_REGISTER_MODEL_PLUGIN
 - Plugin.hh, 1414
- GZ_REGISTER_PHYSICS_ENGINE
 - Classes for physics and dynamics, 72
- GZ_REGISTER_SENSOR_PLUGIN
 - Plugin.hh, 1414
- GZ_REGISTER_STATIC_MSG
 - Messages, 59
- GZ_REGISTER_STATIC_SENSOR
 - Sensors, 89
- GZ_REGISTER_SYSTEM_PLUGIN
 - Plugin.hh, 1415
- GZ_REGISTER_VISUAL_PLUGIN
 - Plugin.hh, 1415
- GZ_REGISTER_WORLD_PLUGIN
 - Plugin.hh, 1415
- GZ_RTOD
 - Angle.hh, 1268
- GZ_SENSOR_COLLIDE
 - PhysicsTypes.hh, 1409
- GZ_SKYX_ALL
 - gazebo::rendering::Scene, 883
- GZ_SKYX_CLOUDS
 - gazebo::rendering::Scene, 883
- GZ_SKYX_MOON
 - gazebo::rendering::Scene, 883
- GZ_SKYX_NONE
 - gazebo::rendering::Scene, 883
- GZ_UINT32_MAX
 - Helpers.hh, 1349
- GZ_UINT32_MIN
 - Helpers.hh, 1349
- GZ_VISIBILITY_ALL
 - RenderTypes.hh, 1430
- GZ_VISIBILITY_GUI
 - RenderTypes.hh, 1430
- GZ_VISIBILITY_SELECTABLE
 - RenderTypes.hh, 1430
- GZ_VISIBILITY_SELECTION
 - RenderTypes.hh, 1430
- gain
 - gazebo::sensors::WirelessTransceiver, 1235
- gaussianNoiseCompositorListener
 - gazebo::sensors::ImageGaussianNoiseModel, 522
- gaussianNoiseInstance
 - gazebo::sensors::ImageGaussianNoiseModel, 522
- GaussianNoiseModel
 - gazebo::sensors::GaussianNoiseModel, 459
- GaussianNoiseModel.hh, 1334
- GaussianNoiseModelPtr
 - gazebo::sensors, 132
- gazebo, 101
 - add_plugin, 103
 - addPlugin, 103
 - find_file, 104
 - fini, 104
 - GUIPluginPtr, 103
 - init, 104
 - load, 104
 - loadWorld, 104
 - ModelPluginPtr, 103
 - print_version, 105
 - printVersion, 105
 - run, 105
 - runWorld, 105
 - SensorPluginPtr, 103
 - setupClient, 105
 - setupServer, 105
 - shutdown, 106
 - stop, 106
 - SystemPluginPtr, 103
 - VisualPluginPtr, 103
 - WorldPluginPtr, 103

- gazebo.hh, 1334
- gazebo::Master, 637
 - ~Master, 638
 - Fini, 638
 - Init, 638
 - Master, 638
 - Run, 638
 - RunOnce, 638
 - RunThread, 638
 - Stop, 639
- gazebo::ModelPlugin, 696
 - ~ModelPlugin, 697
 - Init, 697
 - Load, 697
 - ModelPlugin, 697
 - Reset, 697
- gazebo::PluginT
 - ~PluginT, 795
 - Create, 796
 - filename, 796
 - GetFilename, 796
 - GetHandle, 796
 - GetType, 796
 - handle, 796
 - PluginT, 795
 - TPtr, 795
 - type, 796
- gazebo::PluginT< T >, 794
- gazebo::SensorPlugin, 924
 - ~SensorPlugin, 925
 - Init, 925
 - Load, 925
 - Reset, 925
 - SensorPlugin, 925
- gazebo::Server, 926
 - ~Server, 926
 - Fini, 927
 - GetInitialized, 927
 - LoadFile, 927
 - LoadString, 927
 - ParseArgs, 927
 - PreLoad, 928
 - PrintUsage, 928
 - Run, 928
 - Server, 926
 - SetParams, 928
 - Stop, 928
- gazebo::SystemPlugin, 1098
 - ~SystemPlugin, 1099
 - Init, 1099
 - Load, 1099
 - Reset, 1099
 - SystemPlugin, 1098
- gazebo::VisualPlugin, 1217
 - Init, 1218
 - Load, 1218
 - Reset, 1219
 - VisualPlugin, 1218
- gazebo::WorldPlugin, 1251
 - ~WorldPlugin, 1252
 - Init, 1253
 - Load, 1253
 - Reset, 1253
 - WorldPlugin, 1252
- gazebo::common, 106
 - AnimationPtr, 110
 - DiagnosticTimerPtr, 110
 - NodeMap, 110
 - NodeMapIter, 110
 - NumericAnimationPtr, 110
 - Param_V, 110
 - PoseAnimationPtr, 110
 - RawNodeAnim, 110
 - RawNodeWeights, 110
 - RawSkeletonAnim, 110
 - SpeedOfLight, 110
 - SphericalCoordinatesPtr, 110
 - StrStr_M, 110
- gazebo::common::Animation, 153
 - ~Animation, 155
 - AddTime, 155
 - Animation, 155
 - build, 157
 - GetKeyFrame, 155
 - GetKeyFrameCount, 155
 - GetKeyFramesAtTime, 156
 - GetLength, 156
 - GetTime, 156
 - KeyFrame_V, 155
 - keyFrames, 157
 - length, 157
 - loop, 157
 - name, 157
 - SetLength, 156
 - SetTime, 156
 - timePos, 157
- gazebo::common::AssertionInternalError, 160
 - ~AssertionInternalError, 161
 - AssertionInternalError, 161
- gazebo::common::AudioDecoder, 161
 - ~AudioDecoder, 162
 - AudioDecoder, 162
 - Decode, 162
 - GetFile, 162
 - GetSampleRate, 163
 - SetFile, 163
- gazebo::common::BVHLoader, 191
 - ~BVHLoader, 191

- BVHLoader, 191
- Load, 191
- gazebo::common::ColladaLoader, 234
 - ~ColladaLoader, 235
 - ColladaLoader, 235
 - Load, 235
- gazebo::common::Color, 249
 - ~Color, 252
 - a, 260
 - ABGR, 252
 - ARGB, 252
 - b, 260
 - BGRA, 252
 - Black, 260
 - Blue, 260
 - Color, 252
 - g, 260
 - GetAsABGR, 253
 - GetAsARGB, 253
 - GetAsBGRA, 253
 - GetAsHSV, 253
 - GetAsRGBA, 253
 - GetAsYUV, 253
 - Green, 260
 - operator<<, 259
 - operator>>, 259
 - operator*, 254
 - operator*=:, 254
 - operator+, 255
 - operator+=, 255
 - operator-, 255, 256
 - operator-=, 256
 - operator/, 256
 - operator/=, 257
 - operator=, 257
 - operator==, 257
 - operator[], 257
 - Purple, 260
 - r, 260
 - RGBA, 252
 - Red, 260
 - Reset, 258
 - Set, 258
 - SetFromABGR, 258
 - SetFromARGB, 258
 - SetFromBGRA, 258
 - SetFromHSV, 258
 - SetFromRGBA, 259
 - SetFromYUV, 259
 - White, 260
 - Yellow, 260
- gazebo::common::Console, 277
 - dbg, 278
 - err, 278
 - GetQuiet, 278
 - log, 278
 - msg, 278
 - SetQuiet, 278
 - warn, 279
- gazebo::common::Exception, 444
 - ~Exception, 446
 - Exception, 446
 - GetErrorFile, 446
 - GetErrorStr, 446
 - operator<<, 446
 - Print, 446
- gazebo::common::FileLogger, 447
 - ~FileLogger, 448
 - FileLogger, 448
 - Init, 448
 - operator(), 448
- gazebo::common::FileLogger::Buffer, 189
 - ~Buffer, 190
 - Buffer, 190
 - stream, 190
 - sync, 190
- gazebo::common::HeightmapData, 504
 - ~HeightmapData, 505
 - FillHeightMap, 505
 - GetHeight, 506
 - GetMaxElevation, 506
 - GetWidth, 506
- gazebo::common::Image, 514
 - ~Image, 517
 - BAYER_GBRG8, 516
 - BAYER_GRBG8, 516
 - BAYER_RGGB8, 516
 - BAYER_RGGR8, 516
 - BGR_INT16, 516
 - BGR_INT32, 516
 - BGR_INT8, 516
 - BGRA_INT8, 516
 - ConvertPixelFormat, 517
 - GetAvgColor, 517
 - GetBPP, 517
 - GetData, 517
 - GetFilename, 517
 - GetHeight, 518
 - GetMaxColor, 518
 - GetPitch, 518
 - GetPixel, 518
 - GetPixelFormat, 518
 - GetRGBData, 519
 - GetWidth, 519
 - Image, 516
 - L_INT16, 516
 - L_INT8, 516
 - Load, 519

- PIXEL_FORMAT_COUNT, 516
- PixelFormat, 516
- R_FLOAT16, 516
- R_FLOAT32, 516
- RGB_FLOAT16, 516
- RGB_FLOAT32, 516
- RGB_INT16, 516
- RGB_INT32, 516
- RGB_INT8, 516
- RGBA_INT8, 516
- Rescale, 519
- SavePNG, 519
- SetFromData, 520
- UNKNOWN_PIXEL_FORMAT, 516
- Valid, 520
- gazebo::common::ImageHeightmap, 523
 - FillHeightMap, 524
 - GetFilename, 524
 - GetHeight, 524
 - GetMaxElevation, 524
 - GetWidth, 525
 - ImageHeightmap, 524
 - Load, 525
- gazebo::common::InternalError, 538
 - ~InternalError, 540
 - InternalError, 539
- gazebo::common::KeyEvent, 583
 - EventType, 584
 - key, 584
 - KeyEvent, 584
 - NO_EVENT, 584
 - PRESS, 584
 - RELEASE, 584
 - type, 584
- gazebo::common::KeyFrame, 584
 - ~KeyFrame, 585
 - GetTime, 585
 - KeyFrame, 585
 - time, 585
- gazebo::common::Logger, 625
 - ~Logger, 627
 - color, 628
 - LogType, 627
 - Logger, 627
 - operator(), 627, 628
 - STDERR, 627
 - STDOUT, 627
- gazebo::common::Logger::Buffer, 188
 - ~Buffer, 189
 - Buffer, 189
 - color, 189
 - sync, 189
 - type, 189
- gazebo::common::Material, 639
 - ~Material, 642
 - ADD, 641
 - ambient, 647
 - BLEND_COUNT, 641
 - BLINN, 642
 - BlendMode, 641
 - blendMode, 647
 - BlendModeStr, 647
 - diffuse, 647
 - emissive, 647
 - FLAT, 642
 - GOURAUD, 642
 - GetAmbient, 642
 - GetBlendFactors, 642
 - GetBlendMode, 642
 - GetDepthWrite, 643
 - GetDiffuse, 643
 - GetEmissive, 643
 - GetLighting, 643
 - GetName, 643
 - GetPointSize, 643
 - GetShadeMode, 644
 - GetShininess, 644
 - GetSpecular, 644
 - GetTextureImage, 644
 - GetTransparency, 644
 - MODULATE, 641
 - Material, 642
 - name, 647
 - operator<<, 647
 - PHONG, 642
 - pointSize, 647
 - REPLACE, 641
 - SHADE_COUNT, 642
 - SetAmbient, 644
 - SetBlendFactors, 645
 - SetBlendMode, 645
 - SetDepthWrite, 645
 - SetDiffuse, 645
 - SetEmissive, 645
 - SetLighting, 646
 - SetPointSize, 646
 - SetShadeMode, 646
 - SetShininess, 646
 - SetSpecular, 646
 - SetTextureImage, 646
 - SetTransparency, 647
 - ShadeMode, 641
 - shadeMode, 648
 - ShadeModeStr, 648
 - shininess, 648
 - specular, 648
 - texImage, 648
 - transparency, 648

- gazebo::common::Mesh, 660
 - ~Mesh, 662
 - AddMaterial, 662
 - AddSubMesh, 662
 - Center, 663
 - FillArrays, 663
 - GenSphericalTexCoord, 663
 - GetAABB, 663
 - GetIndexCount, 663
 - GetMaterial, 664
 - GetMaterialCount, 664
 - GetMax, 664
 - GetMin, 664
 - GetName, 664
 - GetNormalCount, 664
 - GetPath, 665
 - GetSkeleton, 665
 - GetSubMesh, 665
 - GetSubMeshCount, 665
 - GetTexCoordCount, 666
 - GetVertexCount, 666
 - HasSkeleton, 666
 - Mesh, 662
 - RecalculateNormals, 666
 - Scale, 666
 - SetName, 666
 - SetPath, 666
 - SetScale, 667
 - SetSkeleton, 667
 - Translate, 667
- gazebo::common::MeshCSG, 667
 - ~MeshCSG, 668
 - BooleanOperation, 668
 - CreateBoolean, 668
 - DIFFERENCE, 668
 - INTERSECTION, 668
 - MeshCSG, 668
 - UNION, 668
- gazebo::common::MeshLoader, 669
 - ~MeshLoader, 669
 - Load, 670
 - MeshLoader, 669
- gazebo::common::MeshManager, 670
 - AddMesh, 671
 - CreateBox, 671
 - CreateCamera, 672
 - CreateCone, 672
 - CreateCylinder, 672
 - CreatePlane, 672, 673
 - CreateSphere, 673
 - CreateTube, 673
 - GenSphericalTexCoord, 673
 - GetMesh, 674
 - GetMeshAABB, 674
 - HasMesh, 674
 - IsValidFilename, 674
 - Load, 674
- gazebo::common::ModelDatabase, 693
- gazebo::common::ModelDatabasePrivate, 694
 - CallbackFunc, 695
 - callbacksMutex, 695
 - deprecatedCallbacks, 695
 - modelCache, 695
 - modelDBUpdated, 695
 - startCacheMutex, 695
 - stop, 696
 - updateCacheCompleteCondition, 696
 - updateCacheCondition, 696
 - updateCacheThread, 696
 - updateMutex, 696
- gazebo::common::MouseEvent, 706
 - alt, 708
 - button, 708
 - Buttons, 707
 - buttons, 708
 - control, 708
 - dragging, 708
 - EventType, 707
 - LEFT, 707
 - MIDDLE, 707
 - MOVE, 708
 - MouseEvent, 708
 - moveScale, 708
 - NO_BUTTON, 707
 - NO_EVENT, 708
 - PRESS, 708
 - pos, 708
 - pressPos, 708
 - prevPos, 709
 - RELEASE, 708
 - RIGHT, 707
 - SCROLL, 708
 - scroll, 709
 - shift, 709
 - type, 709
- gazebo::common::MovingWindowFilter
 - dataPtr, 716
 - MovingWindowFilter, 716
- gazebo::common::MovingWindowFilter< T >, 715
- gazebo::common::MovingWindowFilterPrivate
 - samples, 716
 - sum, 716
 - valHistory, 717
 - vallter, 717
 - valWindowSize, 717
- gazebo::common::MovingWindowFilterPrivate< T >, 716
- gazebo::common::NodeAnimation, 738
 - ~NodeAnimation, 739

- AddKeyFrame, 739
- GetFrameAt, 739
- GetFrameCount, 739
- GetKeyFrame, 740
- GetLength, 740
- GetName, 740
- GetTimeAtX, 740
- keyFrames, 741
- length, 741
- name, 741
- NodeAnimation, 739
- Scale, 741
- SetName, 741
- gazebo::common::NodeAssignment, 741
 - NodeAssignment, 742
 - nodeIndex, 742
 - vertexIndex, 742
 - weight, 742
- gazebo::common::NodeTransform, 742
 - ~NodeTransform, 745
 - Get, 745
 - GetSID, 745
 - GetType, 745
 - MATRIX, 744
 - NodeTransform, 744
 - operator*, 745, 746
 - operator(), 745
 - PrintSource, 746
 - ROTATE, 744
 - RecalculateMatrix, 746
 - SCALE, 744
 - Set, 746
 - SetComponent, 746
 - SetSID, 746
 - SetSourceValues, 746, 747
 - SetType, 747
 - sid, 747
 - source, 747
 - TRANSLATE, 744
 - transform, 747
 - TransformType, 744
 - type, 747
- gazebo::common::NumericAnimation, 752
 - ~NumericAnimation, 753
 - CreateKeyFrame, 753
 - GetInterpolatedKeyFrame, 753
 - NumericAnimation, 753
- gazebo::common::NumericKeyFrame, 754
 - ~NumericKeyFrame, 755
 - GetValue, 755
 - NumericKeyFrame, 754
 - SetValue, 755
 - value, 755
- gazebo::common::PID, 782
 - ~PID, 784
 - GetCmd, 784
 - GetCmdMax, 784
 - GetCmdMin, 784
 - GetDGain, 785
 - GetErrors, 785
 - GetIGain, 785
 - GetIMax, 785
 - GetIMin, 785
 - GetPGain, 785
 - Init, 786
 - operator=, 786
 - PID, 784
 - Reset, 786
 - SetCmd, 786
 - SetCmdMax, 786
 - SetCmdMin, 787
 - SetDGain, 787
 - SetIGain, 787
 - SetIMax, 787
 - SetIMin, 787
 - SetPGain, 787
 - Update, 788
- gazebo::common::PoseAnimation, 805
 - ~PoseAnimation, 807
 - BuildInterpolationSplines, 807
 - CreateKeyFrame, 807
 - GetInterpolatedKeyFrame, 807
 - PoseAnimation, 806
- gazebo::common::PoseKeyFrame, 808
 - ~PoseKeyFrame, 809
 - GetRotation, 809
 - GetTranslation, 809
 - PoseKeyFrame, 809
 - rotate, 810
 - SetRotation, 809
 - SetTranslation, 809
 - translate, 810
- gazebo::common::STLLoader, 1072
 - ~STLLoader, 1073
 - Load, 1073
 - STLLoader, 1073
- gazebo::common::Skeleton, 1024
 - ~Skeleton, 1026
 - AddAnimation, 1026
 - AddVertNodeWeight, 1026
 - anims, 1030
 - bindShapeTransform, 1030
 - BuildNodeMap, 1027
 - GetAnimation, 1027
 - GetBindShapeTransform, 1027
 - GetNodeByHandle, 1027
 - GetNodeById, 1027
 - GetNodeByName, 1028

- GetNodes, 1028
- GetNumAnimations, 1028
- GetNumJoints, 1028
- GetNumNodes, 1028
- GetNumVertNodeWeights, 1028
- GetRootNode, 1029
- GetVertNodeWeight, 1029
- nodes, 1030
- PrintTransforms, 1029
- rawNW, 1030
- root, 1030
- Scale, 1029
- SetBindShapeTransform, 1029
- SetNumVertAttached, 1030
- SetRootNode, 1030
- Skeleton, 1026
- gazebo::common::SkeletonAnimation, 1031
 - ~SkeletonAnimation, 1032
 - AddKeyFrame, 1032
 - animations, 1034
 - GetLength, 1032
 - GetName, 1033
 - GetNodeCount, 1033
 - GetNodePoseAt, 1033
 - GetPoseAt, 1033
 - GetPoseAtX, 1034
 - HasNode, 1034
 - length, 1034
 - name, 1035
 - Scale, 1034
 - SetName, 1034
 - SkeletonAnimation, 1032
- gazebo::common::SkeletonNode, 1035
 - ~SkeletonNode, 1038
 - AddChild, 1038
 - AddRawTransform, 1038
 - children, 1043
 - GetChild, 1038
 - GetChildById, 1038
 - GetChildByName, 1039
 - GetChildCount, 1039
 - GetHandle, 1039
 - GetId, 1039
 - GetInverseBindTransform, 1039
 - GetModelTransform, 1039
 - GetName, 1040
 - GetNumRawTrans, 1040
 - GetParent, 1040
 - GetRawTransform, 1040
 - GetRawTransforms, 1040
 - GetTransform, 1041
 - GetTransforms, 1041
 - handle, 1043
 - id, 1043
 - initialTransform, 1043
 - invBindTransform, 1043
 - IsJoint, 1041
 - IsRootNode, 1041
 - JOINT, 1037
 - modelTransform, 1043
 - NODE, 1037
 - name, 1043
 - parent, 1044
 - rawTransforms, 1044
 - Reset, 1041
 - SetHandle, 1041
 - SetId, 1041
 - SetInitialTransform, 1042
 - SetInverseBindTransform, 1042
 - SetModelTransform, 1042
 - SetName, 1042
 - SetParent, 1042
 - SetTransform, 1042
 - SetType, 1043
 - SkeletonNode, 1037, 1038
 - SkeletonNodeType, 1037
 - transform, 1044
 - type, 1044
 - UpdateChildrenTransforms, 1043
- gazebo::common::SphericalCoordinates, 1057
 - ~SphericalCoordinates, 1059
 - Convert, 1059
 - Distance, 1059
 - EARTH_WGS84, 1058
 - GetElevationReference, 1060
 - GetHeadingOffset, 1060
 - GetLatitudeReference, 1060
 - GetLongitudeReference, 1060
 - GetSurfaceType, 1060
 - GlobalFromLocal, 1061
 - SetElevationReference, 1061
 - SetHeadingOffset, 1061
 - SetLatitudeReference, 1061
 - SetLongitudeReference, 1061
 - SetSurfaceType, 1062
 - SphericalCoordinates, 1059
 - SphericalFromLocal, 1062
 - SurfaceType, 1058
- gazebo::common::SphericalCoordinatesPrivate, 1062
 - elevationReference, 1063
 - headingOffset, 1063
 - latitudeReference, 1063
 - longitudeReference, 1063
 - surfaceType, 1063
- gazebo::common::SubMesh, 1074
 - ~SubMesh, 1077
 - AddIndex, 1077
 - AddNodeAssignment, 1077

- AddNormal, 1077
- AddTexCoord, 1077
- AddVertex, 1078
- Center, 1078
- CopyNormals, 1078
- CopyVertices, 1078
- FillArrays, 1079
- GenSphericalTexCoord, 1079
- GetIndex, 1079
- GetIndexCount, 1079
- GetMaterialIndex, 1079
- GetMax, 1079
- GetMaxIndex, 1079
- GetMin, 1079
- GetName, 1080
- GetNodeAssignment, 1080
- GetNodeAssignmentsCount, 1080
- GetNormal, 1080
- GetNormalCount, 1080
- GetPrimitiveType, 1080
- GetTexCoord, 1081
- GetTexCoordCount, 1081
- GetVertex, 1081
- GetVertexCount, 1081
- GetVertexIndex, 1081
- HasVertex, 1081
- LINES, 1076
- LINESTRIPS, 1076
- POINTS, 1076
- PrimitiveType, 1076
- RecalculateNormals, 1082
- Scale, 1082
- SetIndexCount, 1082
- SetMaterialIndex, 1082
- SetName, 1082
- SetNormal, 1082
- SetNormalCount, 1083
- SetPrimitiveType, 1083
- SetScale, 1083
- SetSubMeshCenter, 1083
- SetTexCoord, 1083
- SetTexCoordCount, 1083
- SetVertex, 1084
- SetVertexCount, 1084
- SubMesh, 1076
- TRIANGLES, 1076
- TRIFANS, 1076
- TRISTRIPS, 1076
- Translate, 1084
- gazebo::common::SystemPaths, 1092
 - AddGazeboPaths, 1094
 - AddModelPaths, 1094
 - AddOgrePaths, 1094
 - AddPluginPaths, 1094
 - AddSearchPathSuffix, 1094
 - ClearGazeboPaths, 1095
 - ClearModelPaths, 1095
 - ClearOgrePaths, 1095
 - ClearPluginPaths, 1095
 - FindFile, 1095
 - FindFileURI, 1095
 - gazeboPathsFromEnv, 1097
 - GetDefaultTestPath, 1095
 - GetGazeboPaths, 1096
 - GetLogPath, 1096
 - GetModelPaths, 1096
 - GetOgrePaths, 1096
 - GetPluginPaths, 1096
 - GetTmpInstancePath, 1096
 - GetTmpPath, 1097
 - GetWorldPathExtension, 1097
 - modelPathsFromEnv, 1097
 - ogrePathsFromEnv, 1097
 - pluginPathsFromEnv, 1097
 - gazebo::common::Time, 1099
 - ~Time, 1104
 - Double, 1104
 - Float, 1104
 - GetWallTime, 1104
 - GetWallTimeAsISOString, 1105
 - MSleep, 1105
 - MicToNano, 1105
 - MilToNano, 1105
 - NSleep, 1106
 - nsec, 1120
 - operator<, 1113, 1114
 - operator<<, 1120
 - operator<=, 1114, 1115
 - operator>, 1117
 - operator>>, 1120
 - operator>=, 1118
 - operator*, 1107
 - operator*=:, 1108
 - operator+, 1108, 1109
 - operator+=, 1109, 1110
 - operator-, 1110
 - operator-=, 1111
 - operator/, 1111, 1112
 - operator/=, 1112, 1113
 - operator=, 1115
 - operator==, 1116
 - sec, 1120
 - SecToNano, 1119
 - Set, 1119
 - SetToWallTime, 1119
 - Sleep, 1119
 - Time, 1103, 1104
 - Zero, 1120

- gazebo::common::Timer, 1120
 - ~Timer, 1121
 - GetElapsed, 1122
 - GetRunning, 1122
 - operator<<, 1122
 - Start, 1122
 - Stop, 1122
 - Timer, 1121
- gazebo::common::UpdateInfo, 1136
 - realTime, 1137
 - simTime, 1137
 - worldName, 1137
- gazebo::common::Video, 1188
 - ~Video, 1188
 - GetHeight, 1188
 - GetNextFrame, 1188
 - GetWidth, 1189
 - Load, 1189
 - Video, 1188
- gazebo::event, 110
 - Connection_V, 111
 - ConnectionPtr, 111
- gazebo::event::Connection, 263
 - ~Connection, 264
 - Connection, 264
 - GetId, 264
- gazebo::event::ConnectionPrivate, 275
 - ConnectionPrivate, 276
 - creationTime, 276
 - event, 276
 - id, 276
- gazebo::event::Event, 416
 - ~Event, 417
 - dataPtr, 418
 - Disconnect, 417, 418
 - Event, 417
 - GetSignaled, 418
- gazebo::event::EventPrivate, 419
 - EventPrivate, 421
 - signaled, 421
- gazebo::event::EventT
 - operator(), 437–439
 - Signal, 440–442
- gazebo::event::EventT< T >, 434
- gazebo::event::EventTPrivate
 - connections, 444
 - connectionsEraseMutex, 444
 - connectionsToErase, 444
- gazebo::event::EventTPrivate< T >, 443
- gazebo::event::Events, 421
 - addEntity, 431
 - ConnectAddEntity, 424
 - ConnectCreateEntity, 424
 - ConnectDeleteEntity, 424
 - ConnectDiagTimerStart, 424
 - ConnectDiagTimerStop, 425
 - ConnectPause, 425
 - ConnectPostRender, 425
 - ConnectPreRender, 426
 - ConnectRender, 426
 - ConnectSetSelectedEntity, 426
 - ConnectSigInt, 426
 - ConnectStep, 427
 - ConnectStop, 427
 - ConnectWorldCreated, 427
 - ConnectWorldUpdateBegin, 427
 - ConnectWorldUpdateEnd, 428
 - deleteEntity, 431
 - diagTimerStart, 431
 - diagTimerStop, 431
 - DisconnectAddEntity, 428
 - DisconnectCreateEntity, 428
 - DisconnectDeleteEntity, 428
 - DisconnectDiagTimerStart, 429
 - DisconnectDiagTimerStop, 429
 - DisconnectPause, 429
 - DisconnectPostRender, 429
 - DisconnectPreRender, 429
 - DisconnectRender, 429
 - DisconnectSetSelectedEntity, 430
 - DisconnectSigInt, 430
 - DisconnectStep, 430
 - DisconnectStop, 430
 - DisconnectWorldCreated, 430
 - DisconnectWorldUpdateBegin, 430
 - DisconnectWorldUpdateEnd, 431
 - entityCreated, 431
 - pause, 431
 - postRender, 431
 - preRender, 431
 - render, 432
 - setSelectedEntity, 432
 - sigInt, 432
 - step, 432
 - stop, 432
 - worldCreated, 432
 - worldUpdateBegin, 432
 - worldUpdateEnd, 432
- gazebo::math, 111
 - GeneratorType, 113
 - NRealGen, 113
 - NormalRealDist, 113
 - UIntGen, 113
 - URealGen, 113
 - UniformIntDist, 113
 - UniformRealDist, 113
- gazebo::math::Angle, 145
 - ~Angle, 147

- Angle, 147
- Degree, 148
- HalfPi, 153
- Normalize, 148
- operator<, 150
- operator<<, 152
- operator<=, 150
- operator>, 151
- operator>>, 152
- operator>=, 151
- operator*, 148
- operator*=: 148
- operator+, 149
- operator+=, 149
- operator-, 149
- operator-=, 149
- operator/, 150
- operator/=, 150
- operator==, 151
- Pi, 153
- Radian, 151
- SetFromDegree, 152
- SetFromRadian, 152
- TwoPi, 153
- Zero, 153
- gazebo::math::Box, 181
 - ~Box, 182
 - Box, 182
 - GetCenter, 182
 - GetSize, 182
 - GetXLength, 183
 - GetYLength, 183
 - GetZLength, 183
 - max, 185
 - Merge, 183
 - min, 185
 - operator<<, 185
 - operator+, 183
 - operator+=, 184
 - operator-, 184
 - operator=, 184
 - operator==, 184
- gazebo::math::Matrix3, 648
 - ~Matrix3, 650
 - IDENTITY, 653
 - m, 653
 - Matrix3, 650
 - operator<<, 653
 - operator*, 650–652
 - operator+, 651
 - operator-, 651
 - operator==, 651
 - operator[], 651
 - SetCol, 652
 - SetFromAxes, 652
 - SetFromAxis, 652
 - ZERO, 653
- gazebo::math::Matrix4, 653
 - ~Matrix4, 656
 - GetAsPose, 656
 - GetEulerRotation, 656
 - GetRotation, 656
 - GetTranslation, 656
 - IDENTITY, 660
 - Inverse, 656
 - IsAffine, 657
 - m, 660
 - Matrix4, 655
 - operator<<, 660
 - operator*, 657
 - operator=, 657, 658
 - operator==, 658
 - operator[], 658
 - Set, 659
 - SetScale, 659
 - SetTranslate, 659
 - TransformAffine, 659
 - ZERO, 660
- gazebo::math::Plane, 788
 - ~Plane, 789
 - d, 790
 - Distance, 789
 - normal, 790
 - operator=, 790
 - Plane, 789
 - Set, 790
 - size, 790
- gazebo::math::Pose, 797
 - ~Pose, 799
 - CoordPoseSolve, 799
 - CoordPositionAdd, 800
 - CoordPositionSub, 800
 - CoordRotationAdd, 800
 - CoordRotationSub, 800
 - Correct, 801
 - GetInverse, 801
 - IsFinite, 801
 - operator<<, 804
 - operator>>, 805
 - operator*, 801
 - operator+, 802
 - operator+=, 802
 - operator-, 802
 - operator-=, 803
 - operator=, 803
 - operator==, 803
 - pos, 805
 - Pose, 799

- Reset, 803
- rot, 805
- RotatePositionAboutOrigin, 803
- Round, 804
- Set, 804
- Zero, 805
- gazebo::math::Quaternion, 824
 - ~Quaternion, 828
 - Correct, 828
 - Dot, 828
 - EulerToQuaternion, 829
 - GetAsAxis, 829
 - GetAsEuler, 829
 - GetAsMatrix3, 829
 - GetAsMatrix4, 829
 - GetExp, 830
 - GetInverse, 830
 - GetLog, 830
 - GetPitch, 830
 - GetRoll, 830
 - GetXAxis, 830
 - GetYAxis, 831
 - GetYaw, 831
 - GetZAxis, 831
 - Invert, 831
 - IsFinite, 831
 - Normalize, 831
 - operator<<, 837
 - operator>>, 837
 - operator*, 832
 - operator*=:, 832
 - operator+, 833
 - operator+=, 833
 - operator-, 833
 - operator-=, 834
 - operator=, 834
 - operator==, 834
 - Quaternion, 827, 828
 - RotateVector, 834
 - RotateVectorReverse, 834
 - Round, 835
 - Scale, 835
 - Set, 835
 - SetFromAxis, 835
 - SetFromEuler, 836
 - SetToIdentity, 836
 - Slerp, 836
 - Squad, 836
 - w, 837
 - x, 837
 - y, 837
 - z, 838
- gazebo::math::Rand, 838
 - GetDbfNormal, 838
 - GetDbfUniform, 839
 - GetIntNormal, 839
 - GetIntUniform, 839
 - GetSeed, 839
 - SetSeed, 839
- gazebo::math::RotationSpline, 871
 - ~RotationSpline, 872
 - AddPoint, 873
 - autoCalc, 875
 - Clear, 873
 - GetNumPoints, 873
 - GetPoint, 873
 - Interpolate, 873, 874
 - points, 875
 - RecalcTangents, 874
 - RotationSpline, 872
 - SetAutoCalculate, 874
 - tangents, 875
 - UpdatePoint, 874
- gazebo::math::Spline, 1063
 - ~Spline, 1064
 - AddPoint, 1065
 - autoCalc, 1067
 - Clear, 1065
 - coeffs, 1067
 - GetPoint, 1065
 - GetPointCount, 1065
 - GetTangent, 1065
 - GetTension, 1065
 - Interpolate, 1066
 - points, 1067
 - RecalcTangents, 1066
 - SetAutoCalculate, 1066
 - SetTension, 1066
 - Spline, 1064
 - tangents, 1067
 - tension, 1067
 - UpdatePoint, 1067
- gazebo::math::Vector2d, 1147
 - ~Vector2d, 1150
 - Cross, 1150
 - Distance, 1150
 - IsFinite, 1150
 - Normalize, 1150
 - operator<<, 1155
 - operator>>, 1155
 - operator*, 1151
 - operator*=:, 1151
 - operator+, 1152
 - operator+=, 1152
 - operator-, 1152
 - operator-=, 1152
 - operator/, 1153
 - operator/=, 1153

- operator=, 1154
- operator==, 1154
- operator[], 1154
- Set, 1155
- Vector2d, 1149
- x, 1155
- y, 1156
- gazebo::math::Vector2i, 1156
 - ~Vector2i, 1158
 - Cross, 1158
 - Distance, 1158
 - IsFinite, 1159
 - Normalize, 1159
 - operator<<, 1164
 - operator>>, 1164
 - operator*, 1159
 - operator*=: 1160
 - operator+, 1160
 - operator+=, 1160
 - operator-, 1161
 - operator-=, 1161
 - operator/, 1161, 1162
 - operator/=, 1162
 - operator=, 1163
 - operator==, 1163
 - operator[], 1163
 - Set, 1163
 - Vector2i, 1158
 - x, 1164
 - y, 1164
- gazebo::math::Vector3, 1165
 - ~Vector3, 1168
 - Correct, 1168
 - Cross, 1168
 - Distance, 1168
 - Dot, 1169
 - Equal, 1169
 - GetAbs, 1169
 - GetDistToLine, 1169
 - GetLength, 1170
 - GetMax, 1170
 - GetMin, 1170
 - GetNormal, 1170
 - GetPerpendicular, 1170
 - GetRounded, 1171
 - GetSquaredLength, 1171
 - GetSum, 1171
 - IsFinite, 1171
 - Normalize, 1171
 - One, 1177
 - operator<<, 1177
 - operator>>, 1177
 - operator*, 1172, 1176
 - operator*=: 1172
 - operator+, 1173
 - operator+=, 1173
 - operator-, 1173
 - operator-=, 1173
 - operator/, 1174
 - operator/=, 1174
 - operator=, 1175
 - operator==, 1175
 - operator[], 1175
 - Round, 1176
 - Set, 1176
 - SetToMax, 1176
 - SetToMin, 1176
 - UnitX, 1177
 - UnitY, 1177
 - UnitZ, 1178
 - Vector3, 1167, 1168
 - x, 1178
 - y, 1178
 - z, 1178
 - Zero, 1178
- gazebo::math::Vector4, 1178
 - ~Vector4, 1181
 - Distance, 1181
 - GetLength, 1181
 - GetSquaredLength, 1181
 - IsFinite, 1181
 - Normalize, 1181
 - operator<<, 1186
 - operator>>, 1187
 - operator*, 1182
 - operator*=: 1183
 - operator+, 1183
 - operator+=, 1183
 - operator-, 1184
 - operator-=, 1184
 - operator/, 1184
 - operator/=, 1185
 - operator=, 1185, 1186
 - operator==, 1186
 - operator[], 1186
 - Set, 1186
 - Vector4, 1180, 1181
 - w, 1187
 - x, 1187
 - y, 1187
 - z, 1187
- gazebo::msgs, 113
 - MsgFactoryFn, 115
- gazebo::msgs::MsgFactory, 717
 - GetMsgTypes, 717
 - NewMsg, 718
 - RegisterMsg, 718
- gazebo::physics, 115

- Actor_V, 122
- ActorPtr, 122
- Base_V, 122
- BasePtr, 122
- BoxShapePtr, 122
- Collision_V, 122
- CollisionPtr, 122
- ContactPtr, 122
- CylinderShapePtr, 122
- DARTCollisionPtr, 122
- DARTJointPtr, 122
- DARTLinkPtr, 122
- DARTModelPtr, 122
- DARTPhysicsPtr, 122
- DARTRayShapePtr, 122
- EntityPtr, 122
- GripperPtr, 122
- HeightmapShapePtr, 122
- InertialPtr, 122
- Joint_V, 122
- JointController_V, 122
- JointControllerPtr, 122
- JointPtr, 122
- JointState_M, 123
- Link_V, 123
- LinkPtr, 123
- LinkState_M, 123
- MeshShapePtr, 123
- Model_V, 123
- ModelPtr, 123
- ModelState_M, 123
- MultiRayShapePtr, 123
- PhysicsEnginePtr, 123
- RayShapePtr, 123
- RoadPtr, 123
- ShapePtr, 123
- SimbodyCollisionPtr, 123
- SimbodyLinkPtr, 123
- SimbodyModelPtr, 123
- SimbodyPhysicsPtr, 123
- SimbodyRayShapePtr, 123
- SphereShapePtr, 123
- SurfaceParamsPtr, 123
- WorldPtr, 123
- gazebo::physics::Actor, 139
 - ~Actor, 141
 - active, 143
 - Actor, 141
 - autoStart, 143
 - bonePosePub, 143
 - Fini, 142
 - GetSDF, 142
 - Init, 142
 - interpolateX, 143
 - isActive, 142
 - lastPos, 143
 - lastScriptTime, 143
 - lastTraj, 143
 - Load, 142
 - loop, 143
 - mainLink, 144
 - mesh, 144
 - oldAction, 144
 - pathLength, 144
 - Play, 142
 - playStartTime, 144
 - prevFrameTime, 144
 - scriptLength, 144
 - skelAnimation, 144
 - skelNodesMap, 144
 - skeleton, 144
 - skinFile, 144
 - skinScale, 145
 - startDelay, 145
 - Stop, 142
 - trajInfo, 145
 - trajectories, 145
 - Update, 142
 - UpdateParameters, 143
 - visualId, 145
 - visualName, 145
- gazebo::physics::BallJoint
 - ~BallJoint, 167
 - BallJoint, 167
 - GetAngleCount, 168
 - Init, 168
 - Load, 168
- gazebo::physics::BallJoint< T >, 167
- gazebo::physics::Base, 168
 - ~Base, 173
 - ACTOR, 172
 - AddChild, 173
 - AddType, 173
 - BALL_JOINT, 172
 - BASE, 172
 - BOX_SHAPE, 172
 - Base, 173
 - COLLISION, 172
 - CYLINDER_SHAPE, 172
 - children, 180
 - ComputeScopedName, 173
 - ENTITY, 172
 - EntityType, 172
 - Fini, 173
 - GEARBOX_JOINT, 172
 - GetById, 174
 - GetByName, 174
 - GetChild, 174

- GetChildCount, 175
- GetId, 175
- GetName, 175
- GetParent, 175
- GetParentId, 175
- GetSDF, 176
- GetSaveable, 175
- GetScopedName, 176
- GetType, 176
- GetWorld, 176
- HEIGHTMAP_SHAPE, 172
- HINGE2_JOINT, 172
- HINGE_JOINT, 172
- HasType, 176
- Init, 177
- IsSelected, 177
- JOINT, 172
- LIGHT, 172
- LINK, 172
- Load, 177
- MAP_SHAPE, 172
- MESH_SHAPE, 173
- MODEL, 172
- MULTIRAY_SHAPE, 172
- operator==, 178
- PLANE_SHAPE, 173
- parent, 180
- Print, 178
- RAY_SHAPE, 172
- RemoveChild, 178
- RemoveChildren, 178
- Reset, 178, 179
- SCREW_JOINT, 172
- SENSOR_COLLISION, 173
- SHAPE, 172
- SLIDER_JOINT, 172
- SPHERE_SHAPE, 173
- sdf, 180
- SetName, 179
- SetParent, 179
- SetSaveable, 179
- SetSelected, 179
- SetWorld, 180
- UNIVERSAL_JOINT, 172
- Update, 180
- UpdateParameters, 180
- VISUAL, 172
- world, 180
- gazebo::physics::BoxShape, 185
 - ~BoxShape, 187
 - BoxShape, 187
 - FillMsg, 187
 - GetSize, 187
 - Init, 187
 - ProcessMsg, 187
 - SetScale, 187
 - SetSize, 188
- gazebo::physics::Collision, 235
 - ~Collision, 238
 - AddContact, 238
 - Collision, 238
 - FillMsg, 238
 - Fin, 239
 - GetBoundingBox, 239
 - GetContactsEnabled, 239
 - GetLaserRetro, 239
 - GetLink, 239
 - GetMaxContacts, 239
 - GetModel, 240
 - GetRelativeAngularAccel, 240
 - GetRelativeAngularVel, 240
 - GetRelativeLinearAccel, 240
 - GetRelativeLinearVel, 240
 - GetShape, 241
 - GetShapeType, 241
 - GetState, 241
 - GetSurface, 241
 - GetWorldAngularAccel, 241
 - GetWorldAngularVel, 242
 - GetWorldLinearAccel, 242
 - GetWorldLinearVel, 242
 - Init, 242
 - IsPlaceable, 242
 - link, 245
 - Load, 243
 - placeable, 245
 - ProcessMsg, 243
 - SetCategoryBits, 243
 - SetCollideBits, 243
 - SetCollision, 243
 - SetContactsEnabled, 243
 - SetLaserRetro, 244
 - SetMaxContacts, 244
 - SetScale, 244
 - SetShape, 244
 - SetState, 244
 - shape, 245
 - surface, 245
 - UpdateParameters, 245
- gazebo::physics::CollisionState, 245
 - ~CollisionState, 247
 - CollisionState, 247
 - FillSDF, 247
 - GetPose, 247
 - IsZero, 248
 - Load, 248
 - operator<<, 249
 - operator+, 248

- operator-, 248
- operator=, 249
- gazebo::physics::Contact, 279
 - ~Contact, 280
 - collision1, 281
 - collision2, 281
 - Contact, 280
 - count, 281
 - DebugString, 280
 - depths, 281
 - FillMsg, 280
 - normals, 282
 - operator=, 281
 - positions, 282
 - Reset, 281
 - time, 282
 - world, 282
 - wrench, 282
- gazebo::physics::ContactManager, 282
 - ~ContactManager, 283
 - Clear, 283
 - ContactManager, 283
 - CreateFilter, 283, 284
 - GetContact, 284
 - GetContactCount, 284
 - GetContacts, 284
 - GetFilterCount, 285
 - HasFilter, 285
 - Init, 285
 - NewContact, 285
 - PublishContacts, 285
 - RemoveFilter, 285
 - ResetCount, 286
- gazebo::physics::ContactPublisher, 287
 - collisionNames, 287
 - collisions, 287
 - contacts, 287
 - publisher, 287
- gazebo::physics::CylinderShape, 298
 - ~CylinderShape, 299
 - CylinderShape, 299
 - FillMsg, 299
 - GetLength, 299
 - GetRadius, 299
 - Init, 300
 - ProcessMsg, 300
 - SetLength, 300
 - SetRadius, 300
 - SetScale, 300
 - SetSize, 300
- gazebo::physics::DARTBallJoint, 301
 - ~DARTBallJoint, 303
 - DARTBallJoint, 303
 - dtBallJoint, 307
- GetAnchor, 304
- GetAngleImpl, 304
- GetGlobalAxis, 304
- GetHighStop, 304
- GetLowStop, 305
- GetMaxForce, 305
- GetVelocity, 305
- Init, 306
- Load, 306
- SetAxis, 306
- SetForceImpl, 306
- SetHighStop, 306
- SetLowStop, 307
- SetMaxForce, 307
- SetVelocity, 307
- gazebo::physics::DARTBoxShape, 308
 - ~DARTBoxShape, 309
 - DARTBoxShape, 309
 - SetSize, 309
- gazebo::physics::DARTCollision, 309
 - ~DARTCollision, 311
 - DARTCollision, 311
 - Fini, 311
 - GetBoundingBox, 311
 - GetCategoryBits, 311
 - GetCollideBits, 312
 - GetDARTBodyNode, 312
 - GetDARTCollisionShape, 312
 - Init, 312
 - Load, 312
 - OnPoseChange, 312
 - SetCategoryBits, 312
 - SetCollideBits, 313
 - SetDARTCollisionShape, 313
- gazebo::physics::DARTCylinderShape, 313
 - ~DARTCylinderShape, 315
 - DARTCylinderShape, 314
 - SetSize, 315
- gazebo::physics::DARTHeightmapShape, 315
 - ~DARTHeightmapShape, 317
 - DARTHeightmapShape, 316
 - Init, 317
- gazebo::physics::DARTHinge2Joint, 317
 - ~DARTHinge2Joint, 319
 - DARTHinge2Joint, 319
 - dtUniversalJoint, 322
 - GetAnchor, 319
 - GetAngleImpl, 320
 - GetGlobalAxis, 320
 - GetMaxForce, 320
 - GetVelocity, 321
 - Init, 321
 - Load, 321
 - SetAxis, 321

- SetForceImpl, 321
- SetMaxForce, 322
- SetVelocity, 322
- gazebo::physics::DARTHingeJoint, 322
 - ~DARTHingeJoint, 324
 - DARTHingeJoint, 324
 - dtRevoluteJoint, 327
 - GetAnchor, 324
 - GetAngleImpl, 325
 - GetGlobalAxis, 325
 - GetMaxForce, 325
 - GetVelocity, 325
 - Init, 326
 - Load, 326
 - SetAxis, 326
 - SetForceImpl, 326
 - SetMaxForce, 327
 - SetVelocity, 327
- gazebo::physics::DARTJoint, 327
 - ~DARTJoint, 330
 - ApplyDamping, 330
 - AreConnected, 330
 - Attach, 330
 - DARTJoint, 330
 - dartPhysicsEngine, 337
 - Detach, 331
 - dtChildBodyNode, 337
 - dtJoint, 337
 - GetAngleCount, 331
 - GetAttribute, 331
 - GetDARTJoint, 331
 - GetDARTModel, 331
 - GetForce, 332
 - GetForceTorque, 332
 - GetHighStop, 332
 - GetJointLink, 333
 - GetLinkForce, 333
 - GetLinkTorque, 333
 - GetLowStop, 333
 - GetParam, 334
 - Init, 334
 - Load, 334
 - Reset, 335
 - SetAnchor, 335
 - SetAttribute, 335
 - SetDamping, 335
 - SetForce, 335
 - SetForceImpl, 336
 - SetHighStop, 336
 - SetLowStop, 336
 - SetParam, 336
 - SetStiffness, 337
 - SetStiffnessDamping, 337
- gazebo::physics::DARTLink, 338
 - ~DARTLink, 340
 - AddDARTChildJoint, 341
 - AddForce, 341
 - AddForceAtRelativePosition, 341
 - AddForceAtWorldPosition, 341
 - AddRelativeForce, 341
 - AddRelativeTorque, 342
 - AddTorque, 342
 - DARTLink, 340
 - Fini, 342
 - GetDARTBodyNode, 342
 - GetDARTModel, 342
 - GetDARTPhysics, 342
 - GetDARTWorld, 342
 - GetEnabled, 343
 - GetGravityMode, 343
 - GetKinematic, 343
 - GetWorldAngularVel, 343
 - GetWorldCoGLinearVel, 343
 - GetWorldForce, 344
 - GetWorldLinearVel, 344
 - GetWorldTorque, 344
 - Init, 345
 - Load, 345
 - OnPoseChange, 345
 - SetAngularDamping, 345
 - SetAngularVel, 345
 - SetAutoDisable, 345
 - SetDARTParentJoint, 346
 - SetEnabled, 346
 - SetForce, 346
 - SetGravityMode, 346
 - SetKinematic, 346
 - SetLinearDamping, 347
 - SetLinearVel, 347
 - SetLinkStatic, 347
 - SetSelfCollide, 347
 - SetTorque, 347
 - updateDirtyPoseFromDARTTransformation, 348
- gazebo::physics::DARTMeshShape, 348
 - ~DARTMeshShape, 349
 - DARTMeshShape, 349
 - Init, 349
 - Load, 349
 - Update, 349
- gazebo::physics::DARTModel, 350
 - ~DARTModel, 351
 - BackupState, 351
 - DARTModel, 351
 - dtConfig, 352
 - dtSkeleton, 352
 - dtVelocity, 352
 - Fini, 351
 - GetDARTPhysics, 351

- GetDARTSkeleton, 351
- GetDARTWorld, 352
- Init, 352
- Load, 352
- RestoreState, 352
- Update, 352
- gazebo::physics::DARTMultiRayShape, 352
- gazebo::physics::DARTPhysics, 354
 - ~DARTPhysics, 356
 - CreateCollision, 356
 - CreateJoint, 356
 - CreateLink, 356
 - CreateModel, 357
 - CreateShape, 357
 - DARTParam, 356
 - DARTPhysics, 356
 - DebugPrint, 357
 - Fini, 357
 - GetDARTWorld, 357
 - GetParam, 357, 358
 - GetType, 358
 - Init, 358
 - InitForThread, 358
 - Load, 358
 - MAX_CONTACTS, 356
 - MIN_STEP_SIZE, 356
 - OnPhysicsMsg, 359
 - OnRequest, 359
 - Reset, 359
 - SetGravity, 359
 - SetParam, 359
 - SetSeed, 360
 - UpdateCollision, 360
 - UpdatePhysics, 361
- gazebo::physics::DARTPlaneShape, 361
 - ~DARTPlaneShape, 362
 - CreatePlane, 362
 - DARTPlaneShape, 362
 - SetAltitude, 362
- gazebo::physics::DARTRayShape, 363
- gazebo::physics::DARTScrewJoint, 364
 - ~DARTScrewJoint, 366
 - DARTScrewJoint, 366
 - dartScrewJoint, 371
 - GetAnchor, 366
 - GetAngleImpl, 366
 - GetGlobalAxis, 366
 - GetHighStop, 367
 - GetLowStop, 367
 - GetMaxForce, 367
 - GetThreadPitch, 368
 - GetVelocity, 368
 - Init, 369
 - Load, 369
- SetAnchor, 369
- SetAxis, 369
- SetForceImpl, 369
- SetMaxForce, 370
- SetThreadPitch, 370
- SetVelocity, 370
- gazebo::physics::DARTSliderJoint, 371
 - ~DARTSliderJoint, 373
 - DARTSliderJoint, 373
 - dtPrismaticJoint, 375
 - GetAnchor, 373
 - GetAngleImpl, 373
 - GetGlobalAxis, 373
 - GetMaxForce, 374
 - GetVelocity, 374
 - Init, 374
 - Load, 374
 - SetAxis, 374
 - SetForceImpl, 375
 - SetMaxForce, 375
 - SetVelocity, 375
- gazebo::physics::DARTSphereShape, 376
 - ~DARTSphereShape, 377
 - DARTSphereShape, 377
 - SetRadius, 377
- gazebo::physics::DARTTypes, 377
- gazebo::physics::DARTUniversalJoint, 378
 - ~DARTUniversalJoint, 380
 - DARTUniversalJoint, 380
 - dtUniversalJoint, 383
 - GetAnchor, 380
 - GetAngleImpl, 380
 - GetGlobalAxis, 380
 - GetMaxForce, 381
 - GetVelocity, 381
 - Init, 381
 - Load, 381
 - SetAxis, 382
 - SetForceImpl, 382
 - SetMaxForce, 382
 - SetVelocity, 382
- gazebo::physics::Entity, 404
 - ~Entity, 407
 - animation, 414
 - animationConnection, 414
 - animationStartPose, 414
 - connections, 415
 - dirtyPose, 415
 - Entity, 407
 - Fini, 407
 - GetBoundingBox, 407
 - GetChildCollision, 408
 - GetChildLink, 408
 - GetCollisionBoundingBox, 408

- GetDirtyPose, 408
- GetInitialRelativePose, 408
- GetNearestEntityBelow, 409
- GetParentModel, 409
- GetRelativeAngularAccel, 409
- GetRelativeAngularVel, 409
- GetRelativeLinearAccel, 409
- GetRelativeLinearVel, 410
- GetRelativePose, 410
- GetWorldAngularAccel, 410
- GetWorldAngularVel, 410
- GetWorldLinearAccel, 410
- GetWorldLinearVel, 411
- GetWorldPose, 411
- IsCanonicalLink, 411
- IsStatic, 411
- Load, 411
- node, 415
- OnPoseChange, 412
- parentEntity, 415
- PlaceOnEntity, 412
- PlaceOnNearestEntityBelow, 412
- prevAnimationTime, 415
- requestPub, 415
- Reset, 412
- scale, 415
- SetAnimation, 412
- SetCanonicalLink, 413
- SetInitialRelativePose, 413
- SetName, 413
- SetRelativePose, 413
- SetStatic, 413
- SetWorldPose, 413
- SetWorldTwist, 414
- StopAnimation, 414
- UpdateParameters, 414
- visPub, 415
- visualMsg, 415
- gazebo::physics::FrictionPyramid, 455
 - ~FrictionPyramid, 456
 - direction1, 457
 - FrictionPyramid, 456
 - GetMuPrimary, 456
 - GetMuSecondary, 456
 - SetMuPrimary, 456
 - SetMuSecondary, 457
- gazebo::physics::GearboxJoint
 - ~GearboxJoint, 463
 - gearRatio, 464
 - GearboxJoint, 462
 - GetAngleCount, 463
 - GetGearboxRatio, 463
 - Init, 463
 - Load, 463
 - referenceBody, 464
 - SetGearboxRatio, 463
- gazebo::physics::GearboxJoint< T >, 462
- gazebo::physics::Gripper, 492
 - ~Gripper, 493
 - GetName, 493
 - Gripper, 493
 - Init, 493
 - IsAttached, 493
 - Load, 493
 - node, 493
- gazebo::physics::HeightmapShape, 506
 - ~HeightmapShape, 508
 - FillMsg, 509
 - flipY, 511
 - GetHeight, 509
 - GetImage, 509
 - GetMaxHeight, 509
 - GetMinHeight, 509
 - GetPos, 509
 - GetSize, 510
 - GetSubSampling, 510
 - GetURI, 510
 - GetVertexCount, 510
 - heightmapData, 511
 - HeightmapShape, 508
 - heights, 511
 - img, 511
 - Init, 510
 - Load, 510
 - ProcessMsg, 511
 - SetScale, 511
 - subSampling, 511
 - vertSize, 512
- gazebo::physics::Hinge2Joint
 - ~Hinge2Joint, 512
 - GetAngleCount, 513
 - Hinge2Joint, 512
 - Load, 513
- gazebo::physics::Hinge2Joint< T >, 512
- gazebo::physics::HingeJoint
 - ~HingeJoint, 514
 - GetAngleCount, 514
 - HingeJoint, 514
 - Init, 514
 - Load, 514
- gazebo::physics::HingeJoint< T >, 513
- gazebo::physics::Inertial, 529
 - ~Inertial, 531
 - GetCoG, 532
 - GetIxx, 532
 - GetIxy, 532
 - GetIxz, 532
 - GetIyy, 532

- GetIYZ, 533
- GetIZZ, 533
- GetInertial, 532
- GetMOI, 533
- GetMass, 533
- GetPose, 534
- GetPrincipalMoments, 534
- GetProductsofInertia, 534
- Inertial, 531
- Load, 534
- operator<<, 538
- operator+, 534
- operator+=", 535
- operator=, 535
- ProcessMsg, 535
- Reset, 535
- Rotate, 535
- SetCoG, 535, 536
- SetIXX, 537
- SetIXY, 537
- SetIXZ, 537
- SetIYY, 537
- SetIYZ, 537
- SetIZZ, 537
- SetInertiaMatrix, 536
- SetMOI, 538
- SetMass, 538
- UpdateParameters, 538
- gazebo::physics::Joint, 541
 - ~Joint, 547
 - anchorLink, 566
 - anchorPos, 566
 - anchorPose, 566
 - ApplyDamping, 547
 - applyDamping, 566
 - ApplyStiffnessDamping, 547
 - AreConnected, 548
 - Attach, 548
 - Attribute, 547
 - axisParentModelFrame, 566
 - CFM, 547
 - CacheForceTorque, 548
 - CheckAndTruncateForce, 548
 - childLink, 566
 - ConnectJointUpdate, 548
 - dampingCoefficient, 567
 - Detach, 549
 - DisconnectJointUpdate, 549
 - dissipationCoefficient, 567
 - ERP, 547
 - effortLimit, 567
 - FMAX, 547
 - FUDGE_FACTOR, 547
 - FillMsg, 549
 - Fini, 549
 - GetAnchor, 549
 - GetAnchorErrorPose, 550
 - GetAngle, 550
 - GetAngleCount, 550
 - GetAngleImpl, 550
 - GetAttribute, 551
 - GetAxisFrame, 551
 - GetChild, 551
 - GetDamping, 551
 - GetDampingCoefficient, 552
 - GetEffortLimit, 552
 - GetForce, 552
 - GetForceTorque, 552
 - GetGlobalAxis, 553
 - GetHighStop, 553
 - GetInertiaRatio, 554
 - GetInitialAnchorPose, 554
 - GetJointLink, 554
 - GetLinkForce, 555
 - GetLinkTorque, 555
 - GetLocalAxis, 555
 - GetLowStop, 556
 - GetLowerLimit, 555
 - GetMaxForce, 556
 - GetParam, 556
 - GetParent, 557
 - GetParentWorldPose, 557
 - GetSpringReferencePosition, 557
 - GetStiffness, 557
 - GetStopDissipation, 558
 - GetStopStiffness, 558
 - GetUpperLimit, 558
 - GetVelocity, 558
 - GetVelocityLimit, 559
 - GetWorldEnergyPotentialSpring, 559
 - GetWorldPose, 559
 - HI_STOP, 547
 - Init, 559
 - Joint, 547
 - LO_STOP, 547
 - Load, 560
 - lowerLimit, 567
 - model, 567
 - parentAnchorPose, 567
 - parentLink, 567
 - provideFeedback, 567
 - Reset, 560
 - STOP_CFM, 547
 - STOP_ERP, 547
 - SUSPENSION_CFM, 547
 - SUSPENSION_ERP, 547
 - SetAnchor, 560
 - SetAngle, 561

- SetAttribute, 561
- SetAxis, 561
- SetDamping, 562
- SetEffortLimit, 562
- SetForce, 562
- SetHighStop, 562
- SetLowStop, 563
- SetLowerLimit, 563
- SetMaxForce, 563
- SetModel, 563
- SetParam, 564
- SetProvideFeedback, 564
- SetState, 564
- SetStiffness, 564
- SetStiffnessDamping, 564
- SetStopDissipation, 565
- SetStopStiffness, 565
- SetUpperLimit, 565
- SetVelocity, 565
- springReferencePosition, 567
- stiffnessCoefficient, 567
- Update, 566
- UpdateParameters, 566
- upperLimit, 567
- useCFMDamping, 568
- VEL, 547
- velocityLimit, 568
- wrench, 568
- gazebo::physics::JointController, 568
 - ~JointController, 569
 - AddJoint, 569
 - GetForces, 569
 - GetJoints, 570
 - GetLastUpdateTime, 570
 - GetPositionPIDs, 570
 - GetPositions, 570
 - GetVelocities, 570
 - GetVelocityPIDs, 570
 - JointController, 569
 - Reset, 571
 - SetJointPosition, 571
 - SetJointPositions, 571
 - SetPositionTarget, 571
 - SetVelocityTarget, 572
 - Update, 572
- gazebo::physics::JointControllerPrivate, 572
 - forces, 573
 - jointCmdSub, 573
 - joints, 573
 - model, 573
 - node, 573
 - posPids, 573
 - positions, 573
 - prevUpdateTime, 573
 - updatedLinks, 573
 - velPids, 574
 - velocities, 574
- gazebo::physics::JointState, 574
 - ~JointState, 576
 - FillSDF, 576
 - GetAngle, 576
 - GetAngleCount, 576
 - GetAngles, 577
 - IsZero, 577
 - JointState, 575, 576
 - Load, 577
 - operator<<, 578
 - operator+, 577
 - operator-, 578
 - operator=, 578
- gazebo::physics::JointWrench, 581
 - body1Force, 582
 - body1Torque, 582
 - body2Force, 583
 - body2Torque, 583
 - operator+, 582
 - operator-, 582
 - operator=, 582
- gazebo::physics::Link, 595
 - ~Link, 601
 - AddChildJoint, 601
 - AddForce, 601
 - AddForceAtRelativePosition, 601
 - AddForceAtWorldPosition, 601
 - AddParentJoint, 602
 - AddRelativeForce, 602
 - AddRelativeTorque, 602
 - AddTorque, 602
 - angularAccel, 617
 - AttachStaticModel, 602
 - attachedModelsOffset, 617
 - cgVisuals, 617
 - ConnectEnabled, 603
 - DetachAllStaticModels, 603
 - DetachStaticModel, 603
 - DisconnectEnabled, 603
 - FillMsg, 603
 - Fini, 604
 - GetAngularDamping, 604
 - GetBoundingBox, 604
 - GetChildJoints, 604
 - GetChildJointsLinks, 604
 - GetCollision, 604, 605
 - GetCollisionById, 605
 - GetCollisions, 605
 - GetEnabled, 605
 - GetGravityMode, 605
 - GetInertial, 606

GetKinematic, 606
 GetLinearDamping, 606
 GetModel, 606
 GetParentJoints, 606
 GetParentJointsLinks, 606
 GetRelativeAngularAccel, 607
 GetRelativeAngularVel, 607
 GetRelativeForce, 607
 GetRelativeLinearAccel, 607
 GetRelativeLinearVel, 607
 GetRelativeTorque, 607
 GetSelfCollide, 608
 GetSensorCount, 608
 GetSensorName, 608
 GetWorldAngularAccel, 608
 GetWorldCoGLinearVel, 609
 GetWorldCoGPose, 609
 GetWorldEnergy, 609
 GetWorldEnergyKinetic, 609
 GetWorldEnergyPotential, 609
 GetWorldForce, 609
 GetWorldInertiaMatrix, 610
 GetWorldInertialPose, 610
 GetWorldLinearAccel, 610
 GetWorldLinearVel, 610, 611
 GetWorldTorque, 611
 inertial, 617
 Init, 611
 initialized, 618
 linearAccel, 618
 Link, 601
 Load, 611
 OnPoseChange, 612
 ProcessMsg, 612
 RemoveChild, 612
 RemoveChildJoint, 612
 RemoveCollision, 612
 RemoveParentJoint, 612
 Reset, 613
 ResetPhysicsStates, 613
 SetAngularAccel, 613
 SetAngularDamping, 613
 SetAngularVel, 613
 SetAutoDisable, 613
 SetCollideMode, 614
 SetEnabled, 614
 SetForce, 614
 SetGravityMode, 614
 SetInertial, 614
 SetKinematic, 614
 SetLaserRetro, 615
 SetLinearAccel, 615
 SetLinearDamping, 615
 SetLinearVel, 615
 SetLinkStatic, 615
 SetPublishData, 616
 SetScale, 616
 SetSelected, 616
 SetSelfCollide, 616
 SetState, 616
 SetTorque, 616
 Update, 617
 UpdateMass, 617
 UpdateParameters, 617
 UpdateSurface, 617
 visuals, 618
 Visuals_M, 601
 gazebo::physics::LinkState, 618
 ~LinkState, 620
 FillSDF, 621
 GetAcceleration, 621
 GetCollisionState, 621
 GetCollisionStateCount, 622
 GetCollisionStates, 622
 GetPose, 622
 GetVelocity, 622
 GetWrench, 622
 IsZero, 622
 LinkState, 620
 Load, 623
 operator<<, 625
 operator+, 623
 operator-, 623
 operator=, 624
 SetRealTime, 624
 SetSimTime, 624
 SetWallTime, 624
 gazebo::physics::MeshShape, 675
 ~MeshShape, 676
 FillMsg, 676
 GetMeshURI, 677
 GetSize, 677
 Init, 677
 mesh, 678
 MeshShape, 676
 ProcessMsg, 677
 SetMesh, 677
 SetScale, 677
 submesh, 678
 Update, 678
 gazebo::physics::Model, 678
 ~Model, 682
 AttachStaticModel, 682
 attachedModels, 693
 attachedModelsOffset, 693
 DetachStaticModel, 682
 FillMsg, 683
 Fini, 683

- GetAutoDisable, 683
- GetBoundingBox, 683
- GetGripper, 683
- GetGripperCount, 684
- GetJoint, 684
- GetJointController, 684
- GetJointCount, 684
- GetJoints, 684
- GetLink, 684
- GetLinkById, 685
- GetLinks, 685
- GetPluginCount, 685
- GetRelativeAngularAccel, 685
- GetRelativeAngularVel, 685
- GetRelativeLinearAccel, 686
- GetRelativeLinearVel, 686
- GetSDF, 686
- GetSensorCount, 686
- GetWorldAngularAccel, 686
- GetWorldAngularVel, 687
- GetWorldEnergy, 687
- GetWorldEnergyKinetic, 687
- GetWorldEnergyPotential, 687
- GetWorldLinearAccel, 687
- GetWorldLinearVel, 688
- Init, 688
- jointPub, 693
- Load, 688
- LoadJoints, 688
- LoadPlugins, 688
- Model, 682
- OnPoseChange, 688
- ProcessMsg, 689
- RemoveChild, 689
- Reset, 689
- SetAngularAccel, 689
- SetAngularVel, 689
- SetAutoDisable, 689
- SetCollideMode, 690
- SetEnabled, 690
- SetGravityMode, 690
- SetJointAnimation, 690
- SetJointPosition, 690
- SetJointPositions, 691
- SetLaserRetro, 691
- SetLinearAccel, 691
- SetLinearVel, 691
- SetLinkWorldPose, 691, 692
- SetScale, 692
- SetState, 692
- StopAnimation, 692
- Update, 692
- UpdateParameters, 692
- gazebo::physics::ModelState, 698
- ~ModelState, 700
- FillSDF, 701
- GetJointState, 701
- GetJointStateCount, 701
- GetJointStates, 702
- GetLinkState, 702
- GetLinkStateCount, 702
- GetLinkStates, 703
- GetPose, 703
- HasJointState, 703
- HasLinkState, 703
- IsZero, 704
- Load, 704
- ModelState, 700
- operator<<, 706
- operator+, 704
- operator-, 705
- operator=, 705
- SetRealTime, 705
- SetSimTime, 705
- SetWallTime, 706
- gazebo::physics::MultiRayShape, 723
- ~MultiRayShape, 726
- AddRay, 726
- ConnectNewLaserScans, 726
- DisconnectNewLaserScans, 727
- FillMsg, 727
- GetFiducial, 727
- GetMaxAngle, 727
- GetMaxRange, 727
- GetMinAngle, 728
- GetMinRange, 728
- GetRange, 728
- GetResRange, 728
- GetRetro, 728
- GetSampleCount, 729
- GetScanResolution, 729
- GetVerticalMaxAngle, 729
- GetVerticalMinAngle, 729
- GetVerticalSampleCount, 729
- GetVerticalScanResolution, 729
- horzElem, 730
- Init, 729
- MultiRayShape, 726
- newLaserScans, 730
- offset, 730
- ProcessMsg, 730
- rangeElem, 731
- rayElem, 731
- rays, 731
- scanElem, 731
- SetScale, 730
- Update, 730
- UpdateRays, 730

- vertElem, 731
- gazebo::physics::PhysicsEngine, 766
 - ~PhysicsEngine, 770
 - contactManager, 780
 - CreateCollision, 770
 - CreateJoint, 771
 - CreateLink, 771
 - CreateModel, 771
 - CreateShape, 771
 - DebugPrint, 771
 - Finis, 772
 - GetAutoDisableFlag, 772
 - GetContactManager, 772
 - GetContactMaxCorrectingVel, 772
 - GetContactSurfaceLayer, 772
 - GetGravity, 772
 - GetMaxContacts, 773
 - GetMaxStepSize, 773
 - GetParam, 773
 - GetPhysicsUpdateMutex, 773
 - GetRealTimeUpdateRate, 773
 - GetSORPGSIlters, 774
 - GetSORPGSPreconlters, 774
 - GetSORPGSW, 774
 - GetTargetRealTimeFactor, 774
 - GetType, 774
 - GetUpdatePeriod, 774
 - GetWorldCFM, 775
 - GetWorldERP, 775
 - Init, 775
 - InitForThread, 775
 - Load, 775
 - maxStepSize, 780
 - node, 780
 - OnPhysicsMsg, 775
 - OnRequest, 776
 - PhysicsEngine, 770
 - physicsSub, 780
 - physicsUpdateMutex, 780
 - realTimeUpdateRate, 780
 - requestSub, 780
 - Reset, 776
 - responsePub, 781
 - sdf, 781
 - SetAutoDisableFlag, 776
 - SetContactMaxCorrectingVel, 776
 - SetContactSurfaceLayer, 776
 - SetGravity, 777
 - SetMaxContacts, 777
 - SetMaxStepSize, 777
 - SetParam, 777
 - SetRealTimeUpdateRate, 778
 - SetSORPGSIlters, 778
 - SetSORPGSPreconlters, 779
 - SetSORPGSW, 779
 - SetSeed, 778
 - SetTargetRealTimeFactor, 779
 - SetWorldCFM, 779
 - SetWorldERP, 779
 - targetRealTimeFactor, 781
 - UpdateCollision, 780
 - UpdatePhysics, 780
 - world, 781
- gazebo::physics::PhysicsFactory, 781
 - IsRegistered, 782
 - NewPhysicsEngine, 782
 - RegisterAll, 782
 - RegisterPhysicsEngine, 782
- gazebo::physics::PlaneShape, 791
 - ~PlaneShape, 792
 - CreatePlane, 792
 - FillMsg, 792
 - GetNormal, 792
 - GetSize, 793
 - Init, 793
 - PlaneShape, 792
 - ProcessMsg, 793
 - SetAltitude, 793
 - SetNormal, 793
 - SetScale, 794
 - SetSize, 794
- gazebo::physics::RayShape, 849
 - ~RayShape, 851
 - contactFiducial, 854
 - contactLen, 854
 - contactRetro, 854
 - FillMsg, 851
 - GetFiducial, 851
 - GetGlobalPoints, 852
 - GetIntersection, 852
 - GetLength, 852
 - GetRelativePoints, 852
 - GetRetro, 852
 - globalEndPos, 854
 - globalStartPos, 854
 - Init, 853
 - ProcessMsg, 853
 - RayShape, 851
 - relativeEndPos, 854
 - relativeStartPos, 854
 - SetFiducial, 853
 - SetLength, 853
 - SetPoints, 853
 - SetRetro, 854
 - SetScale, 854
 - Update, 854
- gazebo::physics::Road, 869
 - ~Road, 870

- Init, 870
- Load, 870
- Road, 870
- gazebo::physics::ScrewJoint
 - ~ScrewJoint, 897
 - GetAngleCount, 897
 - GetThreadPitch, 898
 - Init, 898
 - Load, 898
 - ScrewJoint, 897
 - SetThreadPitch, 898, 899
 - threadPitch, 899
- gazebo::physics::ScrewJoint< T >, 896
- gazebo::physics::Shape, 932
 - ~Shape, 934
 - collisionParent, 935
 - FillMsg, 934
 - GetScale, 934
 - Init, 935
 - ProcessMsg, 935
 - scale, 935
 - SetScale, 935
 - Shape, 934
- gazebo::physics::SimbodyBallJoint, 936
 - ~SimbodyBallJoint, 937
 - GetAnchor, 938
 - GetAngleImpl, 938
 - GetAxis, 938
 - GetGlobalAxis, 938
 - GetHighStop, 938
 - GetLowStop, 939
 - GetMaxForce, 939
 - GetVelocity, 939
 - Load, 940
 - SetAxis, 940
 - SetForceImpl, 940
 - SetHighStop, 940
 - SetLowStop, 941
 - SetMaxForce, 941
 - SetVelocity, 941
 - SimbodyBallJoint, 937
- gazebo::physics::SimbodyBoxShape, 941
 - ~SimbodyBoxShape, 942
 - SetSize, 943
 - SimbodyBoxShape, 942
- gazebo::physics::SimbodyCollision, 943
 - ~SimbodyCollision, 945
 - GetBoundingBox, 945
 - GetCollisionShape, 945
 - Load, 945
 - OnPoseChange, 945
 - SetCategoryBits, 946
 - SetCollideBits, 946
 - SetCollisionShape, 946
 - SimbodyCollision, 945
- gazebo::physics::SimbodyCylinderShape, 946
 - ~SimbodyCylinderShape, 947
 - SetSize, 948
 - SimbodyCylinderShape, 947
- gazebo::physics::SimbodyHeightmapShape, 948
 - ~SimbodyHeightmapShape, 949
 - Init, 950
 - SimbodyHeightmapShape, 949
- gazebo::physics::SimbodyHinge2Joint, 950
 - ~SimbodyHinge2Joint, 952
 - GetAnchor, 952
 - GetAngleImpl, 952
 - GetAxis, 953
 - GetGlobalAxis, 953
 - GetMaxForce, 953
 - GetVelocity, 953
 - Load, 954
 - SetAxis, 954
 - SetForceImpl, 954
 - SetMaxForce, 954
 - SetVelocity, 954
 - SimbodyHinge2Joint, 952
- gazebo::physics::SimbodyHingeJoint, 955
 - ~SimbodyHingeJoint, 957
 - GetAngleImpl, 957
 - GetGlobalAxis, 957
 - GetMaxForce, 958
 - GetVelocity, 958
 - Load, 958
 - RestoreSimbodyState, 959
 - SaveSimbodyState, 959
 - SetAxis, 959
 - SetForceImpl, 959
 - SetMaxForce, 959
 - SetVelocity, 960
 - SimbodyHingeJoint, 957
- gazebo::physics::SimbodyJoint, 960
 - ~SimbodyJoint, 962
 - AreConnected, 963
 - CacheForceTorque, 963
 - constraint, 970
 - damper, 970
 - defxAB, 970
 - Detach, 963
 - GetAnchor, 963
 - GetAttribute, 963
 - GetForce, 964
 - GetForceTorque, 964
 - GetHighStop, 964
 - GetJointLink, 965
 - GetLinkForce, 965
 - GetLinkTorque, 965
 - GetLowStop, 966

- GetParam, 966
- isReversed, 970
- limitForce, 970
- Load, 966
- mobod, 971
- mustBreakLoopHere, 971
- physicsInitialized, 971
- Reset, 967
- RestoreSimbodyState, 967
- SaveSimbodyState, 967
- SetAnchor, 967
- SetAttribute, 967
- SetAxis, 968
- SetDamping, 968
- SetForce, 968
- SetForceImpl, 968
- SetHighStop, 969
- SetLowStop, 969
- SetParam, 969
- SetStiffness, 969
- SetStiffnessDamping, 970
- SimbodyJoint, 962
- simbodyPhysics, 971
- spring, 971
- world, 971
- xCB, 971
- xPA, 971
- gazebo::physics::SimbodyLink, 972
 - ~SimbodyLink, 974
 - AddForce, 974
 - AddForceAtRelativePosition, 974
 - AddForceAtWorldPosition, 975
 - AddRelativeForce, 975
 - AddRelativeTorque, 975
 - AddTorque, 975
 - Fini, 975
 - GetEffectiveMassProps, 976
 - GetEnabled, 976
 - GetGravityMode, 976
 - GetMassProperties, 976
 - GetWorldAngularVel, 976
 - GetWorldCoGLinearVel, 976
 - GetWorldForce, 976
 - GetWorldLinearVel, 977
 - GetWorldTorque, 977
 - Init, 977
 - Load, 978
 - masterMobod, 980
 - mustBeBaseLink, 980
 - OnPoseChange, 978
 - physicsInitialized, 980
 - RestoreSimbodyState, 978
 - SaveSimbodyState, 978
 - SetAngularDamping, 978
 - SetAngularVel, 978
 - SetAutoDisable, 978
 - SetDirtyPose, 979
 - SetEnabled, 979
 - SetForce, 979
 - SetGravityMode, 979
 - SetLinearDamping, 979
 - SetLinearVel, 979
 - SetLinkStatic, 980
 - SetSelfCollide, 980
 - SetTorque, 980
 - SimbodyLink, 974
 - slaveMobods, 980
 - slaveWelds, 980
 - gazebo::physics::SimbodyMeshShape, 981
 - ~SimbodyMeshShape, 982
 - Init, 982
 - Load, 982
 - SimbodyMeshShape, 982
 - gazebo::physics::SimbodyModel, 982
 - ~SimbodyModel, 984
 - Init, 984
 - Load, 984
 - SimbodyModel, 983
 - gazebo::physics::SimbodyMultiRayShape, 984
 - ~SimbodyMultiRayShape, 986
 - AddRay, 986
 - SimbodyMultiRayShape, 986
 - UpdateRays, 986
 - gazebo::physics::SimbodyPhysics, 986
 - ~SimbodyPhysics, 989
 - contact, 996
 - CreateCollision, 989
 - CreateJoint, 989
 - CreateLink, 990
 - CreateModel, 990
 - CreateShape, 990
 - DebugPrint, 990
 - discreteForces, 996
 - Fini, 990
 - forces, 996
 - GetDynamicsWorld, 990
 - GetParam, 991
 - GetPose, 991
 - GetType, 991
 - GetTypeString, 991
 - gravity, 997
 - Init, 992
 - InitForThread, 992
 - InitModel, 992
 - integ, 997
 - Load, 992
 - matter, 997
 - OnPhysicsMsg, 992

- OnRequest, 993
- Pose2Transform, 993
- QuadToQuad, 993
- Reset, 994
- SetGravity, 994
- SetParam, 994
- SetSeed, 995
- SimbodyPhysics, 989
- simbodyPhysicsInitialized, 997
- simbodyPhysicsStepped, 997
- system, 997
- tracker, 997
- Transform2Pose, 995
- UpdateCollision, 996
- UpdatePhysics, 996
- Vec3ToVector3, 996
- Vector3ToVec3, 996
- gazebo::physics::SimbodyPlaneShape, 997
 - ~SimbodyPlaneShape, 998
 - CreatePlane, 999
 - SetAltitude, 999
 - SimbodyPlaneShape, 998
- gazebo::physics::SimbodyRayShape, 999
 - ~SimbodyRayShape, 1001
 - GetIntersection, 1001
 - SetPoints, 1001
 - SimbodyRayShape, 1001
 - Update, 1001
- gazebo::physics::SimbodyScrewJoint, 1002
 - ~SimbodyScrewJoint, 1004
 - GetAngleImpl, 1004
 - GetAttribute, 1004
 - GetGlobalAxis, 1004
 - GetHighStop, 1005
 - GetLowStop, 1005
 - GetMaxForce, 1005
 - GetParam, 1006
 - GetThreadPitch, 1006
 - GetVelocity, 1006
 - Load, 1007
 - SetAttribute, 1007
 - SetAxis, 1007
 - SetForceImpl, 1007
 - SetHighStop, 1008
 - SetLowStop, 1008
 - SetMaxForce, 1008
 - SetParam, 1008
 - SetThreadPitch, 1009
 - SetVelocity, 1009
 - SimbodyScrewJoint, 1004
- gazebo::physics::SimbodySliderJoint, 1010
 - ~SimbodySliderJoint, 1011
 - GetAngleImpl, 1011
 - GetGlobalAxis, 1012
 - GetMaxForce, 1012
 - GetVelocity, 1012
 - Load, 1013
 - SetAxis, 1013
 - SetForceImpl, 1013
 - SetMaxForce, 1013
 - SetVelocity, 1014
 - SimbodySliderJoint, 1011
- gazebo::physics::SimbodySphereShape, 1014
 - ~SimbodySphereShape, 1015
 - SetRadius, 1015
 - SimbodySphereShape, 1015
- gazebo::physics::SimbodyUniversalJoint, 1016
 - ~SimbodyUniversalJoint, 1017
 - GetAnchor, 1017
 - GetAngleImpl, 1018
 - GetAxis, 1018
 - GetGlobalAxis, 1018
 - GetMaxForce, 1018
 - GetVelocity, 1019
 - Load, 1019
 - SetAxis, 1019
 - SetForceImpl, 1019
 - SetMaxForce, 1020
 - SetVelocity, 1020
 - SimbodyUniversalJoint, 1017
- gazebo::physics::SliderJoint
 - ~SliderJoint, 1045
 - GetAngleCount, 1045
 - Load, 1045
 - SliderJoint, 1045
- gazebo::physics::SliderJoint< T >, 1044
- gazebo::physics::SphereShape, 1054
 - ~SphereShape, 1056
 - FillMsg, 1056
 - GetRadius, 1056
 - Init, 1056
 - ProcessMsg, 1056
 - SetRadius, 1056
 - SetScale, 1057
 - SphereShape, 1056
- gazebo::physics::State, 1068
 - ~State, 1069
 - GetName, 1070
 - GetRealTime, 1070
 - GetSimTime, 1070
 - GetWallTime, 1070
 - Load, 1070
 - name, 1072
 - operator-, 1070
 - operator=, 1071
 - realTime, 1072
 - SetName, 1071
 - SetRealTime, 1071

- SetSimTime, 1071
- SetWallTime, 1072
- simTime, 1072
- State, 1069
- wallTime, 1072
- gazebo::physics::SurfaceParams, 1090
 - ~SurfaceParams, 1091
 - collideWithoutContact, 1092
 - collideWithoutContactBitmask, 1092
 - FillMsg, 1091
 - Load, 1091
 - ProcessMsg, 1091
 - SurfaceParams, 1091
- gazebo::physics::TrajectoryInfo, 1129
 - duration, 1130
 - endTime, 1130
 - id, 1130
 - startTime, 1130
 - TrajectoryInfo, 1130
 - translated, 1130
 - type, 1130
- gazebo::physics::UniversalJoint
 - ~UniversalJoint, 1136
 - AXIS_CHILD, 1135
 - AXIS_PARENT, 1135
 - AxisIndex, 1135
 - GetAngleCount, 1136
 - Init, 1136
 - Load, 1136
 - UniversalJoint, 1136
- gazebo::physics::UniversalJoint< T >, 1135
- gazebo::physics::World, 1239
 - ~World, 1242
 - Clear, 1242
 - ClearModels, 1242
 - dirtyPoses, 1251
 - DisableAllModels, 1242
 - EnableAllModels, 1242
 - EnablePhysicsEngine, 1242
 - Fini, 1242
 - GetByName, 1243
 - GetEnablePhysicsEngine, 1243
 - GetEntity, 1243
 - GetEntityBelowPoint, 1243
 - GetIterations, 1244
 - GetModel, 1244
 - GetModelBelowPoint, 1244
 - GetModelCount, 1245
 - GetModels, 1245
 - GetName, 1245
 - GetPauseTime, 1245
 - GetPhysicsEngine, 1245
 - GetRealTime, 1245
 - GetRunning, 1246
 - GetSceneMsg, 1246
 - GetSelectedEntity, 1246
 - GetSetWorldPoseMutex, 1246
 - GetSimTime, 1246
 - GetSphericalCoordinates, 1246
 - GetStartTime, 1247
 - Init, 1247
 - InsertModelFile, 1247
 - InsertModelSDF, 1247
 - InsertModelString, 1247
 - IsLoaded, 1247
 - IsPaused, 1248
 - Load, 1248
 - LoadPlugin, 1248
 - PrintEntityTree, 1248
 - PublishModelPose, 1248
 - RemovePlugin, 1249
 - Reset, 1249
 - ResetEntities, 1249
 - ResetTime, 1249
 - Run, 1249
 - RunBlocking, 1249
 - Save, 1250
 - SetPaused, 1250
 - SetSimTime, 1250
 - SetState, 1250
 - Step, 1250
 - StepWorld, 1250
 - Stop, 1251
 - StripWorldName, 1251
 - UpdateStateSDF, 1251
 - World, 1242
- gazebo::physics::WorldState, 1253
 - ~WorldState, 1255
 - FillSDF, 1255
 - GetModelState, 1255
 - GetModelStateCount, 1256
 - GetModelStates, 1256
 - HasModelState, 1256
 - IsZero, 1256
 - Load, 1257
 - operator<<, 1259
 - operator+, 1257
 - operator-, 1257
 - operator=, 1258
 - SetRealTime, 1258
 - SetSimTime, 1258
 - SetWallTime, 1258
 - SetWorld, 1258
 - WorldState, 1255
- gazebo::rendering, 123
 - ArrowVisualPtr, 127
 - AxisVisualPtr, 127
 - COMVisualPtr, 128

- CameraPtr, 127
- CameraVisualPtr, 128
- ContactVisualPtr, 128
- DepthCameraPtr, 128
- DynamicLinesPtr, 128
- GpuLaserPtr, 128
- JointVisualPtr, 128
- LaserVisualPtr, 128
- LightPtr, 128
- RENDERING_LINE_LIST, 128
- RENDERING_LINE_STRIP, 128
- RENDERING_MESH_RESOURCE, 128
- RENDERING_POINT_LIST, 128
- RENDERING_TRIANGLE_FAN, 128
- RENDERING_TRIANGLE_LIST, 128
- RENDERING_TRIANGLE_STRIP, 128
- RFIDTagVisualPtr, 128
- RFIDVisualPtr, 128
- RenderOpType, 128
- ScenePtr, 128
- SelectionObjPtr, 128
- SonarVisualPtr, 128
- UserCameraPtr, 128
- VisualPtr, 128
- WindowManagerPtr, 128
- WrenchVisualPtr, 128
- gazebo::rendering::ArrowVisual, 157
 - ~ArrowVisual, 158
 - ArrowVisual, 158
 - Load, 159
 - ShowRotation, 159
- gazebo::rendering::ArrowVisualPrivate, 159
 - headNode, 160
 - rotationNode, 160
 - shaftNode, 160
- gazebo::rendering::AxisVisual, 163
 - ~AxisVisual, 164
 - AxisVisual, 164
 - Load, 164
 - ScaleXAxis, 164
 - ScaleYAxis, 165
 - ScaleZAxis, 165
 - SetAxisMaterial, 165
 - ShowRotation, 165
- gazebo::rendering::AxisVisualPrivate, 165
 - xAxis, 166
 - yAxis, 166
 - zAxis, 166
- gazebo::rendering::COMVisual, 260
 - ~COMVisual, 261
 - COMVisual, 261
 - Load, 262
- gazebo::rendering::COMVisualPrivate, 262
 - boxNode, 263
 - crossLines, 263
- gazebo::rendering::Camera, 197
 - ~Camera, 204
 - animState, 222
 - AnimationComplete, 204
 - AttachToVisual, 204, 205
 - AttachToVisualImpl, 205
 - bayerFrameBuffer, 222
 - Camera, 204
 - camera, 222
 - captureData, 222
 - captureDataOnce, 222
 - ConnectNewImageFrame, 206
 - connections, 222
 - CreateRenderTexture, 206
 - DisconnectNewImageFrame, 206
 - EnableSaveFrame, 206
 - Fini, 207
 - GetAspectRatio, 207
 - GetAvgFPS, 207
 - GetCameraToViewportRay, 207
 - GetCaptureData, 207
 - GetDirection, 208
 - GetFarClip, 208
 - GetFrameFilename, 208
 - GetHFOV, 208
 - GetImageByteSize, 208
 - GetImageData, 209
 - GetImageDepth, 209
 - GetImageFormat, 209
 - GetImageHeight, 209
 - GetImageWidth, 209
 - GetInitialized, 210
 - GetLastRenderWallTime, 210
 - GetName, 210
 - GetNearClip, 210
 - GetOgreCamera, 210
 - GetPitchNode, 210
 - GetRenderRate, 211
 - GetRenderTexture, 211
 - GetRight, 211
 - GetScene, 211
 - GetSceneNode, 211
 - GetScopedName, 211
 - GetScreenshotPath, 212
 - GetTextureHeight, 212
 - GetTextureWidth, 212
 - GetTriangleCount, 212
 - GetUp, 212
 - GetVFOV, 212
 - GetViewport, 213
 - GetViewportHeight, 213
 - GetViewportWidth, 213
 - GetWindowId, 213

- GetWorldPointOnPlane, 213
- GetWorldPose, 214
- GetWorldPosition, 214
- GetWorldRotation, 214
- GetZValue, 214
- imageFormat, 222
- imageHeight, 222
- imageWidth, 222
- Init, 214
- initialized, 222
- IsAnimating, 214
- IsVisible, 215
- lastRenderWallTime, 223
- Load, 215
- MoveToPosition, 215
- MoveToPositions, 216
- name, 223
- newData, 223
- newImageFrame, 223
- onAnimationComplete, 223
- PostRender, 216
- prevAnimTime, 223
- ReadPixelBuffer, 216
- Render, 216
- RenderImpl, 217
- renderTarget, 223
- renderTexture, 223
- requests, 223
- RotatePitch, 217
- RotateYaw, 217
- saveCount, 223
- SaveFrame, 217
- saveFrameBuffer, 223
- scene, 224
- sceneNode, 224
- scopedName, 224
- scopedUniqueName, 224
- screenshotPath, 224
- sdf, 224
- SetAspectRatio, 218
- SetCaptureData, 218
- SetCaptureDataOnce, 218
- SetClipDist, 218
- SetHFOV, 218
- SetImageHeight, 218
- SetImageSize, 219
- SetImageWidth, 219
- SetName, 219
- SetRenderRate, 219
- SetRenderTarget, 219
- SetSaveFramePathname, 219
- SetScene, 220
- SetSceneNode, 220
- SetWindowId, 220
- SetWorldPose, 220
- SetWorldPosition, 220
- SetWorldRotation, 220
- ShowWireframe, 220
- textureHeight, 224
- textureWidth, 224
- ToggleShowWireframe, 221
- TrackVisual, 221
- TrackVisualImpl, 221
- Translate, 221
- Update, 222
- viewport, 224
- windowId, 224
- gazebo::rendering::CameraPrivate, 225
 - CameraCmdMsgs_L, 226
 - cameraCounter, 226
 - cmdSub, 226
 - commandMsgs, 226
 - dlGBufferInstance, 226
 - dlMergeInstance, 226
 - dsGBufferInstance, 226
 - dsMergeInstance, 226
 - moveToPositionQueue, 226
 - node, 226
 - receiveMutex, 227
 - renderPeriod, 227
 - ssaoInstance, 227
 - trackVisualPID, 227
 - trackVisualPitchPID, 227
 - trackVisualYawPID, 227
 - trackedVisual, 227
- gazebo::rendering::CameraVisual, 231
 - ~CameraVisual, 232
 - CameraVisual, 232
 - Load, 233
- gazebo::rendering::CameraVisualPrivate, 233
 - camera, 234
 - connections, 234
- gazebo::rendering::ContactVisual, 292
 - ~ContactVisual, 293
 - ContactVisual, 293
 - SetEnabled, 293
- gazebo::rendering::ContactVisualPrivate, 293
 - connections, 295
 - contactsMsg, 295
 - contactsSub, 295
 - enabled, 295
 - mutex, 295
 - node, 295
 - points, 295
 - receivedMsg, 295
 - topicName, 295
- gazebo::rendering::ContactVisualPrivate::ContactPoint, 286

- depth, 286
 - normal, 286
 - sceneNode, 286
- gazebo::rendering::Conversions, 296
 - Convert, 296, 297
- gazebo::rendering::DepthCamera, 383
 - ~DepthCamera, 384
 - ConnectNewDepthFrame, 385
 - ConnectNewRGBPointCloud, 385
 - CreateDepthTexture, 385
 - DepthCamera, 384
 - depthTarget, 387
 - depthTexture, 387
 - depthViewport, 387
 - DisconnectNewDepthFrame, 385
 - DisconnectNewRGBPointCloud, 385
 - Fini, 386
 - GetDepthData, 386
 - Init, 386
 - Load, 386
 - PostRender, 386
 - SetDepthTarget, 386
- gazebo::rendering::DummyPageProvider, 395
 - loadProceduralPage, 396
 - prepareProceduralPage, 396
 - unloadProceduralPage, 396
 - unprepareProceduralPage, 396
- gazebo::rendering::DynamicLines, 397
 - ~DynamicLines, 398
 - AddPoint, 398
 - Clear, 398
 - DynamicLines, 398
 - GetMovableType, 399
 - getMovableType, 399
 - GetPoint, 399
 - GetPointCount, 399
 - SetColor, 399
 - SetPoint, 399
 - Update, 400
- gazebo::rendering::DynamicRenderable, 400
 - ~DynamicRenderable, 401
 - CreateVertexDeclaration, 402
 - DynamicRenderable, 401
 - FillHardwareBuffers, 402
 - getBoundingRadius, 402
 - GetMovableType, 402
 - GetOperationType, 402
 - getSquaredViewDepth, 402
 - indexBufferCapacity, 404
 - Init, 403
 - PrepareHardwareBuffers, 403
 - SetOperationType, 403
 - vertexBufferCapacity, 404
- gazebo::rendering::Events, 432
 - ConnectCreateScene, 433
 - ConnectRemoveScene, 433
 - createScene, 434
 - DisconnectCreateScene, 434
 - DisconnectRemoveScene, 434
 - removeScene, 434
- gazebo::rendering::FPSViewController, 453
 - ~FPSViewController, 454
 - FPSViewController, 454
 - GetTypeString, 454
 - HandleKeyPressEvent, 454
 - HandleKeyReleaseEvent, 455
 - HandleMouseEvent, 455
 - Init, 455
 - Update, 455
- gazebo::rendering::GUIOverlay, 494
 - ~GUIOverlay, 495
 - AttachCameraToImage, 495
 - ButtonCallback, 495
 - callbacks, 498
 - CreateWindow, 496
 - GUIOverlay, 495
 - HandleKeyPressEvent, 496
 - HandleKeyReleaseEvent, 496
 - HandleMouseEvent, 496
 - Hide, 497
 - Init, 497
 - IsInitialized, 497
 - LoadLayout, 497
 - Resize, 497
 - Show, 497
 - Update, 497
- gazebo::rendering::GUIOverlayPrivate, 498
 - connections, 498
 - initialized, 498
 - layoutFilename, 498
 - rttImageSetCount, 498
- gazebo::rendering::GpuLaser, 467
 - ~GpuLaser, 470
 - cameraCount, 475
 - chfov, 475
 - ConnectNewLaserFrame, 470
 - CreateLaserTexture, 470
 - cvfov, 476
 - DisconnectNewLaserFrame, 470
 - far, 476
 - Fini, 470
 - GetCameraCount, 471
 - GetCosHorzFOV, 471
 - GetCosVertFOV, 471
 - GetFarClip, 471
 - GetHorzFOV, 471
 - GetHorzHalfAngle, 471
 - GetLaserData, 472

- GetNearClip, 472
- GetRayCountRatio, 472
- GetVertFOV, 472
- GetVertHalfAngle, 472
- GpuLaser, 470
- hfov, 476
- horzHalfAngle, 476
- Init, 472
- IsHorizontal, 473
- isHorizontal, 476
- Load, 473
- near, 476
- notifyRenderSingleObject, 473
- PostRender, 473
- rayCountRatio, 476
- SetCameraCount, 473
- SetCosHorzFOV, 473
- SetCosVertFOV, 474
- SetFarClip, 474
- SetHorzFOV, 474
- SetHorzHalfAngle, 474
- SetIsHorizontal, 474
- SetNearClip, 474
- SetRangeCount, 475
- SetRayCountRatio, 475
- SetVertFOV, 475
- SetVertHalfAngle, 475
- vertHalfAngle, 476
- vfov, 476
- gazebo::rendering::Grid, 488
 - ~Grid, 489
 - Enable, 490
 - GetCellCount, 490
 - GetCellLength, 490
 - GetColor, 490
 - GetHeight, 490
 - GetLineWidth, 490
 - GetSceneNode, 490
 - Grid, 489
 - Init, 491
 - SetCellCount, 491
 - SetCellLength, 491
 - SetColor, 491
 - SetHeight, 491
 - SetLineWidth, 491
 - SetUserData, 492
- gazebo::rendering::GzTerrainMatGen, 499
 - ~GzTerrainMatGen, 499
 - GzTerrainMatGen, 499
- gazebo::rendering::GzTerrainMatGen::SM2Profile, 1045
 - ~SM2Profile, 1046
 - addTechnique, 1046
 - generate, 1046
 - generateForCompositeMap, 1046
 - SM2Profile, 1046
 - UpdateParams, 1046
 - UpdateParamsForCompositeMap, 1046
- gazebo::rendering::GzTerrainMatGen::SM2Profile::~-
 - ShaderHelperCg, 928
 - defaultVpParams, 929
 - generateFragmentProgram, 929
 - generateVertexProgram, 929
 - generateVertexProgramSource, 929
 - generateVpDynamicShadows, 929
 - generateVpDynamicShadowsParams, 929
 - generateVpFooter, 929
 - generateVpHeader, 930
- gazebo::rendering::GzTerrainMatGen::SM2Profile::~-
 - ShaderHelperGLSL, 930
 - defaultVpParams, 931
 - generateFpDynamicShadows, 931
 - generateFpDynamicShadowsHelpers, 931
 - generateFpDynamicShadowsParams, 931
 - generateFpFooter, 931
 - generateFpHeader, 931
 - generateFpLayer, 931
 - generateFragmentProgram, 931
 - generateFragmentProgramSource, 931
 - generateVertexProgram, 932
 - generateVertexProgramSource, 932
 - generateVpDynamicShadows, 932
 - generateVpDynamicShadowsParams, 932
 - generateVpFooter, 932
 - generateVpHeader, 932
 - updateParams, 932
 - updateVpParams, 932
- gazebo::rendering::Heightmap, 499
 - ~Heightmap, 501
 - Flatten, 501
 - GetAvgHeight, 501
 - GetHeight, 501
 - GetImage, 502
 - GetMouseHit, 502
 - GetOgreTerrain, 502
 - GetTerrainSubdivisionCount, 502
 - Heightmap, 501
 - Load, 502
 - LoadFromMsg, 502
 - Lower, 503
 - Raise, 503
 - SetWireframe, 503
 - Smooth, 503
 - SplitHeights, 504
- gazebo::rendering::JointVisual, 579
 - ~JointVisual, 579
 - JointVisual, 579
 - Load, 580
- gazebo::rendering::JointVisualPrivate, 580

- axisVisual, 581
- gazebo::rendering::LaserVisual, 586
 - ~LaserVisual, 587
 - LaserVisual, 586
 - SetEmissive, 587
- gazebo::rendering::LaserVisualPrivate, 587
 - connection, 588
 - laserMsg, 588
 - laserScanSub, 588
 - mutex, 588
 - node, 588
 - rayFans, 588
 - receivedMsg, 588
- gazebo::rendering::Light, 589
 - ~Light, 591
 - FillMsg, 591
 - GetDiffuseColor, 591
 - GetDirection, 591
 - GetName, 591
 - GetPosition, 591
 - GetSpecularColor, 591
 - GetType, 592
 - GetVisible, 592
 - Light, 590
 - Load, 592
 - LoadFromMsg, 592
 - OnPoseChange, 593
 - SetAttenuation, 593
 - SetCastShadows, 593
 - SetDiffuseColor, 593
 - SetDirection, 593
 - SetLightType, 593
 - SetName, 594
 - SetPosition, 594
 - SetRange, 594
 - SetSelected, 594
 - SetSpecularColor, 594
 - SetSpotFalloff, 594
 - SetSpotInnerAngle, 595
 - SetSpotOuterAngle, 595
 - ShowVisual, 595
 - ToggleShowVisual, 595
 - UpdateFromMsg, 595
- gazebo::rendering::MovableText, 709
 - ~MovableText, 711
 - _setupGeometry, 711
 - _updateColors, 711
 - GetAABB, 711
 - GetBaseline, 711
 - getBoundingRadius, 712
 - GetCharHeight, 712
 - GetColor, 712
 - GetFont, 712
 - getLights, 712
 - getMaterial, 712
 - getRenderOperation, 712
 - GetShowOnTop, 712
 - GetSpaceWidth, 712
 - getSquaredViewDepth, 712
 - GetText, 713
 - getWorldTransforms, 713
 - H_CENTER, 711
 - H_LEFT, 711
 - HorizAlign, 711
 - Load, 713
 - MovableText, 711
 - SetBaseline, 713
 - SetCharHeight, 713
 - SetColor, 713
 - SetFontName, 713
 - SetShowOnTop, 714
 - SetSpaceWidth, 714
 - SetText, 714
 - SetTextAlignment, 714
 - Update, 714
 - V_ABOVE, 711
 - V_BELOW, 711
 - VertAlign, 711
 - visitRenderables, 714
- gazebo::rendering::OrbitViewController, 763
 - ~OrbitViewController, 764
 - GetFocalPoint, 764
 - GetTypeString, 764
 - HandleKeyPressEvent, 765
 - HandleKeyReleaseEvent, 765
 - HandleMouseEvent, 765
 - Init, 765
 - OrbitViewController, 764
 - SetDistance, 766
 - SetFocalPoint, 766
 - Update, 766
- gazebo::rendering::Projector, 810
 - ~Projector, 811
 - GetParent, 811
 - Load, 811
 - Projector, 811
 - SetEnabled, 812
 - SetTexture, 812
 - Toggle, 812
- gazebo::rendering::RFIDTagVisual, 864
 - ~RFIDTagVisual, 865
 - RFIDTagVisual, 865
- gazebo::rendering::RFIDTagVisualPrivate, 865
 - node, 866
 - rfidSub, 866
- gazebo::rendering::RFIDVisual, 867
 - ~RFIDVisual, 867
 - RFIDVisual, 867

- gazebo::rendering::RFIDVisualPrivate, 868
 - node, 868
 - rfidSub, 869
- gazebo::rendering::RTShaderSystem, 875
 - AddScene, 877
 - ApplyShadows, 877
 - AttachEntity, 877
 - AttachViewport, 877
 - Clear, 878
 - DetachEntity, 878
 - DetachViewport, 878
 - Fini, 878
 - GenerateShaders, 878
 - GetPSSMShadowCameraSetup, 878
 - Init, 878
 - LightingModel, 877
 - RemoveScene, 878
 - RemoveShadows, 879
 - SSLM_NormalMapLightingObjectSpace, 877
 - SSLM_NormalMapLightingTangentSpace, 877
 - SSLM_PerPixelLighting, 877
 - SSLM_PerVertexLighting, 877
 - SetPerPixelLighting, 879
 - UpdateShaders, 879
- gazebo::rendering::RenderEngine, 855
 - AddResourcePath, 857
 - CreateScene, 857
 - DEFERRED, 856
 - dummyContext, 858
 - dummyDisplay, 858
 - dummyWindowId, 859
 - FORWARD, 856
 - Fini, 857
 - GetRenderPathType, 857
 - GetScene, 857
 - GetSceneCount, 858
 - GetWindowManager, 858
 - Init, 858
 - Load, 858
 - NONE, 856
 - RENDER_PATH_COUNT, 856
 - RemoveScene, 858
 - RenderPathType, 856
 - root, 859
 - VERTEX, 856
- gazebo::rendering::Road2d, 871
 - ~Road2d, 871
 - Load, 871
 - Road2d, 871
- gazebo::rendering::Scene, 879
 - ~Scene, 883
 - AddVisual, 883
 - Clear, 883
 - CloneVisual, 883
 - CreateCamera, 884
 - CreateDepthCamera, 884
 - CreateGpuLaser, 884
 - CreateGrid, 885
 - CreateUserCamera, 885
 - DrawLine, 885
 - GZ_SKYX_ALL, 883
 - GZ_SKYX_CLOUDS, 883
 - GZ_SKYX_MOON, 883
 - GZ_SKYX_NONE, 883
 - GetAmbientColor, 885
 - GetBackgroundColor, 885
 - GetCamera, 886
 - GetCameraCount, 886
 - GetFirstContact, 886
 - GetGrid, 887
 - GetGridCount, 887
 - GetHeightBelowPoint, 887
 - GetHeightmap, 887
 - GetId, 887
 - GetIdString, 887
 - GetInitialized, 888
 - GetLight, 888
 - GetLightCount, 888
 - GetManager, 888
 - GetModelVisualAt, 889
 - GetName, 889
 - GetSelectedVisual, 889
 - GetShadowsEnabled, 889
 - GetShowClouds, 889
 - GetSimTime, 889
 - GetUserCamera, 890
 - GetUserCameraCount, 890
 - GetVisual, 890
 - GetVisualAt, 891
 - GetVisualBelow, 891
 - GetVisualCount, 891
 - GetVisualsBelowPoint, 892
 - GetWorldVisual, 892
 - Init, 892
 - Load, 892
 - PreRender, 892
 - PrintSceneGraph, 892
 - RemoveCamera, 893
 - RemoveProjectors, 893
 - RemoveVisual, 893
 - Scene, 883
 - SelectVisual, 893
 - SetAmbientColor, 893
 - SetBackgroundColor, 893
 - SetFog, 893
 - SetGrid, 894
 - SetShadowsEnabled, 894
 - SetSkyXMode, 894

- SetTransparent, 894
- SetVisible, 894
- SetWireframe, 895
- ShowCOMs, 895
- ShowClouds, 895
- ShowCollisions, 895
- ShowContacts, 895
- ShowJoints, 895
- SkyXMode, 883
- skyx, 896
- SnapVisualToNearestBelow, 896
- StripSceneName, 896
- gazebo::rendering::SelectionObj, 899
 - ~SelectionObj, 901
 - Attach, 902
 - Detach, 902
 - GetMode, 902
 - GetState, 902
 - Load, 902
 - ROT, 901
 - ROT_X, 901
 - ROT_Y, 901
 - ROT_Z, 901
 - SCALE, 901
 - SCALE_X, 901
 - SCALE_Y, 901
 - SCALE_Z, 901
 - SELECTION_NONE, 901
 - SelectionMode, 901
 - SelectionObj, 901
 - SetGlobal, 902
 - SetMode, 902
 - SetState, 903
 - TRANS, 901
 - TRANS_X, 901
 - TRANS_Y, 901
 - TRANS_Z, 901
 - UpdateSize, 903
- gazebo::rendering::SelectionObjPrivate, 903
 - maxScale, 905
 - minScale, 905
 - mode, 905
 - rotVisual, 905
 - rotXVisual, 905
 - rotYVisual, 905
 - rotZVisual, 906
 - scaleVisual, 906
 - scaleXVisual, 906
 - scaleYVisual, 906
 - scaleZVisual, 906
 - selectedVis, 906
 - state, 906
 - transVisual, 906
 - transXVisual, 906
 - transYVisual, 906
 - transZVisual, 906
 - xAxisMat, 907
 - xAxisMatOverlay, 907
 - yAxisMat, 907
 - yAxisMatOverlay, 907
 - zAxisMat, 907
 - zAxisMatOverlay, 907
- gazebo::rendering::SonarVisual, 1051
 - ~SonarVisual, 1052
 - Load, 1052
 - SonarVisual, 1052
- gazebo::rendering::SonarVisualPrivate, 1052
 - coneNode, 1054
 - connections, 1054
 - mutex, 1054
 - node, 1054
 - receivedMsg, 1054
 - sonarMsg, 1054
 - sonarRay, 1054
 - sonarSub, 1054
- gazebo::rendering::TransmitterVisual, 1131
 - ~TransmitterVisual, 1132
 - Load, 1132
 - TransmitterVisual, 1132
 - Update, 1132
- gazebo::rendering::TransmitterVisualPrivate, 1132
 - connections, 1134
 - gridMsg, 1134
 - isFirst, 1134
 - mutex, 1134
 - node, 1134
 - points, 1134
 - receivedMsg, 1134
 - signalPropagationSub, 1134
 - vectorLink, 1134
- gazebo::rendering::UserCamera, 1137
 - ~UserCamera, 1140
 - AnimationComplete, 1140
 - AttachToVisualImpl, 1140
 - EnableViewController, 1140
 - Fini, 1141
 - GetAvgFPS, 1141
 - GetGUIOverlay, 1141
 - GetImageHeight, 1141
 - GetImageWidth, 1141
 - GetTriangleCount, 1141
 - GetViewControllerTypeString, 1142
 - GetVisual, 1142
 - HandleKeyPressEvent, 1142
 - HandleKeyReleaseEvent, 1142
 - HandleMouseEvent, 1143
 - Init, 1143
 - IsCameraSetInWorldFile, 1143

- Load, 1143
- MoveToPosition, 1143
- MoveToVisual, 1144
- PostRender, 1144
- Resize, 1144
- SetFocalPoint, 1144
- SetRenderTarget, 1144
- SetUseSDFPose, 1145
- SetViewController, 1145
- SetViewportDimensions, 1145
- SetWorldPose, 1145
- TrackVisualImpl, 1146
- Update, 1146
- UserCamera, 1140
- gazebo::rendering::UserCameraPrivate, 1146
 - fpsViewController, 1147
 - gui, 1147
 - isCameraSetInWorldFile, 1147
 - orbitViewController, 1147
 - selectionBuffer, 1147
 - viewController, 1147
- gazebo::rendering::VideoVisual, 1189
 - ~VideoVisual, 1190
 - VideoVisual, 1190
- gazebo::rendering::VideoVisualPrivate, 1191
 - connections, 1191
 - height, 1192
 - imageBuffer, 1192
 - texture, 1192
 - video, 1192
 - width, 1192
- gazebo::rendering::ViewController, 1192
 - ~ViewController, 1194
 - camera, 1195
 - enabled, 1196
 - GetTypeString, 1194
 - HandleKeyPressEvent, 1194
 - HandleKeyReleaseEvent, 1194
 - HandleMouseEvent, 1195
 - Init, 1195
 - SetEnabled, 1195
 - typeString, 1196
 - Update, 1195
 - ViewController, 1194
- gazebo::rendering::Visual, 1196
 - ~Visual, 1202
 - AttachAxes, 1202
 - AttachLineVertex, 1203
 - AttachMesh, 1203
 - AttachObject, 1203
 - AttachVisual, 1203
 - ClearParent, 1203
 - Clone, 1203
 - CreateDynamicLine, 1204
 - dataPtr, 1217
 - DeleteDynamicLine, 1204
 - DetachObjects, 1204
 - DetachVisual, 1204
 - DisableTrackVisual, 1204
 - EnableTrackVisual, 1205
 - Fin, 1205
 - GetAttachedObjectCount, 1205
 - GetBoundingBox, 1205
 - GetChild, 1205
 - GetChildCount, 1205
 - GetHighlighted, 1206
 - GetId, 1206
 - GetMaterialName, 1206
 - GetMeshName, 1206
 - GetName, 1206
 - GetNormalMap, 1206
 - GetParent, 1206
 - GetPose, 1207
 - GetPosition, 1207
 - GetRootVisual, 1207
 - GetRotation, 1207
 - GetScale, 1207
 - GetScene, 1207
 - GetSceneNode, 1208
 - GetShaderType, 1208
 - GetSubMeshName, 1208
 - GetTransparency, 1208
 - GetVisibilityFlags, 1208
 - GetVisible, 1208
 - GetWorldPose, 1209
 - HasAttachedObject, 1209
 - Init, 1209
 - InsertMesh, 1209
 - IsPlane, 1210
 - IsStatic, 1210
 - Load, 1210
 - LoadFromMsg, 1210
 - LoadPlugin, 1210
 - MakeStatic, 1211
 - MoveToPosition, 1211
 - MoveToPositions, 1211
 - RemovePlugin, 1211
 - SetAmbient, 1211
 - SetCastShadows, 1211
 - SetDiffuse, 1212
 - SetEmissive, 1212
 - SetHighlighted, 1212
 - SetId, 1212
 - SetLighting, 1212
 - SetMaterial, 1212
 - SetName, 1213
 - SetNormalMap, 1213
 - SetPose, 1213

- SetPosition, 1213
- SetRibbonTrail, 1213
- SetRotation, 1214
- SetScale, 1214
- SetScene, 1214
- SetShaderType, 1214
- SetSkeletonPose, 1214
- SetSpecular, 1214
- SetTransparency, 1215
- SetVisibilityFlags, 1215
- SetVisible, 1215
- SetWireframe, 1215
- SetWorldPose, 1215
- SetWorldPosition, 1216
- SetWorldRotation, 1216
- ShowBoundingBox, 1216
- ShowCOM, 1216
- ShowCollision, 1216
- ShowJoints, 1216
- ShowSkeleton, 1217
- ToggleVisible, 1217
- Update, 1217
- UpdateFromMsg, 1217
- Visual, 1202
- gazebo::rendering::VisualPrivate, 1219
 - animState, 1222
 - boundingBox, 1222
 - children, 1222
 - id, 1222
 - initialized, 1222
 - isStatic, 1222
 - lighting, 1222
 - lineVertices, 1223
 - lines, 1222
 - myMaterialName, 1223
 - name, 1223
 - onAnimationComplete, 1223
 - origMaterialName, 1223
 - parent, 1223
 - plugins, 1223
 - preRenderConnection, 1223
 - prevAnimTime, 1223
 - ribbonTrail, 1223
 - scale, 1223
 - scene, 1224
 - sceneNode, 1224
 - sdf, 1224
 - skeleton, 1224
 - staticGeom, 1224
 - transparency, 1224
 - useRTShader, 1224
 - visible, 1224
 - visualIdCount, 1224
- gazebo::rendering::WindowManager, 1225
 - ~WindowManager, 1225
 - CreateWindow, 1226
 - Fini, 1226
 - GetAvgFPS, 1226
 - GetTriangleCount, 1226
 - GetWindow, 1226
 - Moved, 1227
 - Resize, 1227
 - SetCamera, 1227
 - WindowManager, 1225
- gazebo::rendering::WireBox, 1227
 - ~WireBox, 1228
 - GetBox, 1228
 - GetVisible, 1228
 - Init, 1228
 - SetVisible, 1229
 - WireBox, 1228
- gazebo::rendering::WireBoxPrivate, 1229
 - lines, 1229
 - parent, 1229
- gazebo::rendering::WrenchVisual, 1259
 - ~WrenchVisual, 1260
 - Load, 1260
 - SetEnabled, 1260
 - WrenchVisual, 1260
- gazebo::rendering::WrenchVisualPrivate, 1261
 - coneXNode, 1262
 - coneYNode, 1262
 - coneZNode, 1262
 - connections, 1262
 - enabled, 1262
 - forceLine, 1262
 - forceNode, 1262
 - mutex, 1262
 - node, 1263
 - receivedMsg, 1263
 - wrenchMsg, 1263
 - wrenchSub, 1263
- gazebo::sensors, 129
 - CATEGORY_COUNT, 133
 - CameraSensor_V, 132
 - CameraSensorPtr, 132
 - ContactSensor_V, 132
 - ContactSensorPtr, 132
 - DepthCameraSensor_V, 132
 - DepthCameraSensorPtr, 132
 - ForceTorqueSensorPtr, 132
 - GaussianNoiseModelPtr, 132
 - GpsSensorPtr, 132
 - GpuRaySensor_V, 132
 - GpuRaySensorPtr, 132
 - IMAGE, 133
 - ImageGaussianNoiseModelPtr, 132
 - ImuSensor_V, 132

- ImuSensorPtr, 132
- MultiCameraSensor_V, 132
- MultiCameraSensorPtr, 132
- NoisePtr, 132
- OTHER, 133
- RAY, 133
- RFIDSensor_V, 133
- RFIDSensorPtr, 133
- RFIDTag_V, 133
- RFIDTagPtr, 133
- RaySensor_V, 132
- RaySensorPtr, 132
- Sensor_V, 133
- SensorCategory, 133
- SensorFactoryFn, 133
- SensorPtr, 133
- SonarSensorPtr, 133
- WirelessReceiver_V, 133
- WirelessReceiverPtr, 133
- WirelessTransceiver_V, 133
- WirelessTransceiverPtr, 133
- WirelessTransmitter_V, 133
- WirelessTransmitterPtr, 133
- gazebo::sensors::CameraSensor, 227
 - ~CameraSensor, 229
 - CameraSensor, 229
 - Fini, 229
 - GetCamera, 229
 - GetImageData, 229
 - GetImageHeight, 229
 - GetImageWidth, 229
 - GetTopic, 230
 - Init, 230
 - IsActive, 230
 - Load, 230
 - SaveFrame, 231
 - UpdateImpl, 231
- gazebo::sensors::ContactSensor, 287
 - ~ContactSensor, 289
 - ContactSensor, 289
 - Fini, 289
 - GetCollisionContactCount, 289
 - GetCollisionCount, 289
 - GetCollisionName, 289
 - GetContacts, 290
 - Init, 291
 - IsActive, 291
 - Load, 291
 - UpdateImpl, 291
- gazebo::sensors::DepthCameraSensor, 387
 - ~DepthCameraSensor, 388
 - DepthCameraSensor, 388
 - Fini, 388
 - GetDepthCamera, 388
- Init, 389
- Load, 389
- SaveFrame, 389
- SetActive, 389
- UpdateImpl, 390
- gazebo::sensors::ForceTorqueSensor, 449
 - ~ForceTorqueSensor, 450
 - ConnectUpdate, 451
 - DisconnectUpdate, 451
 - Fini, 451
 - ForceTorqueSensor, 450
 - GetForce, 451
 - GetJoint, 451
 - GetTopic, 451
 - GetTorque, 452
 - Init, 452
 - IsActive, 452
 - Load, 452
 - update, 453
 - UpdateImpl, 452
- gazebo::sensors::GaussianNoiseModel, 457
 - ~GaussianNoiseModel, 459
 - ApplyImpl, 459
 - bias, 460
 - Fini, 459
 - GaussianNoiseModel, 459
 - GetBias, 459
 - GetMean, 460
 - GetStdDev, 460
 - Load, 460
 - mean, 460
 - precision, 460
 - quantized, 461
 - stdDev, 461
- gazebo::sensors::GpsSensor, 464
 - ~GpsSensor, 465
 - Fini, 465
 - GetAltitude, 465
 - GetLatitude, 465
 - GetLongitude, 466
 - GpsSensor, 465
 - Init, 466
 - Load, 466
 - UpdateImpl, 466
- gazebo::sensors::GpuRaySensor, 477
 - ~GpuRaySensor, 479
 - cameraElem, 487
 - ConnectNewLaserFrame, 480
 - DisconnectNewLaserFrame, 480
 - Fini, 480
 - GetAngleMax, 480
 - GetAngleMin, 480
 - GetAngleResolution, 480
 - GetCameraCount, 480

- GetCosHorzFOV, 481
- GetCosVertFOV, 481
- GetFiducial, 481
- GetHorzFOV, 481
- GetHorzHalfAngle, 481
- GetLaserCamera, 482
- GetRange, 482
- GetRangeCount, 482
- GetRangeCountRatio, 482
- GetRangeMax, 482
- GetRangeMin, 483
- GetRangeResolution, 483
- GetRanges, 483
- GetRayCount, 483
- GetRayCountRatio, 483
- GetRetro, 483
- GetTopic, 484
- GetVertFOV, 484
- GetVertHalfAngle, 484
- GetVerticalAngleMax, 484
- GetVerticalAngleMin, 484
- GetVerticalAngleResolution, 485
- GetVerticalRangeCount, 485
- GetVerticalRayCount, 485
- GpuRaySensor, 479
- horzElem, 487
- horzRangeCount, 487
- horzRayCount, 487
- Init, 485
- IsActive, 485
- IsHorizontal, 485
- Load, 486
- rangeCountRatio, 487
- rangeElem, 488
- scanElem, 488
- SetAngleMax, 486
- SetAngleMin, 486
- SetVerticalAngleMax, 486
- SetVerticalAngleMin, 487
- UpdateImpl, 487
- vertElem, 488
- vertRangeCount, 488
- vertRayCount, 488
- gazebo::sensors::ImageGaussianNoiseModel, 520
 - ~ImageGaussianNoiseModel, 522
 - Fini, 522
 - gaussianNoiseCompositorListener, 522
 - gaussianNoiseInstance, 522
 - ImageGaussianNoiseModel, 521
 - Load, 522
 - SetCamera, 522
- gazebo::sensors::ImuSensor, 525
 - ~ImuSensor, 527
 - Fini, 527
 - GetAngularVelocity, 527
 - GetImuMessage, 527
 - GetLinearAcceleration, 527
 - GetOrientation, 527
 - ImuSensor, 527
 - Init, 528
 - IsActive, 528
 - Load, 528
 - SetReferencePose, 528
 - UpdateImpl, 528
- gazebo::sensors::MultiCameraSensor, 718
 - ~MultiCameraSensor, 720
 - Fini, 720
 - GetCamera, 720
 - GetCameraCount, 720
 - GetImageData, 721
 - GetImageHeight, 721
 - GetImageWidth, 721
 - GetTopic, 722
 - Init, 722
 - IsActive, 722
 - Load, 722
 - MultiCameraSensor, 720
 - SaveFrame, 722
 - UpdateImpl, 723
- gazebo::sensors::Noise, 748
 - ~Noise, 749
 - Apply, 749
 - ApplyImpl, 750
 - CUSTOM, 749
 - Fini, 750
 - GAUSSIAN, 749
 - GetNoiseType, 750
 - Load, 750
 - NONE, 749
 - Noise, 749
 - NoiseType, 749
 - SetCamera, 750
 - SetCustomNoiseCallback, 751
- gazebo::sensors::NoiseFactory, 751
 - NewNoiseModel, 751
- gazebo::sensors::RFIDSensor, 859
 - ~RFIDSensor, 860
 - AddTag, 860
 - Fini, 860
 - Init, 860
 - Load, 860, 861
 - RFIDSensor, 860
 - UpdateImpl, 861
- gazebo::sensors::RFIDTag, 861
 - ~RFIDTag, 863
 - Fini, 863
 - GetTagPose, 863
 - Init, 863

- Load, 863
- RFIDTag, 863
- UpdateImpl, 864
- gazebo::sensors::RaySensor, 842
 - ~RaySensor, 844
 - Fini, 844
 - GetAngleMax, 844
 - GetAngleMin, 844
 - GetAngleResolution, 844
 - GetFiducial, 845
 - GetLaserShape, 845
 - GetRange, 845
 - GetRangeCount, 846
 - GetRangeMax, 846
 - GetRangeMin, 846
 - GetRangeResolution, 846
 - GetRanges, 846
 - GetRayCount, 846
 - GetRetro, 847
 - GetTopic, 847
 - GetVerticalAngleMax, 847
 - GetVerticalAngleMin, 847
 - GetVerticalAngleResolution, 847
 - GetVerticalRangeCount, 848
 - GetVerticalRayCount, 848
 - Init, 848
 - IsActive, 848
 - Load, 848
 - RaySensor, 844
 - UpdateImpl, 848
- gazebo::sensors::Sensor, 907
 - ~Sensor, 911
 - active, 917
 - ConnectUpdated, 911
 - connections, 917
 - DisconnectUpdated, 911
 - FillMsg, 911
 - Fini, 912
 - GetCategory, 912
 - GetId, 912
 - GetLastMeasurementTime, 912
 - GetLastUpdateTime, 912
 - GetName, 912
 - GetNoise, 913
 - GetParentId, 913
 - GetParentName, 913
 - GetPose, 913
 - GetScopedName, 913
 - GetTopic, 914
 - GetType, 914
 - GetUpdateRate, 914
 - GetVisualize, 914
 - GetWorldName, 914
 - Init, 914
 - IsActive, 915
 - lastMeasurementTime, 917
 - lastUpdateTime, 917
 - Load, 915
 - NeedsUpdate, 916
 - node, 917
 - noises, 918
 - parentId, 918
 - parentName, 918
 - plugins, 918
 - pose, 918
 - poseSub, 918
 - ResetLastUpdateTime, 916
 - scene, 918
 - sdf, 918
 - Sensor, 911
 - SetActive, 916
 - SetParent, 916
 - SetUpdateRate, 916
 - Update, 916
 - UpdateImpl, 917
 - updatePeriod, 918
 - world, 918
- gazebo::sensors::SensorFactory, 919
 - GetSensorTypes, 919
 - NewSensor, 919
 - RegisterAll, 920
 - RegisterSensor, 920
- gazebo::sensors::SensorManager, 920
 - CreateSensor, 922
 - Fini, 922
 - GetSensor, 922
 - GetSensorTypes, 923
 - GetSensors, 923
 - Init, 923
 - RemoveSensor, 923
 - RemoveSensors, 923
 - ResetLastUpdateTimes, 923
 - RunThreads, 923
 - SensorsInitialized, 924
 - Stop, 924
 - Update, 924
- gazebo::sensors::SimTimeEvent, 1020
 - condition, 1021
 - time, 1021
- gazebo::sensors::SimTimeEventHandler, 1021
 - ~SimTimeEventHandler, 1021
 - AddRelativeEvent, 1022
 - SimTimeEventHandler, 1021
- gazebo::sensors::SonarSensor, 1047
 - ~SonarSensor, 1048
 - ConnectUpdate, 1048
 - DisconnectUpdate, 1048
 - Fini, 1049

- GetRadius, 1049
- GetRange, 1049
- GetRangeMax, 1049
- GetRangeMin, 1049
- GetTopic, 1049
- Init, 1050
- IsActive, 1050
- Load, 1050
- SonarSensor, 1048
- update, 1051
- UpdateImpl, 1050
- gazebo::sensors::WirelessReceiver, 1230
 - ~WirelessReceiver, 1231
 - Fini, 1231
 - GetMaxFreqFiltered, 1231
 - GetMinFreqFiltered, 1231
 - GetSensitivity, 1231
 - Init, 1231
 - Load, 1232
 - WirelessReceiver, 1231
- gazebo::sensors::WirelessTransceiver, 1232
 - ~WirelessTransceiver, 1233
 - Fini, 1234
 - gain, 1235
 - GetGain, 1234
 - GetPower, 1234
 - GetTopic, 1234
 - Init, 1234
 - Load, 1234
 - parentEntity, 1235
 - power, 1235
 - pub, 1235
 - referencePose, 1235
 - WirelessTransceiver, 1233
- gazebo::sensors::WirelessTransmitter, 1235
 - ~WirelessTransmitter, 1237
 - freq, 1238
 - GetESSID, 1237
 - GetFreq, 1237
 - GetSignalStrength, 1237
 - Init, 1238
 - Load, 1238
 - ModelStdDesv, 1238
 - NEmpty, 1238
 - NObstacle, 1239
 - UpdateImpl, 1238
 - WirelessTransmitter, 1237
- gazebo::transport, 133
 - ConnectionPtr, 136
 - MessagePtr, 136
 - NodePtr, 136
 - PublicationPtr, 136
 - PublicationTransportPtr, 136
 - PublisherPtr, 136
 - SubscriberPtr, 136
 - SubscriptionTransportPtr, 136
 - gazebo::transport::CallbackHelper, 192
 - ~CallbackHelper, 193
 - CallbackHelper, 193
 - GetId, 193
 - GetLatching, 193
 - GetMsgType, 193
 - HandleData, 193
 - HandleMessage, 194
 - IsLocal, 194
 - latching, 195
 - SetLatching, 194
 - gazebo::transport::CallbackHelperT
 - CallbackHelperT, 196
 - GetMsgType, 196
 - HandleData, 196
 - HandleMessage, 197
 - IsLocal, 197
 - gazebo::transport::CallbackHelperT< M >, 195
 - gazebo::transport::Connection, 264
 - ~Connection, 266
 - AcceptCallback, 266
 - AsyncRead, 266
 - Cancel, 266
 - Connect, 267
 - ConnectToShutdown, 267
 - Connection, 266
 - DisconnectShutdown, 267
 - EnqueueMsg, 267, 268
 - GetIPWhiteList, 268
 - GetId, 268
 - GetLocalAddress, 268
 - GetLocalHostname, 268
 - GetLocalPort, 268
 - GetLocalURI, 269
 - GetRemoteAddress, 269
 - GetRemoteHostname, 269
 - GetRemotePort, 269
 - GetRemoteURI, 269
 - IsOpen, 269
 - Listen, 270
 - ProcessWriteQueue, 270
 - Read, 270
 - ReadCallback, 266
 - Shutdown, 270
 - StartRead, 270
 - StopRead, 270
 - ValidateIP, 270
 - gazebo::transport::ConnectionManager, 271
 - Advertise, 272
 - ConnectToRemoteHost, 272
 - eventConnections, 275
 - Fini, 273

- GetAllPublishers, 273
- GetTopicNamespaces, 273
- Init, 273
- IsRunning, 273
- RegisterTopicNamespace, 274
- RemoveConnection, 274
- Run, 274
- Stop, 274
- Subscribe, 274
- TriggerUpdate, 274
- Unadvertise, 274
- Unsubscribe, 275
- gazebo::transport::ConnectionReadTask, 276
 - ConnectionReadTask, 277
 - execute, 277
- gazebo::transport::IOManager, 540
 - ~IOManager, 540
 - DecCount, 541
 - GetCount, 541
 - GetIO, 541
 - IOManager, 540
 - IncCount, 541
 - Stop, 541
- gazebo::transport::Node, 731
 - ~Node, 733
 - Advertise, 733
 - DecodeTopicName, 733
 - EncodeTopicName, 733
 - Fini, 734
 - GetId, 734
 - GetMsgType, 734
 - GetTopicNamespace, 734
 - HandleData, 734
 - HandleMessage, 734
 - HasLatchedSubscriber, 735
 - Init, 735
 - InsertLatchedMsg, 735
 - Node, 733
 - ProcessIncoming, 736
 - ProcessPublishers, 736
 - Publish, 736
 - RemoveCallback, 736
 - Subscribe, 736, 737
- gazebo::transport::Publication, 812
 - ~Publication, 814
 - AddPublisher, 814
 - AddSubscription, 814
 - AddTransport, 814
 - GetCallbackCount, 814
 - GetLocallyAdvertised, 815
 - GetMsgType, 815
 - GetNodeCount, 815
 - GetPrevMsg, 815
 - GetRemoteSubscriptionCount, 815
 - GetTransportCount, 816
 - HasTransport, 816
 - LocalPublish, 816
 - Publication, 814
 - Publish, 816
 - RemovePublisher, 816
 - RemoveSubscription, 817
 - RemoveTransport, 817
 - SetLocallyAdvertised, 817
 - SetPrevMsg, 817
- gazebo::transport::PublicationTransport, 818
 - ~PublicationTransport, 818
 - AddCallback, 819
 - Fini, 819
 - GetConnection, 819
 - GetMsgType, 819
 - GetTopic, 819
 - Init, 819
 - PublicationTransport, 818
- gazebo::transport::PublishTask, 823
 - execute, 824
 - PublishTask, 824
- gazebo::transport::Publisher, 820
 - ~Publisher, 821
 - Fini, 821
 - GetMsgType, 821
 - GetOutgoingCount, 821
 - GetPrevMsg, 821
 - GetPrevMsgPtr, 822
 - GetTopic, 822
 - HasConnections, 822
 - Publish, 822
 - Publisher, 821
 - SendMessage, 822
 - SetNode, 823
 - SetPublication, 823
 - WaitForConnection, 823
- gazebo::transport::RawCallbackHelper, 840
 - GetMsgType, 841
 - HandleData, 841
 - HandleMessage, 841
 - IsLocal, 842
 - RawCallbackHelper, 841
- gazebo::transport::SubscribeOptions, 1084
 - GetLatching, 1085
 - GetMsgType, 1085
 - GetNode, 1085
 - GetTopic, 1085
 - Init, 1086
 - SubscribeOptions, 1085
- gazebo::transport::Subscriber, 1086
 - ~Subscriber, 1087
 - GetCallbackId, 1087
 - GetTopic, 1087

- SetCallbackId, 1087
- Subscriber, 1087
- Unsubscribe, 1087
- gazebo::transport::SubscriptionTransport, 1088
 - ~SubscriptionTransport, 1089
 - GetConnection, 1089
 - HandleData, 1089
 - HandleMessage, 1089
 - Init, 1090
 - IsLocal, 1090
 - SubscriptionTransport, 1089
- gazebo::transport::TopicManager, 1122
 - AddNode, 1124
 - AddNodeToProcess, 1125
 - Advertise, 1125
 - ClearBuffers, 1125
 - ConnectPubToSub, 1125
 - ConnectSubToPub, 1126
 - ConnectSubscribers, 1125
 - DisconnectPubFromSub, 1126
 - DisconnectSubFromPub, 1126
 - FindPublication, 1126
 - Fini, 1126
 - GetTopicNamespaces, 1126
 - Init, 1127
 - IsAdvertised, 1127
 - PauseIncoming, 1127
 - ProcessNodes, 1127
 - Publish, 1127
 - RegisterTopicNamespace, 1128
 - RemoveNode, 1128
 - SubNodeMap, 1124
 - Subscribe, 1128
 - Unadvertise, 1128
 - Unsubscribe, 1129
 - UpdatePublications, 1129
- gazebo::util, 136
 - DiagnosticTimerPtr, 137
 - OpenALSinkPtr, 137
 - OpenALSourcePtr, 137
- gazebo::util::DiagnosticManager, 390
 - GetLabel, 392
 - GetLogPath, 392
 - GetTime, 392
 - GetTimerCount, 392
 - Init, 393
 - Lap, 393
 - StartTimer, 393
 - StopTimer, 393
- gazebo::util::DiagnosticTimer, 394
 - ~DiagnosticTimer, 395
 - DiagnosticTimer, 394
 - GetName, 395
 - Lap, 395
 - Start, 395
 - Stop, 395
- gazebo::util::LogPlay, 628
 - GetChunk, 629
 - GetChunkCount, 629
 - GetEncoding, 629
 - GetGazeboVersion, 630
 - GetHeader, 630
 - GetLogVersion, 630
 - GetRandSeed, 630
 - IsOpen, 630
 - Open, 630
 - Step, 631
- gazebo::util::LogRecord, 631
 - Add, 633
 - Fini, 634
 - GetBasePath, 634
 - GetBufferSize, 634
 - GetEncoding, 634
 - GetFileSize, 634
 - GetFilename, 634
 - GetFirstUpdate, 635
 - GetPaused, 635
 - GetRunTime, 635
 - GetRunning, 635
 - Init, 635
 - IsReadyToStart, 636
 - Notify, 636
 - Remove, 636
 - SetBasePath, 636
 - SetPaused, 636
 - Start, 637
 - Stop, 637
 - Write, 637
- gazebo::util::OpenAL, 755
 - CreateSink, 756
 - CreateSource, 756
 - Fini, 757
 - Load, 757
- gazebo::util::OpenALSink, 757
 - ~OpenALSink, 758
 - OpenALSink, 758
 - SetPose, 758
 - SetVelocity, 758
- gazebo::util::OpenALSource, 758
 - ~OpenALSource, 759
 - FillBufferFromFile, 760
 - FillBufferFromPCM, 760
 - GetCollisionNames, 760
 - GetOnContact, 760
 - HasCollisionName, 760
 - IsPlaying, 761
 - Load, 761
 - OpenALSource, 759

- Pause, 761
- Play, 761
- Rewind, 761
- SetGain, 761
- SetLoop, 762
- SetPitch, 762
- SetPose, 762
- SetVelocity, 762
- Stop, 763
- gazebo_core.hh, 1336
- GazeboGenerator
 - google::protobuf::compiler::cpp::GazeboGenerator, 461
- GazeboGenerator.hh, 1337
- gazeboPathsFromEnv
 - gazebo::common::SystemPaths, 1097
- gearRatio
 - gazebo::physics::GearboxJoint, 464
- GearboxJoint
 - gazebo::physics::GearboxJoint, 462
- GearboxJoint.hh, 1338
- GenSphericalTexCoord
 - gazebo::common::Mesh, 663
 - gazebo::common::MeshManager, 673
 - gazebo::common::SubMesh, 1079
- Generate
 - google::protobuf::compiler::cpp::GazeboGenerator, 461
- generate
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 1046
- generateForCompositeMap
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 1046
- generateFpDynamicShadows
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 931
- generateFpDynamicShadowsHelpers
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 931
- generateFpDynamicShadowsParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 931
- generateFpFooter
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 931
- generateFpHeader
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 931
- generateFpLayer
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 931
- generateFragmentProgram
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 929
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 931
- generateFragmentProgramSource
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 931
- GenerateShaders
 - gazebo::rendering::RTShaderSystem, 878
- generateVertexProgram
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 929
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 932
- generateVertexProgramSource
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 929
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 932
- generateVpDynamicShadows
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 929
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 932
- generateVpDynamicShadowsParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 929
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 932
- generateVpFooter
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 929
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 932
- generateVpHeader
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 930
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 932
- GeneratorType
 - gazebo::math, 113
- GeometryFromSDF
 - Messages, 63
- Get
 - Common, 42
 - gazebo::common::NodeTransform, 745
- get_master_uri
 - Transport, 94
- get_scene
 - Rendering, 85
- get_sensor
 - Sensors, 90
- get_sha1
 - Common, 42

- get_topic_namespaces
 - Transport, 95
- get_world
 - Classes for physics and dynamics, 72
- GetAABB
 - gazebo::common::Mesh, 663
 - gazebo::rendering::MovableText, 711
- GetAbs
 - gazebo::math::Vector3, 1169
- GetAcceleration
 - gazebo::physics::LinkState, 621
- getAdvertisedTopics
 - Transport, 95
- GetAllPublishers
 - gazebo::transport::ConnectionManager, 273
- GetAltitude
 - gazebo::sensors::GpsSensor, 465
- GetAmbient
 - gazebo::common::Material, 642
- GetAmbientColor
 - gazebo::rendering::Scene, 885
- GetAnchor
 - gazebo::physics::DARTBallJoint, 304
 - gazebo::physics::DARTHinge2Joint, 319
 - gazebo::physics::DARTHingeJoint, 324
 - gazebo::physics::DARTScrewJoint, 366
 - gazebo::physics::DARTSliderJoint, 373
 - gazebo::physics::DARTUniversalJoint, 380
 - gazebo::physics::Joint, 549
 - gazebo::physics::SimbodyBallJoint, 938
 - gazebo::physics::SimbodyHinge2Joint, 952
 - gazebo::physics::SimbodyJoint, 963
 - gazebo::physics::SimbodyUniversalJoint, 1017
- GetAnchorErrorPose
 - gazebo::physics::Joint, 550
- GetAngle
 - gazebo::physics::Joint, 550
 - gazebo::physics::JointState, 576
- GetAngleCount
 - gazebo::physics::BallJoint, 168
 - gazebo::physics::DARTJoint, 331
 - gazebo::physics::GearboxJoint, 463
 - gazebo::physics::Hinge2Joint, 513
 - gazebo::physics::HingeJoint, 514
 - gazebo::physics::Joint, 550
 - gazebo::physics::JointState, 576
 - gazebo::physics::ScrewJoint, 897
 - gazebo::physics::SliderJoint, 1045
 - gazebo::physics::UniversalJoint, 1136
- GetAngleImpl
 - gazebo::physics::DARTBallJoint, 304
 - gazebo::physics::DARTHinge2Joint, 320
 - gazebo::physics::DARTHingeJoint, 325
 - gazebo::physics::DARTScrewJoint, 366
 - gazebo::physics::DARTSliderJoint, 373
 - gazebo::physics::DARTUniversalJoint, 380
 - gazebo::physics::Joint, 550
 - gazebo::physics::SimbodyBallJoint, 938
 - gazebo::physics::SimbodyHinge2Joint, 952
 - gazebo::physics::SimbodyScrewJoint, 1004
 - gazebo::physics::SimbodySliderJoint, 1011
 - gazebo::physics::SimbodyUniversalJoint, 1018
- GetAngleMax
 - gazebo::sensors::GpuRaySensor, 480
 - gazebo::sensors::RaySensor, 844
- GetAngleMin
 - gazebo::sensors::GpuRaySensor, 480
 - gazebo::sensors::RaySensor, 844
- GetAngleResolution
 - gazebo::sensors::GpuRaySensor, 480
 - gazebo::sensors::RaySensor, 844
- GetAngles
 - gazebo::physics::JointState, 577
- GetAngularDamping
 - gazebo::physics::Link, 604
- GetAngularVelocity
 - gazebo::sensors::ImuSensor, 527
- GetAnimation
 - gazebo::common::Skeleton, 1027
- GetAsABGR
 - gazebo::common::Color, 253
- GetAsARGB
 - gazebo::common::Color, 253
- GetAsAxis
 - gazebo::math::Quaternion, 829
- GetAsBGRA
 - gazebo::common::Color, 253
- GetAsEuler
 - gazebo::math::Quaternion, 829
- GetAsHSV
 - gazebo::common::Color, 253
- GetAsMatrix3
 - gazebo::math::Quaternion, 829
- GetAsMatrix4
 - gazebo::math::Quaternion, 829
- GetAsPose
 - gazebo::math::Matrix4, 656
- GetAsRGBA
 - gazebo::common::Color, 253
- GetAsYUV
 - gazebo::common::Color, 253
- GetAspectRatio
 - gazebo::rendering::Camera, 207
- GetAttachedObjectCount
 - gazebo::rendering::Visual, 1205
- GetAttribute
 - gazebo::physics::DARTJoint, 331

- gazebo::physics::Joint, 551
- gazebo::physics::SimbodyJoint, 963
- gazebo::physics::SimbodyScrewJoint, 1004
- GetAutoDisable
 - gazebo::physics::Model, 683
- GetAutoDisableFlag
 - gazebo::physics::PhysicsEngine, 772
- GetAvgColor
 - gazebo::common::Image, 517
- GetAvgFPS
 - gazebo::rendering::Camera, 207
 - gazebo::rendering::UserCamera, 1141
 - gazebo::rendering::WindowManager, 1226
- GetAvgHeight
 - gazebo::rendering::Heightmap, 501
- GetAxis
 - gazebo::physics::SimbodyBallJoint, 938
 - gazebo::physics::SimbodyHinge2Joint, 953
 - gazebo::physics::SimbodyUniversalJoint, 1018
- GetAxisFrame
 - gazebo::physics::Joint, 551
- GetBPP
 - gazebo::common::Image, 517
- GetBackgroundColor
 - gazebo::rendering::Scene, 885
- GetBasePath
 - gazebo::util::LogRecord, 634
- GetBaseline
 - gazebo::rendering::MovableText, 711
- GetBias
 - gazebo::sensors::GaussianNoiseModel, 459
- GetBindShapeTransform
 - gazebo::common::Skeleton, 1027
- GetBlendFactors
 - gazebo::common::Material, 642
- GetBlendMode
 - gazebo::common::Material, 642
- GetBoundingBox
 - gazebo::physics::Collision, 239
 - gazebo::physics::DARTCollision, 311
 - gazebo::physics::Entity, 407
 - gazebo::physics::Link, 604
 - gazebo::physics::Model, 683
 - gazebo::physics::SimbodyCollision, 945
 - gazebo::rendering::Visual, 1205
- getBoundingRadius
 - gazebo::rendering::DynamicRenderable, 402
 - gazebo::rendering::MovableText, 712
- GetBox
 - gazebo::rendering::WireBox, 1228
- GetBufferSize
 - gazebo::util::LogRecord, 634
- GetById
 - gazebo::physics::Base, 174
- GetByName
 - gazebo::physics::Base, 174
 - gazebo::physics::World, 1243
- GetCallbackCount
 - gazebo::transport::Publication, 814
- GetCallbackId
 - gazebo::transport::Subscriber, 1087
- GetCamera
 - gazebo::rendering::Scene, 886
 - gazebo::sensors::CameraSensor, 229
 - gazebo::sensors::MultiCameraSensor, 720
- GetCameraCount
 - gazebo::rendering::GpuLaser, 471
 - gazebo::rendering::Scene, 886
 - gazebo::sensors::GpuRaySensor, 480
 - gazebo::sensors::MultiCameraSensor, 720
- GetCameraToViewportRay
 - gazebo::rendering::Camera, 207
- GetCaptureData
 - gazebo::rendering::Camera, 207
- GetCategory
 - gazebo::sensors::Sensor, 912
- GetCategoryBits
 - gazebo::physics::DARTCollision, 311
- GetCellCount
 - gazebo::rendering::Grid, 490
- GetCellLength
 - gazebo::rendering::Grid, 490
- GetCenter
 - gazebo::math::Box, 182
- GetCharHeight
 - gazebo::rendering::MovableText, 712
- GetChild
 - gazebo::common::SkeletonNode, 1038
 - gazebo::physics::Base, 174
 - gazebo::physics::Joint, 551
 - gazebo::rendering::Visual, 1205
- GetChildById
 - gazebo::common::SkeletonNode, 1038
- GetChildByName
 - gazebo::common::SkeletonNode, 1039
- GetChildCollision
 - gazebo::physics::Entity, 408
- GetChildCount
 - gazebo::common::SkeletonNode, 1039
 - gazebo::physics::Base, 175
 - gazebo::rendering::Visual, 1205
- GetChildJoints
 - gazebo::physics::Link, 604
- GetChildJointsLinks
 - gazebo::physics::Link, 604
- GetChildLink
 - gazebo::physics::Entity, 408
- GetChunk

- gazebo::util::LogPlay, 629
- GetChunkCount
 - gazebo::util::LogPlay, 629
- GetCmd
 - gazebo::common::PID, 784
- GetCmdMax
 - gazebo::common::PID, 784
- GetCmdMin
 - gazebo::common::PID, 784
- GetCoG
 - gazebo::physics::Inertial, 532
- GetCollideBits
 - gazebo::physics::DARTCollision, 312
- GetCollision
 - gazebo::physics::Link, 604, 605
- GetCollisionBoundingBox
 - gazebo::physics::Entity, 408
- GetCollisionById
 - gazebo::physics::Link, 605
- GetCollisionContactCount
 - gazebo::sensors::ContactSensor, 289
- GetCollisionCount
 - gazebo::sensors::ContactSensor, 289
- GetCollisionName
 - gazebo::sensors::ContactSensor, 289
- GetCollisionNames
 - gazebo::util::OpenALSource, 760
- GetCollisionShape
 - gazebo::physics::SimbodyCollision, 945
- GetCollisionState
 - gazebo::physics::LinkState, 621
- GetCollisionStateCount
 - gazebo::physics::LinkState, 622
- GetCollisionStates
 - gazebo::physics::LinkState, 622
- GetCollisions
 - gazebo::physics::Link, 605
- GetColor
 - gazebo::rendering::Grid, 490
 - gazebo::rendering::MovableText, 712
- GetConnection
 - gazebo::transport::PublicationTransport, 819
 - gazebo::transport::SubscriptionTransport, 1089
- GetContact
 - gazebo::physics::ContactManager, 284
- GetContactCount
 - gazebo::physics::ContactManager, 284
- GetContactManager
 - gazebo::physics::PhysicsEngine, 772
- GetContactMaxCorrectingVel
 - gazebo::physics::PhysicsEngine, 772
- GetContactSurfaceLayer
 - gazebo::physics::PhysicsEngine, 772
- GetContacts
 - gazebo::physics::ContactManager, 284
 - gazebo::sensors::ContactSensor, 290
- GetContactsEnabled
 - gazebo::physics::Collision, 239
- GetCosHorzFOV
 - gazebo::rendering::GpuLaser, 471
 - gazebo::sensors::GpuRaySensor, 481
- GetCosVertFOV
 - gazebo::rendering::GpuLaser, 471
 - gazebo::sensors::GpuRaySensor, 481
- GetCount
 - gazebo::transport::IOManager, 541
- GetCurrentDir
 - SystemPaths.hh, 1486
- GetDARTBodyNode
 - gazebo::physics::DARTCollision, 312
 - gazebo::physics::DARTLink, 342
- GetDARTCollisionShape
 - gazebo::physics::DARTCollision, 312
- GetDARTJoint
 - gazebo::physics::DARTJoint, 331
- GetDARTModel
 - gazebo::physics::DARTJoint, 331
 - gazebo::physics::DARTLink, 342
- GetDARTPhysics
 - gazebo::physics::DARTLink, 342
 - gazebo::physics::DARTModel, 351
- GetDARTSkeleton
 - gazebo::physics::DARTModel, 351
- GetDARTWorld
 - gazebo::physics::DARTLink, 342
 - gazebo::physics::DARTModel, 352
 - gazebo::physics::DARTPhysics, 357
- GetDBConfig
 - Common, 43
- GetDGain
 - gazebo::common::PID, 785
- GetDamping
 - gazebo::physics::Joint, 551
- GetDampingCoefficient
 - gazebo::physics::Joint, 552
- GetData
 - gazebo::common::Image, 517
- GetDbINormal
 - gazebo::math::Rand, 838
- GetDbIUniform
 - gazebo::math::Rand, 839
- GetDefaultTestPath
 - gazebo::common::SystemPaths, 1095
- GetDepthCamera
 - gazebo::sensors::DepthCameraSensor, 388
- GetDepthData
 - gazebo::rendering::DepthCamera, 386
- GetDepthWrite

- gazebo::common::Material, 643
- GetDiffuse
 - gazebo::common::Material, 643
- GetDiffuseColor
 - gazebo::rendering::Light, 591
- GetDirection
 - gazebo::rendering::Camera, 208
 - gazebo::rendering::Light, 591
- GetDirtyPose
 - gazebo::physics::Entity, 408
- GetDistToLine
 - gazebo::math::Vector3, 1169
- GetDynamicsWorld
 - gazebo::physics::SimbodyPhysics, 990
- GetESSID
 - gazebo::sensors::WirelessTransmitter, 1237
- GetEffectiveMassProps
 - gazebo::physics::SimbodyLink, 976
- GetEffortLimit
 - gazebo::physics::Joint, 552
- GetElapsed
 - gazebo::common::Timer, 1122
- GetElevationReference
 - gazebo::common::SphericalCoordinates, 1060
- GetEmissive
 - gazebo::common::Material, 643
- GetEnablePhysicsEngine
 - gazebo::physics::World, 1243
- GetEnabled
 - gazebo::physics::DARTLink, 343
 - gazebo::physics::Link, 605
 - gazebo::physics::SimbodyLink, 976
- GetEncoding
 - gazebo::util::LogPlay, 629
 - gazebo::util::LogRecord, 634
- GetEntity
 - gazebo::physics::World, 1243
- GetEntityBelowPoint
 - gazebo::physics::World, 1243
- GetErrorFile
 - gazebo::common::Exception, 446
- GetErrorStr
 - gazebo::common::Exception, 446
- GetErrors
 - gazebo::common::PID, 785
- GetEulerRotation
 - gazebo::math::Matrix4, 656
- GetExp
 - gazebo::math::Quaternion, 830
- GetFarClip
 - gazebo::rendering::Camera, 208
 - gazebo::rendering::GpuLaser, 471
- GetFiducial
 - gazebo::physics::MultiRayShape, 727
 - gazebo::physics::RayShape, 851
- gazebo::sensors::GpuRaySensor, 481
- gazebo::sensors::RaySensor, 845
- GetFile
 - gazebo::common::AudioDecoder, 162
- GetFileSize
 - gazebo::util::LogRecord, 634
- GetFilename
 - gazebo::common::Image, 517
 - gazebo::common::ImageHeightmap, 524
 - gazebo::PluginT, 796
 - gazebo::util::LogRecord, 634
- GetFilterCount
 - gazebo::physics::ContactManager, 285
- GetFirstContact
 - gazebo::rendering::Scene, 886
- GetFirstUpdate
 - gazebo::util::LogRecord, 635
- GetFocalPoint
 - gazebo::rendering::OrbitViewController, 764
- GetFont
 - gazebo::rendering::MovableText, 712
- GetForce
 - gazebo::physics::DARTJoint, 332
 - gazebo::physics::Joint, 552
 - gazebo::physics::SimbodyJoint, 964
 - gazebo::sensors::ForceTorqueSensor, 451
- GetForceTorque
 - gazebo::physics::DARTJoint, 332
 - gazebo::physics::Joint, 552
 - gazebo::physics::SimbodyJoint, 964
- GetForces
 - gazebo::physics::JointController, 569
- GetFrameAt
 - gazebo::common::NodeAnimation, 739
- GetFrameCount
 - gazebo::common::NodeAnimation, 739
- GetFrameFilename
 - gazebo::rendering::Camera, 208
- GetFreq
 - gazebo::sensors::WirelessTransmitter, 1237
- GetGUIOverlay
 - gazebo::rendering::UserCamera, 1141
- GetGain
 - gazebo::sensors::WirelessTransceiver, 1234
- GetGazeboPaths
 - gazebo::common::SystemPaths, 1096
- GetGazeboVersion
 - gazebo::util::LogPlay, 630
- GetGearboxRatio
 - gazebo::physics::GearboxJoint, 463
- GetGlobalAxis
 - gazebo::physics::DARTBallJoint, 304
 - gazebo::physics::DARTHinge2Joint, 320

- gazebo::physics::DARTHingeJoint, 325
- gazebo::physics::DARTScrewJoint, 366
- gazebo::physics::DARTSliderJoint, 373
- gazebo::physics::DARTUniversalJoint, 380
- gazebo::physics::Joint, 553
- gazebo::physics::SimbodyBallJoint, 938
- gazebo::physics::SimbodyHinge2Joint, 953
- gazebo::physics::SimbodyHingeJoint, 957
- gazebo::physics::SimbodyScrewJoint, 1004
- gazebo::physics::SimbodySliderJoint, 1012
- gazebo::physics::SimbodyUniversalJoint, 1018
- GetGlobalPoints
 - gazebo::physics::RayShape, 852
- GetGravity
 - gazebo::physics::PhysicsEngine, 772
- GetGravityMode
 - gazebo::physics::DARTLink, 343
 - gazebo::physics::Link, 605
 - gazebo::physics::SimbodyLink, 976
- GetGrid
 - gazebo::rendering::Scene, 887
- GetGridCount
 - gazebo::rendering::Scene, 887
- GetGripper
 - gazebo::physics::Model, 683
- GetGripperCount
 - gazebo::physics::Model, 684
- GetHFOV
 - gazebo::rendering::Camera, 208
- GetHandle
 - gazebo::common::SkeletonNode, 1039
 - gazebo::PluginT, 796
- GetHeader
 - gazebo::util::LogPlay, 630
 - Messages, 63
- GetHeadingOffset
 - gazebo::common::SphericalCoordinates, 1060
- GetHeight
 - gazebo::common::HeightmapData, 506
 - gazebo::common::Image, 518
 - gazebo::common::ImageHeightmap, 524
 - gazebo::common::Video, 1188
 - gazebo::physics::HeightmapShape, 509
 - gazebo::rendering::Grid, 490
 - gazebo::rendering::Heightmap, 501
- GetHeightBelowPoint
 - gazebo::rendering::Scene, 887
- GetHeightmap
 - gazebo::rendering::Scene, 887
- GetHighStop
 - gazebo::physics::DARTBallJoint, 304
 - gazebo::physics::DARTJoint, 332
 - gazebo::physics::DARTScrewJoint, 367
 - gazebo::physics::Joint, 553
 - gazebo::physics::SimbodyBallJoint, 938
 - gazebo::physics::SimbodyJoint, 964
 - gazebo::physics::SimbodyScrewJoint, 1005
- GetHighlighted
 - gazebo::rendering::Visual, 1206
- GetHorzFOV
 - gazebo::rendering::GpuLaser, 471
 - gazebo::sensors::GpuRaySensor, 481
- GetHorzHalfAngle
 - gazebo::rendering::GpuLaser, 471
 - gazebo::sensors::GpuRaySensor, 481
- GetIGain
 - gazebo::common::PID, 785
- GetIMax
 - gazebo::common::PID, 785
- GetIMin
 - gazebo::common::PID, 785
- GetIO
 - gazebo::transport::IOManager, 541
- GetIPWhiteList
 - gazebo::transport::Connection, 268
- GetIXX
 - gazebo::physics::Inertial, 532
- GetIXY
 - gazebo::physics::Inertial, 532
- GetIXZ
 - gazebo::physics::Inertial, 532
- GetIYY
 - gazebo::physics::Inertial, 532
- GetIYZ
 - gazebo::physics::Inertial, 533
- GetIZZ
 - gazebo::physics::Inertial, 533
- GetId
 - gazebo::common::SkeletonNode, 1039
 - gazebo::event::Connection, 264
 - gazebo::physics::Base, 175
 - gazebo::rendering::Scene, 887
 - gazebo::rendering::Visual, 1206
 - gazebo::sensors::Sensor, 912
 - gazebo::transport::CallbackHelper, 193
 - gazebo::transport::Connection, 268
 - gazebo::transport::Node, 734
- GetIdString
 - gazebo::rendering::Scene, 887
- GetImage
 - gazebo::physics::HeightmapShape, 509
 - gazebo::rendering::Heightmap, 502
- GetImageByteSize
 - gazebo::rendering::Camera, 208
- GetImageData
 - gazebo::rendering::Camera, 209
 - gazebo::sensors::CameraSensor, 229
 - gazebo::sensors::MultiCameraSensor, 721

- GetImageDepth
 - gazebo::rendering::Camera, 209
- GetImageFormat
 - gazebo::rendering::Camera, 209
- GetImageHeight
 - gazebo::rendering::Camera, 209
 - gazebo::rendering::UserCamera, 1141
 - gazebo::sensors::CameraSensor, 229
 - gazebo::sensors::MultiCameraSensor, 721
- GetImageWidth
 - gazebo::rendering::Camera, 209
 - gazebo::rendering::UserCamera, 1141
 - gazebo::sensors::CameraSensor, 229
 - gazebo::sensors::MultiCameraSensor, 721
- GetImuMessage
 - gazebo::sensors::ImuSensor, 527
- GetIndex
 - gazebo::common::SubMesh, 1079
- GetIndexCount
 - gazebo::common::Mesh, 663
 - gazebo::common::SubMesh, 1079
- GetInertiaRatio
 - gazebo::physics::Joint, 554
- GetInertial
 - gazebo::physics::Inertial, 532
 - gazebo::physics::Link, 606
- GetInitialAnchorPose
 - gazebo::physics::Joint, 554
- GetInitialRelativePose
 - gazebo::physics::Entity, 408
- GetInitialized
 - gazebo::rendering::Camera, 210
 - gazebo::rendering::Scene, 888
 - gazebo::Server, 927
- GetIntNormal
 - gazebo::math::Rand, 839
- GetIntUniform
 - gazebo::math::Rand, 839
- GetInterpolatedKeyFrame
 - gazebo::common::NumericAnimation, 753
 - gazebo::common::PoseAnimation, 807
- GetIntersection
 - DART Physics, 78
 - gazebo::physics::RayShape, 852
 - gazebo::physics::SimbodyRayShape, 1001
- GetInverse
 - gazebo::math::Pose, 801
 - gazebo::math::Quaternion, 830
- GetInverseBindTransform
 - gazebo::common::SkeletonNode, 1039
- GetIterations
 - gazebo::physics::World, 1244
- GetJoint
 - gazebo::physics::Model, 684
 - gazebo::sensors::ForceTorqueSensor, 451
- GetJointController
 - gazebo::physics::Model, 684
- GetJointCount
 - gazebo::physics::Model, 684
- GetJointLink
 - gazebo::physics::DARTJoint, 333
 - gazebo::physics::Joint, 554
 - gazebo::physics::SimbodyJoint, 965
- GetJointState
 - gazebo::physics::ModelState, 701
- GetJointStateCount
 - gazebo::physics::ModelState, 701
- GetJointStates
 - gazebo::physics::ModelState, 702
- GetJoints
 - gazebo::physics::JointController, 570
 - gazebo::physics::Model, 684
- GetKeyFrame
 - gazebo::common::Animation, 155
 - gazebo::common::NodeAnimation, 740
- GetKeyFrameCount
 - gazebo::common::Animation, 155
- GetKeyFramesAtTime
 - gazebo::common::Animation, 156
- GetKinematic
 - gazebo::physics::DARTLink, 343
 - gazebo::physics::Link, 606
- GetLabel
 - gazebo::util::DiagnosticManager, 392
- GetLaserCamera
 - gazebo::sensors::GpuRaySensor, 482
- GetLaserData
 - gazebo::rendering::GpuLaser, 472
- GetLaserRetro
 - gazebo::physics::Collision, 239
- GetLaserShape
 - gazebo::sensors::RaySensor, 845
- GetLastMeasurementTime
 - gazebo::sensors::Sensor, 912
- GetLastRenderWallTime
 - gazebo::rendering::Camera, 210
- GetLastUpdateTime
 - gazebo::physics::JointController, 570
 - gazebo::sensors::Sensor, 912
- GetLatching
 - gazebo::transport::CallbackHelper, 193
 - gazebo::transport::SubscribeOptions, 1085
- GetLatitude
 - gazebo::sensors::GpsSensor, 465
- GetLatitudeReference
 - gazebo::common::SphericalCoordinates, 1060
- GetLength
 - gazebo::common::Animation, 156

- gazebo::common::NodeAnimation, 740
- gazebo::common::SkeletonAnimation, 1032
- gazebo::math::Vector3, 1170
- gazebo::math::Vector4, 1181
- gazebo::physics::CylinderShape, 299
- gazebo::physics::RayShape, 852
- GetLight
 - gazebo::rendering::Scene, 888
- GetLightCount
 - gazebo::rendering::Scene, 888
- GetLighting
 - gazebo::common::Material, 643
- getLights
 - gazebo::rendering::MovableText, 712
- GetLineWidth
 - gazebo::rendering::Grid, 490
- GetLinearAcceleration
 - gazebo::sensors::ImuSensor, 527
- GetLinearDamping
 - gazebo::physics::Link, 606
- GetLink
 - gazebo::physics::Collision, 239
 - gazebo::physics::Model, 684
- GetLinkById
 - gazebo::physics::Model, 685
- GetLinkForce
 - gazebo::physics::DARTJoint, 333
 - gazebo::physics::Joint, 555
 - gazebo::physics::SimbodyJoint, 965
- GetLinkState
 - gazebo::physics::ModelState, 702
- GetLinkStateCount
 - gazebo::physics::ModelState, 702
- GetLinkStates
 - gazebo::physics::ModelState, 703
- GetLinkTorque
 - gazebo::physics::DARTJoint, 333
 - gazebo::physics::Joint, 555
 - gazebo::physics::SimbodyJoint, 965
- GetLinks
 - gazebo::physics::Model, 685
- GetLocalAddress
 - gazebo::transport::Connection, 268
- GetLocalAxis
 - gazebo::physics::Joint, 555
- GetLocalHostname
 - gazebo::transport::Connection, 268
- GetLocalPort
 - gazebo::transport::Connection, 268
- GetLocalURI
 - gazebo::transport::Connection, 269
- GetLocallyAdvertised
 - gazebo::transport::Publication, 815
- GetLog
 - gazebo::math::Quaternion, 830
- GetLogPath
 - gazebo::common::SystemPaths, 1096
 - gazebo::util::DiagnosticManager, 392
- GetLogVersion
 - gazebo::util::LogPlay, 630
- GetLongitude
 - gazebo::sensors::GpsSensor, 466
- GetLongitudeReference
 - gazebo::common::SphericalCoordinates, 1060
- GetLowStop
 - gazebo::physics::DARTBallJoint, 305
 - gazebo::physics::DARTJoint, 333
 - gazebo::physics::DARTScrewJoint, 367
 - gazebo::physics::Joint, 556
 - gazebo::physics::SimbodyBallJoint, 939
 - gazebo::physics::SimbodyJoint, 966
 - gazebo::physics::SimbodyScrewJoint, 1005
- GetLowerLimit
 - gazebo::physics::Joint, 555
- GetMOI
 - gazebo::physics::Inertial, 533
- GetManager
 - gazebo::rendering::Scene, 888
- GetMass
 - gazebo::physics::Inertial, 533
- GetMassProperties
 - gazebo::physics::SimbodyLink, 976
- GetMaterial
 - gazebo::common::Mesh, 664
- getMaterial
 - gazebo::rendering::MovableText, 712
- GetMaterialCount
 - gazebo::common::Mesh, 664
- GetMaterialIndex
 - gazebo::common::SubMesh, 1079
- GetMaterialName
 - gazebo::rendering::Visual, 1206
- GetMax
 - gazebo::common::Mesh, 664
 - gazebo::common::SubMesh, 1079
 - gazebo::math::Vector3, 1170
- GetMaxAngle
 - gazebo::physics::MultiRayShape, 727
- GetMaxColor
 - gazebo::common::Image, 518
- GetMaxContacts
 - gazebo::physics::Collision, 239
 - gazebo::physics::PhysicsEngine, 773
- GetMaxElevation
 - gazebo::common::HeightmapData, 506
 - gazebo::common::ImageHeightmap, 524
- GetMaxForce
 - gazebo::physics::DARTBallJoint, 305

- gazebo::physics::DARTHinge2Joint, 320
- gazebo::physics::DARTHingeJoint, 325
- gazebo::physics::DARTScrewJoint, 367
- gazebo::physics::DARTSliderJoint, 374
- gazebo::physics::DARTUniversalJoint, 381
- gazebo::physics::Joint, 556
- gazebo::physics::SimbodyBallJoint, 939
- gazebo::physics::SimbodyHinge2Joint, 953
- gazebo::physics::SimbodyHingeJoint, 958
- gazebo::physics::SimbodyScrewJoint, 1005
- gazebo::physics::SimbodySliderJoint, 1012
- gazebo::physics::SimbodyUniversalJoint, 1018
- GetMaxFreqFiltered
 - gazebo::sensors::WirelessReceiver, 1231
- GetMaxHeight
 - gazebo::physics::HeightmapShape, 509
- GetMaxIndex
 - gazebo::common::SubMesh, 1079
- GetMaxRange
 - gazebo::physics::MultiRayShape, 727
- GetMaxStepSize
 - gazebo::physics::PhysicsEngine, 773
- GetMean
 - gazebo::sensors::GaussianNoiseModel, 460
- GetMesh
 - gazebo::common::MeshManager, 674
- GetMeshAABB
 - gazebo::common::MeshManager, 674
- GetMeshName
 - gazebo::rendering::Visual, 1206
- GetMeshURI
 - gazebo::physics::MeshShape, 677
- GetMin
 - gazebo::common::Mesh, 664
 - gazebo::common::SubMesh, 1079
 - gazebo::math::Vector3, 1170
- GetMinAngle
 - gazebo::physics::MultiRayShape, 728
- GetMinFreqFiltered
 - gazebo::sensors::WirelessReceiver, 1231
- GetMinHeight
 - gazebo::physics::HeightmapShape, 509
- GetMinRange
 - gazebo::physics::MultiRayShape, 728
- getMinimalComms
 - Transport, 95
- GetMode
 - gazebo::rendering::SelectionObj, 902
- GetModel
 - gazebo::physics::Collision, 240
 - gazebo::physics::Link, 606
 - gazebo::physics::World, 1244
- GetModelBelowPoint
 - gazebo::physics::World, 1244
- GetModelConfig
 - Common, 43
- GetModelCount
 - gazebo::physics::World, 1245
- GetModelFile
 - Common, 43
- GetModelName
 - Common, 43
- GetModelPath
 - Common, 43
- GetModelPaths
 - gazebo::common::SystemPaths, 1096
- GetModelState
 - gazebo::physics::WorldState, 1255
- GetModelStateCount
 - gazebo::physics::WorldState, 1256
- GetModelStates
 - gazebo::physics::WorldState, 1256
- GetModelTransform
 - gazebo::common::SkeletonNode, 1039
- GetModelVisualAt
 - gazebo::rendering::Scene, 889
- GetModels
 - Common, 44
 - gazebo::physics::World, 1245
- GetMouseHit
 - gazebo::rendering::Heightmap, 502
- GetMovableType
 - gazebo::rendering::DynamicLines, 399
 - gazebo::rendering::DynamicRenderable, 402
- getMovableType
 - gazebo::rendering::DynamicLines, 399
- GetMsgType
 - gazebo::transport::CallbackHelper, 193
 - gazebo::transport::CallbackHelperT, 196
 - gazebo::transport::Node, 734
 - gazebo::transport::Publication, 815
 - gazebo::transport::PublicationTransport, 819
 - gazebo::transport::Publisher, 821
 - gazebo::transport::RawCallbackHelper, 841
 - gazebo::transport::SubscribeOptions, 1085
- GetMsgTypes
 - gazebo::msgs::MsgFactory, 717
- GetMuPrimary
 - gazebo::physics::FrictionPyramid, 456
- GetMuSecondary
 - gazebo::physics::FrictionPyramid, 456
- GetName
 - gazebo::common::Material, 643
 - gazebo::common::Mesh, 664
 - gazebo::common::NodeAnimation, 740
 - gazebo::common::SkeletonAnimation, 1033
 - gazebo::common::SkeletonNode, 1040
 - gazebo::common::SubMesh, 1080

- gazebo::physics::Base, 175
- gazebo::physics::Gripper, 493
- gazebo::physics::State, 1070
- gazebo::physics::World, 1245
- gazebo::rendering::Camera, 210
- gazebo::rendering::Light, 591
- gazebo::rendering::Scene, 889
- gazebo::rendering::Visual, 1206
- gazebo::sensors::Sensor, 912
- gazebo::util::DiagnosticTimer, 395
- GetNearClip
 - gazebo::rendering::Camera, 210
 - gazebo::rendering::GpuLaser, 472
- GetNearestEntityBelow
 - gazebo::physics::Entity, 409
- GetNextFrame
 - gazebo::common::Video, 1188
- GetNode
 - gazebo::transport::SubscribeOptions, 1085
- GetNodeAssignment
 - gazebo::common::SubMesh, 1080
- GetNodeAssignmentsCount
 - gazebo::common::SubMesh, 1080
- GetNodeByHandle
 - gazebo::common::Skeleton, 1027
- GetNodeById
 - gazebo::common::Skeleton, 1027
- GetNodeByName
 - gazebo::common::Skeleton, 1028
- GetNodeCount
 - gazebo::common::SkeletonAnimation, 1033
 - gazebo::transport::Publication, 815
- GetNodePoseAt
 - gazebo::common::SkeletonAnimation, 1033
- GetNodes
 - gazebo::common::Skeleton, 1028
- GetNoise
 - gazebo::sensors::Sensor, 913
- GetNoiseType
 - gazebo::sensors::Noise, 750
- GetNormal
 - gazebo::common::SubMesh, 1080
 - gazebo::math::Vector3, 1170
 - gazebo::physics::PlaneShape, 792
- GetNormalCount
 - gazebo::common::Mesh, 664
 - gazebo::common::SubMesh, 1080
- GetNormalMap
 - gazebo::rendering::Visual, 1206
- GetNumAnimations
 - gazebo::common::Skeleton, 1028
- GetNumJoints
 - gazebo::common::Skeleton, 1028
- GetNumNodes
 - gazebo::common::Skeleton, 1028
- GetNumPoints
 - gazebo::math::RotationSpline, 873
- GetNumRawTrans
 - gazebo::common::SkeletonNode, 1040
- GetNumVertNodeWeights
 - gazebo::common::Skeleton, 1028
- GetOgreCamera
 - gazebo::rendering::Camera, 210
- GetOgrePaths
 - gazebo::common::SystemPaths, 1096
- GetOgreTerrain
 - gazebo::rendering::Heightmap, 502
- GetOnContact
 - gazebo::util::OpenALSource, 760
- GetOperationType
 - gazebo::rendering::DynamicRenderable, 402
- GetOrientation
 - gazebo::sensors::ImuSensor, 527
- GetOutgoingCount
 - gazebo::transport::Publisher, 821
- GetPGain
 - gazebo::common::PID, 785
- GetPSSMShadowCameraSetup
 - gazebo::rendering::RTShaderSystem, 878
- GetParam
 - gazebo::physics::DARTJoint, 334
 - gazebo::physics::DARTPhysics, 357, 358
 - gazebo::physics::Joint, 556
 - gazebo::physics::PhysicsEngine, 773
 - gazebo::physics::SimbodyJoint, 966
 - gazebo::physics::SimbodyPhysics, 991
 - gazebo::physics::SimbodyScrewJoint, 1006
- GetParent
 - gazebo::common::SkeletonNode, 1040
 - gazebo::physics::Base, 175
 - gazebo::physics::Joint, 557
 - gazebo::rendering::Projector, 811
 - gazebo::rendering::Visual, 1206
- GetParentId
 - gazebo::physics::Base, 175
 - gazebo::sensors::Sensor, 913
- GetParentJoints
 - gazebo::physics::Link, 606
- GetParentJointsLinks
 - gazebo::physics::Link, 606
- GetParentModel
 - gazebo::physics::Entity, 409
- GetParentName
 - gazebo::sensors::Sensor, 913
- GetParentWorldPose
 - gazebo::physics::Joint, 557
- GetPath
 - gazebo::common::Mesh, 665

- GetPauseTime
 - gazebo::physics::World, 1245
- GetPaused
 - gazebo::util::LogRecord, 635
- GetPerpendicular
 - gazebo::math::Vector3, 1170
- GetPhysicsEngine
 - gazebo::physics::World, 1245
- GetPhysicsUpdateMutex
 - gazebo::physics::PhysicsEngine, 773
- GetPitch
 - gazebo::common::Image, 518
 - gazebo::math::Quaternion, 830
- GetPitchNode
 - gazebo::rendering::Camera, 210
- GetPixel
 - gazebo::common::Image, 518
- GetPixelFormat
 - gazebo::common::Image, 518
- GetPluginCount
 - gazebo::physics::Model, 685
- GetPluginPaths
 - gazebo::common::SystemPaths, 1096
- GetPoint
 - gazebo::math::RotationSpline, 873
 - gazebo::math::Spline, 1065
 - gazebo::rendering::DynamicLines, 399
- GetPointCount
 - gazebo::math::Spline, 1065
 - gazebo::rendering::DynamicLines, 399
- GetPointSize
 - gazebo::common::Material, 643
- GetPos
 - gazebo::physics::HeightmapShape, 509
- GetPose
 - gazebo::physics::CollisionState, 247
 - gazebo::physics::Inertial, 534
 - gazebo::physics::LinkState, 622
 - gazebo::physics::ModelState, 703
 - gazebo::physics::SimbodyPhysics, 991
 - gazebo::rendering::Visual, 1207
 - gazebo::sensors::Sensor, 913
- GetPoseAt
 - gazebo::common::SkeletonAnimation, 1033
- GetPoseAtX
 - gazebo::common::SkeletonAnimation, 1034
- GetPosition
 - gazebo::rendering::Light, 591
 - gazebo::rendering::Visual, 1207
- GetPositionPIDs
 - gazebo::physics::JointController, 570
- GetPositions
 - gazebo::physics::JointController, 570
- GetPower
 - gazebo::sensors::WirelessTransceiver, 1234
- GetPrevMsg
 - gazebo::transport::Publication, 815
 - gazebo::transport::Publisher, 821
- GetPrevMsgPtr
 - gazebo::transport::Publisher, 822
- GetPrimitiveType
 - gazebo::common::SubMesh, 1080
- GetPrincipalMoments
 - gazebo::physics::Inertial, 534
- GetProductsofInertia
 - gazebo::physics::Inertial, 534
- GetQuiet
 - gazebo::common::Console, 278
- GetRGBData
 - gazebo::common::Image, 519
- GetRadius
 - gazebo::physics::CylinderShape, 299
 - gazebo::physics::SphereShape, 1056
 - gazebo::sensors::SonarSensor, 1049
- GetRandSeed
 - gazebo::util::LogPlay, 630
- GetRange
 - gazebo::physics::MultiRayShape, 728
 - gazebo::sensors::GpuRaySensor, 482
 - gazebo::sensors::RaySensor, 845
 - gazebo::sensors::SonarSensor, 1049
- GetRangeCount
 - gazebo::sensors::GpuRaySensor, 482
 - gazebo::sensors::RaySensor, 846
- GetRangeCountRatio
 - gazebo::sensors::GpuRaySensor, 482
- GetRangeMax
 - gazebo::sensors::GpuRaySensor, 482
 - gazebo::sensors::RaySensor, 846
 - gazebo::sensors::SonarSensor, 1049
- GetRangeMin
 - gazebo::sensors::GpuRaySensor, 483
 - gazebo::sensors::RaySensor, 846
 - gazebo::sensors::SonarSensor, 1049
- GetRangeResolution
 - gazebo::sensors::GpuRaySensor, 483
 - gazebo::sensors::RaySensor, 846
- GetRanges
 - gazebo::sensors::GpuRaySensor, 483
 - gazebo::sensors::RaySensor, 846
- GetRawTransform
 - gazebo::common::SkeletonNode, 1040
- GetRawTransforms
 - gazebo::common::SkeletonNode, 1040
- GetRayCount
 - gazebo::sensors::GpuRaySensor, 483
 - gazebo::sensors::RaySensor, 846
- GetRayCountRatio

- gazebo::rendering::GpuLaser, 472
- gazebo::sensors::GpuRaySensor, 483
- GetRealTime
 - gazebo::physics::State, 1070
 - gazebo::physics::World, 1245
- GetRealTimeUpdateRate
 - gazebo::physics::PhysicsEngine, 773
- GetRelativeAngularAccel
 - gazebo::physics::Collision, 240
 - gazebo::physics::Entity, 409
 - gazebo::physics::Link, 607
 - gazebo::physics::Model, 685
- GetRelativeAngularVel
 - gazebo::physics::Collision, 240
 - gazebo::physics::Entity, 409
 - gazebo::physics::Link, 607
 - gazebo::physics::Model, 685
- GetRelativeForce
 - gazebo::physics::Link, 607
- GetRelativeLinearAccel
 - gazebo::physics::Collision, 240
 - gazebo::physics::Entity, 409
 - gazebo::physics::Link, 607
 - gazebo::physics::Model, 686
- GetRelativeLinearVel
 - gazebo::physics::Collision, 240
 - gazebo::physics::Entity, 410
 - gazebo::physics::Link, 607
 - gazebo::physics::Model, 686
- GetRelativePoints
 - gazebo::physics::RayShape, 852
- GetRelativePose
 - gazebo::physics::Entity, 410
- GetRelativeTorque
 - gazebo::physics::Link, 607
- GetRemoteAddress
 - gazebo::transport::Connection, 269
- GetRemoteHostname
 - gazebo::transport::Connection, 269
- GetRemotePort
 - gazebo::transport::Connection, 269
- GetRemoteSubscriptionCount
 - gazebo::transport::Publication, 815
- GetRemoteURI
 - gazebo::transport::Connection, 269
- getRenderOperation
 - gazebo::rendering::MovableText, 712
- GetRenderPathType
 - gazebo::rendering::RenderEngine, 857
- GetRenderRate
 - gazebo::rendering::Camera, 211
- GetRenderTexture
 - gazebo::rendering::Camera, 211
- GetResRange
 - gazebo::physics::MultiRayShape, 728
- GetRetro
 - gazebo::physics::MultiRayShape, 728
 - gazebo::physics::RayShape, 852
 - gazebo::sensors::GpuRaySensor, 483
 - gazebo::sensors::RaySensor, 847
- GetRight
 - gazebo::rendering::Camera, 211
- GetRoll
 - gazebo::math::Quaternion, 830
- GetRootNode
 - gazebo::common::Skeleton, 1029
- GetRootVisual
 - gazebo::rendering::Visual, 1207
- GetRotation
 - gazebo::common::PoseKeyFrame, 809
 - gazebo::math::Matrix4, 656
 - gazebo::rendering::Visual, 1207
- GetRounded
 - gazebo::math::Vector3, 1171
- GetRunTime
 - gazebo::util::LogRecord, 635
- GetRunning
 - gazebo::common::Timer, 1122
 - gazebo::physics::World, 1246
 - gazebo::util::LogRecord, 635
- GetSDF
 - gazebo::physics::Actor, 142
 - gazebo::physics::Base, 176
 - gazebo::physics::Model, 686
- GetSID
 - gazebo::common::NodeTransform, 745
- GetSORPGSIlters
 - gazebo::physics::PhysicsEngine, 774
- GetSORPGSPreconIlters
 - gazebo::physics::PhysicsEngine, 774
- GetSORPGSW
 - gazebo::physics::PhysicsEngine, 774
- GetSampleCount
 - gazebo::physics::MultiRayShape, 729
- GetSampleRate
 - gazebo::common::AudioDecoder, 163
- GetSaveable
 - gazebo::physics::Base, 175
- GetScale
 - gazebo::physics::Shape, 934
 - gazebo::rendering::Visual, 1207
- GetScanResolution
 - gazebo::physics::MultiRayShape, 729
- GetScene
 - gazebo::rendering::Camera, 211
 - gazebo::rendering::RenderEngine, 857
 - gazebo::rendering::Visual, 1207
- GetSceneCount

- gazebo::rendering::RenderEngine, 858
- GetSceneMsg
 - gazebo::physics::World, 1246
- GetSceneNode
 - gazebo::rendering::Camera, 211
 - gazebo::rendering::Grid, 490
 - gazebo::rendering::Visual, 1208
- GetScopedName
 - gazebo::physics::Base, 176
 - gazebo::rendering::Camera, 211
 - gazebo::sensors::Sensor, 913
- GetScreenshotPath
 - gazebo::rendering::Camera, 212
- GetSeed
 - gazebo::math::Rand, 839
- GetSelectedEntity
 - gazebo::physics::World, 1246
- GetSelectedVisual
 - gazebo::rendering::Scene, 889
- GetSelfCollide
 - gazebo::physics::Link, 608
- GetSensitivity
 - gazebo::sensors::WirelessReceiver, 1231
- GetSensor
 - gazebo::sensors::SensorManager, 922
- GetSensorCount
 - gazebo::physics::Link, 608
 - gazebo::physics::Model, 686
- GetSensorName
 - gazebo::physics::Link, 608
- GetSensorTypes
 - gazebo::sensors::SensorFactory, 919
 - gazebo::sensors::SensorManager, 923
- GetSensors
 - gazebo::sensors::SensorManager, 923
- GetSetWorldPoseMutex
 - gazebo::physics::World, 1246
- GetShadeMode
 - gazebo::common::Material, 644
- GetShaderType
 - gazebo::rendering::Visual, 1208
- GetShadowsEnabled
 - gazebo::rendering::Scene, 889
- GetShape
 - gazebo::physics::Collision, 241
- GetShapeType
 - gazebo::physics::Collision, 241
- GetShininess
 - gazebo::common::Material, 644
- GetShowClouds
 - gazebo::rendering::Scene, 889
- GetShowOnTop
 - gazebo::rendering::MovableText, 712
- GetSignalStrength
 - gazebo::sensors::WirelessTransmitter, 1237
- GetSignaled
 - gazebo::event::Event, 418
- GetSimTime
 - gazebo::physics::State, 1070
 - gazebo::physics::World, 1246
 - gazebo::rendering::Scene, 889
- GetSize
 - gazebo::math::Box, 182
 - gazebo::physics::BoxShape, 187
 - gazebo::physics::HeightmapShape, 510
 - gazebo::physics::MeshShape, 677
 - gazebo::physics::PlaneShape, 793
- GetSkeleton
 - gazebo::common::Mesh, 665
- GetSpaceWidth
 - gazebo::rendering::MovableText, 712
- GetSpecular
 - gazebo::common::Material, 644
- GetSpecularColor
 - gazebo::rendering::Light, 591
- GetSphericalCoordinates
 - gazebo::physics::World, 1246
- GetSpringReferencePosition
 - gazebo::physics::Joint, 557
- GetSquaredLength
 - gazebo::math::Vector3, 1171
 - gazebo::math::Vector4, 1181
- getSquaredViewDepth
 - gazebo::rendering::DynamicRenderable, 402
 - gazebo::rendering::MovableText, 712
- GetStartTime
 - gazebo::physics::World, 1247
- GetState
 - gazebo::physics::Collision, 241
 - gazebo::rendering::SelectionObj, 902
- GetStdDev
 - gazebo::sensors::GaussianNoiseModel, 460
- GetStiffness
 - gazebo::physics::Joint, 557
- GetStopDissipation
 - gazebo::physics::Joint, 558
- GetStopStiffness
 - gazebo::physics::Joint, 558
- GetSubMesh
 - gazebo::common::Mesh, 665
- GetSubMeshCount
 - gazebo::common::Mesh, 665
- GetSubMeshName
 - gazebo::rendering::Visual, 1208
- GetSubSampling
 - gazebo::physics::HeightmapShape, 510
- GetSum
 - gazebo::math::Vector3, 1171

- GetSurface
 - gazebo::physics::Collision, 241
- GetSurfaceType
 - gazebo::common::SphericalCoordinates, 1060
- GetTagPose
 - gazebo::sensors::RFIDTag, 863
- GetTangent
 - gazebo::math::Spline, 1065
- GetTargetRealTimeFactor
 - gazebo::physics::PhysicsEngine, 774
- GetTension
 - gazebo::math::Spline, 1065
- GetTerrainSubdivisionCount
 - gazebo::rendering::Heightmap, 502
- GetTexCoord
 - gazebo::common::SubMesh, 1081
- GetTexCoordCount
 - gazebo::common::Mesh, 666
 - gazebo::common::SubMesh, 1081
- GetText
 - gazebo::rendering::MovableText, 713
- GetTextureHeight
 - gazebo::rendering::Camera, 212
- GetTextureImage
 - gazebo::common::Material, 644
- GetTextureWidth
 - gazebo::rendering::Camera, 212
- GetThreadPitch
 - gazebo::physics::DARTScrewJoint, 368
 - gazebo::physics::ScrewJoint, 898
 - gazebo::physics::SimbodyScrewJoint, 1006
- GetTime
 - gazebo::common::Animation, 156
 - gazebo::common::KeyFrame, 585
 - gazebo::util::DiagnosticManager, 392
- GetTimeAtX
 - gazebo::common::NodeAnimation, 740
- GetTimerCount
 - gazebo::util::DiagnosticManager, 392
- GetTplInstancePath
 - gazebo::common::SystemPaths, 1096
- GetTmpPath
 - gazebo::common::SystemPaths, 1097
- GetTopic
 - gazebo::sensors::CameraSensor, 230
 - gazebo::sensors::ForceTorqueSensor, 451
 - gazebo::sensors::GpuRaySensor, 484
 - gazebo::sensors::MultiCameraSensor, 722
 - gazebo::sensors::RaySensor, 847
 - gazebo::sensors::Sensor, 914
 - gazebo::sensors::SonarSensor, 1049
 - gazebo::sensors::WirelessTransceiver, 1234
 - gazebo::transport::PublicationTransport, 819
 - gazebo::transport::Publisher, 822
 - gazebo::transport::SubscribeOptions, 1085
 - gazebo::transport::Subscriber, 1087
- getTopicMsgType
 - Transport, 95
- GetTopicNamespace
 - gazebo::transport::Node, 734
- GetTopicNamespaces
 - gazebo::transport::ConnectionManager, 273
 - gazebo::transport::TopicManager, 1126
- GetTorque
 - gazebo::sensors::ForceTorqueSensor, 452
- GetTransform
 - gazebo::common::SkeletonNode, 1041
- GetTransforms
 - gazebo::common::SkeletonNode, 1041
- GetTranslation
 - gazebo::common::PoseKeyFrame, 809
 - gazebo::math::Matrix4, 656
- GetTransparency
 - gazebo::common::Material, 644
 - gazebo::rendering::Visual, 1208
- GetTransportCount
 - gazebo::transport::Publication, 816
- GetTriangleCount
 - gazebo::rendering::Camera, 212
 - gazebo::rendering::UserCamera, 1141
 - gazebo::rendering::WindowManager, 1226
- GetType
 - gazebo::common::NodeTransform, 745
 - gazebo::physics::Base, 176
 - gazebo::physics::DARTPhysics, 358
 - gazebo::physics::PhysicsEngine, 774
 - gazebo::physics::SimbodyPhysics, 991
 - gazebo::PluginT, 796
 - gazebo::rendering::Light, 592
 - gazebo::sensors::Sensor, 914
- GetTypeString
 - gazebo::physics::SimbodyPhysics, 991
 - gazebo::rendering::FPSViewController, 454
 - gazebo::rendering::OrbitViewController, 764
 - gazebo::rendering::ViewController, 1194
- GetURI
 - Common, 44
 - gazebo::physics::HeightmapShape, 510
- getUniqueld
 - Classes for physics and dynamics, 73
- GetUp
 - gazebo::rendering::Camera, 212
- GetUpdatePeriod
 - gazebo::physics::PhysicsEngine, 774
- GetUpdateRate
 - gazebo::sensors::Sensor, 914
- GetUpperLimit
 - gazebo::physics::Joint, 558

- GetUserCamera
 - gazebo::rendering::Scene, 890
- GetUserCameraCount
 - gazebo::rendering::Scene, 890
- GetVFOV
 - gazebo::rendering::Camera, 212
- GetValue
 - gazebo::common::NumericKeyFrame, 755
- GetVelocities
 - gazebo::physics::JointController, 570
- GetVelocity
 - gazebo::physics::DARTBallJoint, 305
 - gazebo::physics::DARTHinge2Joint, 321
 - gazebo::physics::DARTHingeJoint, 325
 - gazebo::physics::DARTScrewJoint, 368
 - gazebo::physics::DARTSliderJoint, 374
 - gazebo::physics::DARTUniversalJoint, 381
 - gazebo::physics::Joint, 558
 - gazebo::physics::LinkState, 622
 - gazebo::physics::SimbodyBallJoint, 939
 - gazebo::physics::SimbodyHinge2Joint, 953
 - gazebo::physics::SimbodyHingeJoint, 958
 - gazebo::physics::SimbodyScrewJoint, 1006
 - gazebo::physics::SimbodySliderJoint, 1012
 - gazebo::physics::SimbodyUniversalJoint, 1019
- GetVelocityLimit
 - gazebo::physics::Joint, 559
- GetVelocityPIDs
 - gazebo::physics::JointController, 570
- GetVertFOV
 - gazebo::rendering::GpuLaser, 472
 - gazebo::sensors::GpuRaySensor, 484
- GetVertHalfAngle
 - gazebo::rendering::GpuLaser, 472
 - gazebo::sensors::GpuRaySensor, 484
- GetVertNodeWeight
 - gazebo::common::Skeleton, 1029
- GetVertex
 - gazebo::common::SubMesh, 1081
- GetVertexCount
 - gazebo::common::Mesh, 666
 - gazebo::common::SubMesh, 1081
 - gazebo::physics::HeightmapShape, 510
- GetVertexIndex
 - gazebo::common::SubMesh, 1081
- GetVerticalAngleMax
 - gazebo::sensors::GpuRaySensor, 484
 - gazebo::sensors::RaySensor, 847
- GetVerticalAngleMin
 - gazebo::sensors::GpuRaySensor, 484
 - gazebo::sensors::RaySensor, 847
- GetVerticalAngleResolution
 - gazebo::sensors::GpuRaySensor, 485
 - gazebo::sensors::RaySensor, 847
- GetVerticalMaxAngle
 - gazebo::physics::MultiRayShape, 729
- GetVerticalMinAngle
 - gazebo::physics::MultiRayShape, 729
- GetVerticalRangeCount
 - gazebo::sensors::GpuRaySensor, 485
 - gazebo::sensors::RaySensor, 848
- GetVerticalRayCount
 - gazebo::sensors::GpuRaySensor, 485
 - gazebo::sensors::RaySensor, 848
- GetVerticalSampleCount
 - gazebo::physics::MultiRayShape, 729
- GetVerticalScanResolution
 - gazebo::physics::MultiRayShape, 729
- GetViewControllerTypeString
 - gazebo::rendering::UserCamera, 1142
- GetViewport
 - gazebo::rendering::Camera, 213
- GetViewportHeight
 - gazebo::rendering::Camera, 213
- GetViewportWidth
 - gazebo::rendering::Camera, 213
- GetVisibilityFlags
 - gazebo::rendering::Visual, 1208
- GetVisible
 - gazebo::rendering::Light, 592
 - gazebo::rendering::Visual, 1208
 - gazebo::rendering::WireBox, 1228
- GetVisual
 - gazebo::rendering::Scene, 890
 - gazebo::rendering::UserCamera, 1142
- GetVisualAt
 - gazebo::rendering::Scene, 891
- GetVisualBelow
 - gazebo::rendering::Scene, 891
- GetVisualCount
 - gazebo::rendering::Scene, 891
- GetVisualize
 - gazebo::sensors::Sensor, 914
- GetVisualsBelowPoint
 - gazebo::rendering::Scene, 892
- GetWallTime
 - gazebo::common::Time, 1104
 - gazebo::physics::State, 1070
- GetWallTimeAsISOString
 - gazebo::common::Time, 1105
- GetWidth
 - gazebo::common::HeightmapData, 506
 - gazebo::common::Image, 519
 - gazebo::common::ImageHeightmap, 525
 - gazebo::common::Video, 1189
- GetWindow
 - gazebo::rendering::WindowManager, 1226
- GetWindowFilled

- Common, 44
- GetWindowId
 - gazebo::rendering::Camera, 213
- GetWindowManager
 - gazebo::rendering::RenderEngine, 858
- GetWindowSize
 - Common, 45
- GetWorld
 - gazebo::physics::Base, 176
- GetWorldAngularAccel
 - gazebo::physics::Collision, 241
 - gazebo::physics::Entity, 410
 - gazebo::physics::Link, 608
 - gazebo::physics::Model, 686
- GetWorldAngularVel
 - gazebo::physics::Collision, 242
 - gazebo::physics::DARTLink, 343
 - gazebo::physics::Entity, 410
 - gazebo::physics::Model, 687
 - gazebo::physics::SimbodyLink, 976
- GetWorldCFM
 - gazebo::physics::PhysicsEngine, 775
- GetWorldCoGLinearVel
 - gazebo::physics::DARTLink, 343
 - gazebo::physics::Link, 609
 - gazebo::physics::SimbodyLink, 976
- GetWorldCoGPose
 - gazebo::physics::Link, 609
- GetWorldERP
 - gazebo::physics::PhysicsEngine, 775
- GetWorldEnergy
 - gazebo::physics::Link, 609
 - gazebo::physics::Model, 687
- GetWorldEnergyKinetic
 - gazebo::physics::Link, 609
 - gazebo::physics::Model, 687
- GetWorldEnergyPotential
 - gazebo::physics::Link, 609
 - gazebo::physics::Model, 687
- GetWorldEnergyPotentialSpring
 - gazebo::physics::Joint, 559
- GetWorldForce
 - gazebo::physics::DARTLink, 344
 - gazebo::physics::Link, 609
 - gazebo::physics::SimbodyLink, 976
- GetWorldInertiaMatrix
 - gazebo::physics::Link, 610
- GetWorldInertialPose
 - gazebo::physics::Link, 610
- GetWorldLinearAccel
 - gazebo::physics::Collision, 242
 - gazebo::physics::Entity, 410
 - gazebo::physics::Link, 610
 - gazebo::physics::Model, 687
- GetWorldLinearVel
 - gazebo::physics::Collision, 242
 - gazebo::physics::DARTLink, 344
 - gazebo::physics::Entity, 411
 - gazebo::physics::Link, 610, 611
 - gazebo::physics::Model, 688
 - gazebo::physics::SimbodyLink, 977
- GetWorldName
 - gazebo::sensors::Sensor, 914
- GetWorldPathExtension
 - gazebo::common::SystemPaths, 1097
- GetWorldPointOnPlane
 - gazebo::rendering::Camera, 213
- GetWorldPose
 - gazebo::physics::Entity, 411
 - gazebo::physics::Joint, 559
 - gazebo::rendering::Camera, 214
 - gazebo::rendering::Visual, 1209
- GetWorldPosition
 - gazebo::rendering::Camera, 214
- GetWorldRotation
 - gazebo::rendering::Camera, 214
- GetWorldTorque
 - gazebo::physics::DARTLink, 344
 - gazebo::physics::Link, 611
 - gazebo::physics::SimbodyLink, 977
- getWorldTransforms
 - gazebo::rendering::MovableText, 713
- GetWorldVisual
 - gazebo::rendering::Scene, 892
- GetWrench
 - gazebo::physics::LinkState, 622
- GetXAxis
 - gazebo::math::Quaternion, 830
- GetXLength
 - gazebo::math::Box, 183
- GetYAxis
 - gazebo::math::Quaternion, 831
- GetYLength
 - gazebo::math::Box, 183
- GetYaw
 - gazebo::math::Quaternion, 831
- GetZAxis
 - gazebo::math::Quaternion, 831
- GetZLength
 - gazebo::math::Box, 183
- GetZValue
 - gazebo::rendering::Camera, 214
- globalEndPos
 - gazebo::physics::RayShape, 854
- GlobalFromLocal
 - gazebo::common::SphericalCoordinates, 1061
- globalStartPos
 - gazebo::physics::RayShape, 854

- google, 137
- google::protobuf, 137
- google::protobuf::compiler, 137
- google::protobuf::compiler::cpp, 137
- google::protobuf::compiler::cpp::GazeboGenerator, 461
 - ~GazeboGenerator, 461
 - GazeboGenerator, 461
 - Generate, 461
- GpsSensor
 - gazebo::sensors::GpsSensor, 465
- GpsSensor.hh, 1339
- GpsSensorPtr
 - gazebo::sensors, 132
- GpuLaser
 - gazebo::rendering::GpuLaser, 470
- GpuLaser.hh, 1339
- GpuLaserPtr
 - gazebo::rendering, 128
- GpuRaySensor
 - gazebo::sensors::GpuRaySensor, 479
- GpuRaySensor.hh, 1340
- GpuRaySensor_V
 - gazebo::sensors, 132
- GpuRaySensorPtr
 - gazebo::sensors, 132
- gravity
 - gazebo::physics::SimbodyPhysics, 997
- Green
 - gazebo::common::Color, 260
- Grid
 - gazebo::rendering::Grid, 489
- Grid.hh, 1341
- gridMsg
 - gazebo::rendering::TransmitterVisualPrivate, 1134
- Gripper
 - gazebo::physics::Gripper, 493
- Gripper.hh, 1341
- GripperPtr
 - gazebo::physics, 122
- GtsSurface
 - MeshCSG.hh, 1380
- gui
 - gazebo::rendering::UserCameraPrivate, 1147
- gzLogInit
 - Common, 40
- GzTerrainMatGen
 - gazebo::rendering::GzTerrainMatGen, 499
- gzdbg
 - Common, 39
- gzerr
 - Common, 39
- gzlog
 - Common, 40
- gzmsg
 - Common, 40
- gzthrow
 - Common, 40
- gzwarn
 - Common, 40
- H_CENTER
 - gazebo::rendering::MovableText, 711
- H_LEFT
 - gazebo::rendering::MovableText, 711
- HEADER_LENGTH
 - Connection.hh, 1299
- HEIGHTMAP_SHAPE
 - gazebo::physics::Base, 172
- HI_STOP
 - gazebo::physics::Joint, 547
- HINGE2_JOINT
 - gazebo::physics::Base, 172
- HINGE_JOINT
 - gazebo::physics::Base, 172
- HalfPi
 - gazebo::math::Angle, 153
- handle
 - gazebo::common::SkeletonNode, 1043
 - gazebo::PluginT, 796
- HandleData
 - gazebo::transport::CallbackHelper, 193
 - gazebo::transport::CallbackHelperT, 196
 - gazebo::transport::Node, 734
 - gazebo::transport::RawCallbackHelper, 841
 - gazebo::transport::SubscriptionTransport, 1089
- HandleKeyPressEvent
 - gazebo::rendering::FPSViewController, 454
 - gazebo::rendering::GUIOverlay, 496
 - gazebo::rendering::OrbitViewController, 765
 - gazebo::rendering::UserCamera, 1142
 - gazebo::rendering::ViewController, 1194
- HandleKeyReleaseEvent
 - gazebo::rendering::FPSViewController, 455
 - gazebo::rendering::GUIOverlay, 496
 - gazebo::rendering::OrbitViewController, 765
 - gazebo::rendering::UserCamera, 1142
 - gazebo::rendering::ViewController, 1194
- HandleMessage
 - gazebo::transport::CallbackHelper, 194
 - gazebo::transport::CallbackHelperT, 197
 - gazebo::transport::Node, 734
 - gazebo::transport::RawCallbackHelper, 841
 - gazebo::transport::SubscriptionTransport, 1089
- HandleMouseEvent
 - gazebo::rendering::FPSViewController, 455
 - gazebo::rendering::GUIOverlay, 496
 - gazebo::rendering::OrbitViewController, 765
 - gazebo::rendering::UserCamera, 1143

- gazebo::rendering::ViewController, 1195
- HasAttachedObject
 - gazebo::rendering::Visual, 1209
- HasCollisionName
 - gazebo::util::OpenALSource, 760
- HasConnections
 - gazebo::transport::Publisher, 822
- HasFilter
 - gazebo::physics::ContactManager, 285
- HasJointState
 - gazebo::physics::ModelState, 703
- HasLatchedSubscriber
 - gazebo::transport::Node, 735
- HasLinkState
 - gazebo::physics::ModelState, 703
- HasMesh
 - gazebo::common::MeshManager, 674
- HasModel
 - Common, 45
- HasModelState
 - gazebo::physics::WorldState, 1256
- HasNode
 - gazebo::common::SkeletonAnimation, 1034
- HasSkeleton
 - gazebo::common::Mesh, 666
- HasTransport
 - gazebo::transport::Publication, 816
- HasType
 - gazebo::physics::Base, 176
- HasVertex
 - gazebo::common::SubMesh, 1081
- headNode
 - gazebo::rendering::ArrowVisualPrivate, 160
- headingOffset
 - gazebo::common::SphericalCoordinatesPrivate, 1063
- height
 - gazebo::rendering::VideoVisualPrivate, 1192
- Heightmap
 - gazebo::rendering::Heightmap, 501
- Heightmap.hh, 1344
- heightmapData
 - gazebo::physics::HeightmapShape, 511
- HeightmapData.hh, 1345
- HeightmapShape
 - gazebo::physics::HeightmapShape, 508
- HeightmapShape.hh, 1346
- HeightmapShapePtr
 - gazebo::physics, 122
- heights
 - gazebo::physics::HeightmapShape, 511
- Helpers.hh, 1347
 - GZ_DBL_MAX, 1349
 - GZ_DBL_MIN, 1349
 - GZ_FLT_MAX, 1349
 - GZ_FLT_MIN, 1349
 - GZ_INT32_MAX, 1349
 - GZ_INT32_MIN, 1349
 - GZ_UINT32_MAX, 1349
 - GZ_UINT32_MIN, 1349
- hfov
 - gazebo::rendering::GpuLaser, 476
- Hide
 - gazebo::rendering::GUIOverlay, 497
- Hinge2Joint
 - gazebo::physics::Hinge2Joint, 512
- Hinge2Joint.hh, 1350
- HingeJoint
 - gazebo::physics::HingeJoint, 514
- HingeJoint.hh, 1351
- HorizAlign
 - gazebo::rendering::MovableText, 711
- horzElem
 - gazebo::physics::MultiRayShape, 730
 - gazebo::sensors::GpuRaySensor, 487
- horzHalfAngle
 - gazebo::rendering::GpuLaser, 476
- horzRangeCount
 - gazebo::sensors::GpuRaySensor, 487
- horzRayCount
 - gazebo::sensors::GpuRaySensor, 487
- IDENTITY
 - gazebo::math::Matrix3, 653
 - gazebo::math::Matrix4, 660
- IMAGE
 - gazebo::sensors, 133
- INTERSECTION
 - gazebo::common::MeshCSG, 668
- IOManager
 - gazebo::transport::IOManager, 540
- IOManager.hh, 1355
- id
 - gazebo::common::SkeletonNode, 1043
 - gazebo::event::ConnectionPrivate, 276
 - gazebo::physics::TrajectoryInfo, 1130
 - gazebo::rendering::VisualPrivate, 1222
- Image
 - gazebo::common::Image, 516
- Image.hh, 1352
- imageBuffer
 - gazebo::rendering::VideoVisualPrivate, 1192
- imageFormat
 - gazebo::rendering::Camera, 222
- ImageGaussianNoiseModel
 - gazebo::sensors::ImageGaussianNoiseModel, 521
- ImageGaussianNoiseModelPtr
 - gazebo::sensors, 132

- imageHeight
 - gazebo::rendering::Camera, 222
- ImageHeightmap
 - gazebo::common::ImageHeightmap, 524
- ImageHeightmap.hh, 1352
- imageWidth
 - gazebo::rendering::Camera, 222
- img
 - gazebo::physics::HeightmapShape, 511
- ImuSensor
 - gazebo::sensors::ImuSensor, 527
- ImuSensor.hh, 1354
- ImuSensor_V
 - gazebo::sensors, 132
- ImuSensorPtr
 - gazebo::sensors, 132
- IncCount
 - gazebo::transport::IOManager, 541
- indexBufferCapacity
 - gazebo::rendering::DynamicRenderable, 404
- Inertial
 - gazebo::physics::Inertial, 531
- inertial
 - gazebo::physics::Link, 617
- Inertial.hh, 1354
- InertialPtr
 - gazebo::physics, 122
- Init
 - gazebo::common::FileLogger, 448
 - gazebo::common::PID, 786
 - gazebo::Master, 638
 - gazebo::ModelPlugin, 697
 - gazebo::physics::Actor, 142
 - gazebo::physics::BallJoint, 168
 - gazebo::physics::Base, 177
 - gazebo::physics::BoxShape, 187
 - gazebo::physics::Collision, 242
 - gazebo::physics::ContactManager, 285
 - gazebo::physics::CylinderShape, 300
 - gazebo::physics::DARTBallJoint, 306
 - gazebo::physics::DARTCollision, 312
 - gazebo::physics::DARTHeightmapShape, 317
 - gazebo::physics::DARTHinge2Joint, 321
 - gazebo::physics::DARTHingeJoint, 326
 - gazebo::physics::DARTJoint, 334
 - gazebo::physics::DARTLink, 345
 - gazebo::physics::DARTMeshShape, 349
 - gazebo::physics::DARTModel, 352
 - gazebo::physics::DARTPhysics, 358
 - gazebo::physics::DARTScrewJoint, 369
 - gazebo::physics::DARTSliderJoint, 374
 - gazebo::physics::DARTUniversalJoint, 381
 - gazebo::physics::GearboxJoint, 463
 - gazebo::physics::Gripper, 493
 - gazebo::physics::HeightmapShape, 510
 - gazebo::physics::HingeJoint, 514
 - gazebo::physics::Joint, 559
 - gazebo::physics::Link, 611
 - gazebo::physics::MeshShape, 677
 - gazebo::physics::Model, 688
 - gazebo::physics::MultiRayShape, 729
 - gazebo::physics::PhysicsEngine, 775
 - gazebo::physics::PlaneShape, 793
 - gazebo::physics::RayShape, 853
 - gazebo::physics::Road, 870
 - gazebo::physics::ScrewJoint, 898
 - gazebo::physics::Shape, 935
 - gazebo::physics::SimbodyHeightmapShape, 950
 - gazebo::physics::SimbodyLink, 977
 - gazebo::physics::SimbodyMeshShape, 982
 - gazebo::physics::SimbodyModel, 984
 - gazebo::physics::SimbodyPhysics, 992
 - gazebo::physics::SphereShape, 1056
 - gazebo::physics::UniversalJoint, 1136
 - gazebo::physics::World, 1247
 - gazebo::rendering::Camera, 214
 - gazebo::rendering::DepthCamera, 386
 - gazebo::rendering::DynamicRenderable, 403
 - gazebo::rendering::FPSViewController, 455
 - gazebo::rendering::GpuLaser, 472
 - gazebo::rendering::Grid, 491
 - gazebo::rendering::GUIOverlay, 497
 - gazebo::rendering::OrbitViewController, 765
 - gazebo::rendering::RenderEngine, 858
 - gazebo::rendering::RTShaderSystem, 878
 - gazebo::rendering::Scene, 892
 - gazebo::rendering::UserCamera, 1143
 - gazebo::rendering::ViewController, 1195
 - gazebo::rendering::Visual, 1209
 - gazebo::rendering::WireBox, 1228
 - gazebo::SensorPlugin, 925
 - gazebo::sensors::CameraSensor, 230
 - gazebo::sensors::ContactSensor, 291
 - gazebo::sensors::DepthCameraSensor, 389
 - gazebo::sensors::ForceTorqueSensor, 452
 - gazebo::sensors::GpsSensor, 466
 - gazebo::sensors::GpuRaySensor, 485
 - gazebo::sensors::ImuSensor, 528
 - gazebo::sensors::MultiCameraSensor, 722
 - gazebo::sensors::RaySensor, 848
 - gazebo::sensors::RFIDSensor, 860
 - gazebo::sensors::RFIDTag, 863
 - gazebo::sensors::Sensor, 914
 - gazebo::sensors::SensorManager, 923
 - gazebo::sensors::SonarSensor, 1050
 - gazebo::sensors::WirelessReceiver, 1231
 - gazebo::sensors::WirelessTransceiver, 1234
 - gazebo::sensors::WirelessTransmitter, 1238

- gazebo::SystemPlugin, 1099
- gazebo::transport::ConnectionManager, 273
- gazebo::transport::Node, 735
- gazebo::transport::PublicationTransport, 819
- gazebo::transport::SubscribeOptions, 1086
- gazebo::transport::SubscriptionTransport, 1090
- gazebo::transport::TopicManager, 1127
- gazebo::util::DiagnosticManager, 393
- gazebo::util::LogRecord, 635
- gazebo::VisualPlugin, 1218
- gazebo::WorldPlugin, 1253
- Messages, 63
- init
 - gazebo, 104
 - Rendering, 85
 - Sensors, 90
 - Transport, 96
- init_world
 - Classes for physics and dynamics, 73
- init_worlds
 - Classes for physics and dynamics, 73
- InitForThread
 - gazebo::physics::DARTPhysics, 358
 - gazebo::physics::PhysicsEngine, 775
 - gazebo::physics::SimbodyPhysics, 992
- InitModel
 - gazebo::physics::SimbodyPhysics, 992
- initialTransform
 - gazebo::common::SkeletonNode, 1043
- initialized
 - gazebo::physics::Link, 618
 - gazebo::rendering::Camera, 222
 - gazebo::rendering::GUIOverlayPrivate, 498
 - gazebo::rendering::VisualPrivate, 1222
- InsertLatchedMsg
 - gazebo::transport::Node, 735
- InsertMesh
 - gazebo::rendering::Visual, 1209
- InsertModelFile
 - gazebo::physics::World, 1247
- InsertModelSDF
 - gazebo::physics::World, 1247
- InsertModelString
 - gazebo::physics::World, 1247
- Instance
 - SingletonT, 1024
- integ
 - gazebo::physics::SimbodyPhysics, 997
- InternalError
 - gazebo::common::InternalError, 539
- Interpolate
 - gazebo::math::RotationSpline, 873, 874
 - gazebo::math::Spline, 1066
- interpolateX
 - gazebo::physics::Actor, 143
- invBindTransform
 - gazebo::common::SkeletonNode, 1043
- Inverse
 - gazebo::math::Matrix4, 656
- Invert
 - gazebo::math::Quaternion, 831
- is_stopped
 - Transport, 96
- IsActive
 - gazebo::physics::Actor, 142
 - gazebo::sensors::CameraSensor, 230
 - gazebo::sensors::ContactSensor, 291
 - gazebo::sensors::ForceTorqueSensor, 452
 - gazebo::sensors::GpuRaySensor, 485
 - gazebo::sensors::ImuSensor, 528
 - gazebo::sensors::MultiCameraSensor, 722
 - gazebo::sensors::RaySensor, 848
 - gazebo::sensors::Sensor, 915
 - gazebo::sensors::SonarSensor, 1050
- IsAdvertised
 - gazebo::transport::TopicManager, 1127
- IsAffine
 - gazebo::math::Matrix4, 657
- IsAnimating
 - gazebo::rendering::Camera, 214
- IsAttached
 - gazebo::physics::Gripper, 493
- IsCameraSetInWorldFile
 - gazebo::rendering::UserCamera, 1143
- isCameraSetInWorldFile
 - gazebo::rendering::UserCameraPrivate, 1147
- IsCanonicalLink
 - gazebo::physics::Entity, 411
- IsFinite
 - gazebo::math::Pose, 801
 - gazebo::math::Quaternion, 831
 - gazebo::math::Vector2d, 1150
 - gazebo::math::Vector2i, 1159
 - gazebo::math::Vector3, 1171
 - gazebo::math::Vector4, 1181
- isFirst
 - gazebo::rendering::TransmitterVisualPrivate, 1134
- IsHorizontal
 - gazebo::rendering::GpuLaser, 473
 - gazebo::sensors::GpuRaySensor, 485
- isHorizontal
 - gazebo::rendering::GpuLaser, 476
- IsInitialized
 - gazebo::rendering::GUIOverlay, 497
- IsJoint
 - gazebo::common::SkeletonNode, 1041
- IsLoaded
 - gazebo::physics::World, 1247

- IsLocal
 - gazebo::transport::CallbackHelper, 194
 - gazebo::transport::CallbackHelperT, 197
 - gazebo::transport::RawCallbackHelper, 842
 - gazebo::transport::SubscriptionTransport, 1090
- IsOpen
 - gazebo::transport::Connection, 269
 - gazebo::util::LogPlay, 630
- IsPaused
 - gazebo::physics::World, 1248
- IsPlaceable
 - gazebo::physics::Collision, 242
- IsPlane
 - gazebo::rendering::Visual, 1210
- IsPlaying
 - gazebo::util::OpenALSource, 761
- isPowerOfTwo
 - Math, 53
- IsReadyToStart
 - gazebo::util::LogRecord, 636
- IsRegistered
 - gazebo::physics::PhysicsFactory, 782
- isReversed
 - gazebo::physics::SimbodyJoint, 970
- IsRootNode
 - gazebo::common::SkeletonNode, 1041
- IsRunning
 - gazebo::transport::ConnectionManager, 273
- IsSelected
 - gazebo::physics::Base, 177
- IsStatic
 - gazebo::physics::Entity, 411
 - gazebo::rendering::Visual, 1210
- isStatic
 - gazebo::rendering::VisualPrivate, 1222
- IsValidFilename
 - gazebo::common::MeshManager, 674
- IsVisible
 - gazebo::rendering::Camera, 215
- IsZero
 - gazebo::physics::CollisionState, 248
 - gazebo::physics::JointState, 577
 - gazebo::physics::LinkState, 622
 - gazebo::physics::ModelState, 704
 - gazebo::physics::WorldState, 1256
- isnan
 - Math, 53
- JOINT
 - gazebo::common::SkeletonNode, 1037
 - gazebo::physics::Base, 172
- Joint
 - gazebo::physics::Joint, 547
- Joint.hh, 1356
 - MAX_JOINT_AXIS, 1357
- Joint_V
 - gazebo::physics, 122
- jointCmdSub
 - gazebo::physics::JointControllerPrivate, 573
- JointController
 - gazebo::physics::JointController, 569
- JointController.hh, 1358
- JointController_V
 - gazebo::physics, 122
- JointControllerPrivate.hh, 1359
- JointControllerPtr
 - gazebo::physics, 122
- JointPtr
 - gazebo::physics, 122
- jointPub
 - gazebo::physics::Model, 693
- JointState
 - gazebo::physics::JointState, 575, 576
- JointState.hh, 1359
- JointState_M
 - gazebo::physics, 123
- JointVisual
 - gazebo::rendering::JointVisual, 579
- JointVisual.hh, 1360
- JointVisualPrivate.hh, 1361
- JointVisualPtr
 - gazebo::rendering, 128
- JointWrench.hh, 1362
- joints
 - gazebo::physics::JointControllerPrivate, 573
- key
 - gazebo::common::KeyEvent, 584
- KeyEvent
 - gazebo::common::KeyEvent, 584
- KeyEvent.hh, 1363
- KeyFrame
 - gazebo::common::KeyFrame, 585
- KeyFrame.hh, 1364
- KeyFrame_V
 - gazebo::common::Animation, 155
- keyFrames
 - gazebo::common::Animation, 157
 - gazebo::common::NodeAnimation, 741
- L_INT16
 - gazebo::common::Image, 516
- L_INT8
 - gazebo::common::Image, 516
- LEFT
 - gazebo::common::MouseEvent, 707
- LIGHT
 - gazebo::physics::Base, 172
- LINE_MAX_LEN

- STLLoader.hh, 1479
- LINES
 - gazebo::common::SubMesh, 1076
- LINESTRIPS
 - gazebo::common::SubMesh, 1076
- LINK
 - gazebo::physics::Base, 172
- LINUX
 - SystemPaths.hh, 1486
- LO_STOP
 - gazebo::physics::Joint, 547
- Lap
 - gazebo::util::DiagnosticManager, 393
 - gazebo::util::DiagnosticTimer, 395
- laserMsg
 - gazebo::rendering::LaserVisualPrivate, 588
- laserScanSub
 - gazebo::rendering::LaserVisualPrivate, 588
- LaserVisual
 - gazebo::rendering::LaserVisual, 586
- LaserVisual.hh, 1365
- LaserVisualPrivate.hh, 1366
- LaserVisualPtr
 - gazebo::rendering, 128
- lastMeasurementTime
 - gazebo::sensors::Sensor, 917
- lastPos
 - gazebo::physics::Actor, 143
- lastRenderWallTime
 - gazebo::rendering::Camera, 223
- lastScriptTime
 - gazebo::physics::Actor, 143
- lastTraj
 - gazebo::physics::Actor, 143
- lastUpdateTime
 - gazebo::sensors::Sensor, 917
- latching
 - gazebo::transport::CallbackHelper, 195
- latitudeReference
 - gazebo::common::SphericalCoordinatesPrivate, 1063
- layoutFilename
 - gazebo::rendering::GUIOverlayPrivate, 498
- length
 - gazebo::common::Animation, 157
 - gazebo::common::NodeAnimation, 741
 - gazebo::common::SkeletonAnimation, 1034
- Light
 - gazebo::rendering::Light, 590
- Light.hh, 1367
- LightFromSDF
 - Messages, 64
- LightPtr
 - gazebo::rendering, 128
- lighting
 - gazebo::rendering::VisualPrivate, 1222
- LightingModel
 - gazebo::rendering::RTShaderSystem, 877
- limitForce
 - gazebo::physics::SimbodyJoint, 970
- lineVertices
 - gazebo::rendering::VisualPrivate, 1223
- linearAccel
 - gazebo::physics::Link, 618
- lines
 - gazebo::rendering::VisualPrivate, 1222
 - gazebo::rendering::WireBoxPrivate, 1229
- Link
 - gazebo::physics::Link, 601
- link
 - gazebo::physics::Collision, 245
- Link.hh, 1367
- Link_V
 - gazebo::physics, 123
- LinkPtr
 - gazebo::physics, 123
- LinkState
 - gazebo::physics::LinkState, 620
- LinkState.hh, 1369
- LinkState_M
 - gazebo::physics, 123
- Listen
 - gazebo::transport::Connection, 270
- Load
 - gazebo::common::BVHLoader, 191
 - gazebo::common::ColladaLoader, 235
 - gazebo::common::Image, 519
 - gazebo::common::ImageHeightmap, 525
 - gazebo::common::MeshLoader, 670
 - gazebo::common::MeshManager, 674
 - gazebo::common::STLLoader, 1073
 - gazebo::common::Video, 1189
 - gazebo::ModelPlugin, 697
 - gazebo::physics::Actor, 142
 - gazebo::physics::BallJoint, 168
 - gazebo::physics::Base, 177
 - gazebo::physics::Collision, 243
 - gazebo::physics::CollisionState, 248
 - gazebo::physics::DARTBallJoint, 306
 - gazebo::physics::DARTCollision, 312
 - gazebo::physics::DARTHinge2Joint, 321
 - gazebo::physics::DARTHingeJoint, 326
 - gazebo::physics::DARTJoint, 334
 - gazebo::physics::DARTLink, 345
 - gazebo::physics::DARTMeshShape, 349
 - gazebo::physics::DARTModel, 352
 - gazebo::physics::DARTPhysics, 358
 - gazebo::physics::DARTScrewJoint, 369

- gazebo::physics::DARTSliderJoint, 374
- gazebo::physics::DARTUniversalJoint, 381
- gazebo::physics::Entity, 411
- gazebo::physics::GearboxJoint, 463
- gazebo::physics::Gripper, 493
- gazebo::physics::HeightmapShape, 510
- gazebo::physics::Hinge2Joint, 513
- gazebo::physics::HingeJoint, 514
- gazebo::physics::Inertial, 534
- gazebo::physics::Joint, 560
- gazebo::physics::JointState, 577
- gazebo::physics::Link, 611
- gazebo::physics::LinkState, 623
- gazebo::physics::Model, 688
- gazebo::physics::ModelState, 704
- gazebo::physics::PhysicsEngine, 775
- gazebo::physics::Road, 870
- gazebo::physics::ScrewJoint, 898
- gazebo::physics::SimbodyBallJoint, 940
- gazebo::physics::SimbodyCollision, 945
- gazebo::physics::SimbodyHinge2Joint, 954
- gazebo::physics::SimbodyHingeJoint, 958
- gazebo::physics::SimbodyJoint, 966
- gazebo::physics::SimbodyLink, 978
- gazebo::physics::SimbodyMeshShape, 982
- gazebo::physics::SimbodyModel, 984
- gazebo::physics::SimbodyPhysics, 992
- gazebo::physics::SimbodyScrewJoint, 1007
- gazebo::physics::SimbodySliderJoint, 1013
- gazebo::physics::SimbodyUniversalJoint, 1019
- gazebo::physics::SliderJoint, 1045
- gazebo::physics::State, 1070
- gazebo::physics::SurfaceParams, 1091
- gazebo::physics::UniversalJoint, 1136
- gazebo::physics::World, 1248
- gazebo::physics::WorldState, 1257
- gazebo::rendering::ArrowVisual, 159
- gazebo::rendering::AxisVisual, 164
- gazebo::rendering::Camera, 215
- gazebo::rendering::CameraVisual, 233
- gazebo::rendering::COMVisual, 262
- gazebo::rendering::DepthCamera, 386
- gazebo::rendering::GpuLaser, 473
- gazebo::rendering::Heightmap, 502
- gazebo::rendering::JointVisual, 580
- gazebo::rendering::Light, 592
- gazebo::rendering::MovableText, 713
- gazebo::rendering::Projector, 811
- gazebo::rendering::RenderEngine, 858
- gazebo::rendering::Road2d, 871
- gazebo::rendering::Scene, 892
- gazebo::rendering::SelectionObj, 902
- gazebo::rendering::SonarVisual, 1052
- gazebo::rendering::TransmitterVisual, 1132
- gazebo::rendering::UserCamera, 1143
- gazebo::rendering::Visual, 1210
- gazebo::rendering::WrenchVisual, 1260
- gazebo::SensorPlugin, 925
- gazebo::sensors::CameraSensor, 230
- gazebo::sensors::ContactSensor, 291
- gazebo::sensors::DepthCameraSensor, 389
- gazebo::sensors::ForceTorqueSensor, 452
- gazebo::sensors::GaussianNoiseModel, 460
- gazebo::sensors::GpsSensor, 466
- gazebo::sensors::GpuRaySensor, 486
- gazebo::sensors::ImageGaussianNoiseModel, 522
- gazebo::sensors::ImuSensor, 528
- gazebo::sensors::MultiCameraSensor, 722
- gazebo::sensors::Noise, 750
- gazebo::sensors::RaySensor, 848
- gazebo::sensors::RFIDSensor, 860, 861
- gazebo::sensors::RFIDTag, 863
- gazebo::sensors::Sensor, 915
- gazebo::sensors::SonarSensor, 1050
- gazebo::sensors::WirelessReceiver, 1232
- gazebo::sensors::WirelessTransceiver, 1234
- gazebo::sensors::WirelessTransmitter, 1238
- gazebo::SystemPlugin, 1099
- gazebo::util::OpenAL, 757
- gazebo::util::OpenALSource, 761
- gazebo::VisualPlugin, 1218
- gazebo::WorldPlugin, 1253
- load
 - Classes for physics and dynamics, 73
 - Common, 45
 - gazebo, 104
 - Rendering, 86
 - Sensors, 90
- load_world
 - Classes for physics and dynamics, 73
- load_worlds
 - Classes for physics and dynamics, 73
- LoadFile
 - gazebo::Server, 927
- LoadFromMsg
 - gazebo::rendering::Heightmap, 502
 - gazebo::rendering::Light, 592
 - gazebo::rendering::Visual, 1210
- LoadJoints
 - gazebo::physics::Model, 688
- LoadLayout
 - gazebo::rendering::GUIOverlay, 497
- LoadPlugin
 - gazebo::physics::World, 1248
 - gazebo::rendering::Visual, 1210
- LoadPlugins
 - gazebo::physics::Model, 688
- loadProceduralPage

- gazebo::rendering::DummyPageProvider, 396
- LoadString
 - gazebo::Server, 927
- loadWorld
 - gazebo, 104
- LocalPublish
 - gazebo::transport::Publication, 816
- log
 - gazebo::common::Console, 278
- LogPlay.hh, 1370
- LogRecord.hh, 1370
 - GZ_LOG_VERSION, 1371
- LogType
 - gazebo::common::Logger, 627
- Logger
 - gazebo::common::Logger, 627
- Logplay, 631
- longitudeReference
 - gazebo::common::SphericalCoordinatesPrivate, 1063
- loop
 - gazebo::common::Animation, 157
 - gazebo::physics::Actor, 143
- Lower
 - gazebo::rendering::Heightmap, 503
- lowerLimit
 - gazebo::physics::Joint, 567
- m
 - gazebo::math::Matrix3, 653
 - gazebo::math::Matrix4, 660
- MAP_SHAPE
 - gazebo::physics::Base, 172
- MATRIX
 - gazebo::common::NodeTransform, 744
- MAX_COLLIDE_RETURNS
 - Contact.hh, 1302
- MAX_CONTACT_JOINTS
 - Contact.hh, 1302
- MAX_CONTACTS
 - gazebo::physics::DARTPhysics, 356
- MAX_JOINT_AXIS
 - Joint.hh, 1357
- MESH_SHAPE
 - gazebo::physics::Base, 173
- MIDDLE
 - gazebo::common::MouseEvent, 707
- MIN_STEP_SIZE
 - gazebo::physics::DARTPhysics, 356
- MODEL
 - gazebo::physics::Base, 172
- MODEL_PLUGIN
 - Common, 40
- MODULATE
 - gazebo::common::Material, 641
- MOVE
 - gazebo::common::MouseEvent, 708
- MSleep
 - gazebo::common::Time, 1105
- MULTIRAY_SHAPE
 - gazebo::physics::Base, 172
- mainLink
 - gazebo::physics::Actor, 144
- mainpage.html, 1371
- MakeStatic
 - gazebo::rendering::Visual, 1211
- MapShape.hh, 1371
- Master
 - gazebo::Master, 638
- Master.hh, 1372
- masterMobod
 - gazebo::physics::SimbodyLink, 980
- Material
 - gazebo::common::Material, 642
- Material.hh, 1373, 1375
- Math, 50
 - clamp, 52
 - equal, 52
 - fixnan, 52
 - isPowerOfTwo, 53
 - isnan, 53
 - max, 53
 - mean, 54
 - min, 54
 - NAN_D, 56
 - NAN_I, 56
 - parseFloat, 54
 - parseInt, 55
 - precision, 55
 - roundUpPowerOfTwo, 55
 - variance, 55
- MathTypes.hh, 1375
- Matrix3
 - gazebo::math::Matrix3, 650
- Matrix3.hh, 1376
- Matrix4
 - gazebo::math::Matrix4, 655
- Matrix4.hh, 1376
- matter
 - gazebo::physics::SimbodyPhysics, 997
- max
 - gazebo::math::Box, 185
- Math, 53
- maxScale
 - gazebo::rendering::SelectionObjPrivate, 905
- maxStepSize
 - gazebo::physics::PhysicsEngine, 780
- mean

- gazebo::sensors::GaussianNoiseModel, 460
- Math, 54
- Merge
 - gazebo::math::Box, 183
- Mesh
 - gazebo::common::Mesh, 662
- mesh
 - gazebo::physics::Actor, 144
 - gazebo::physics::MeshShape, 678
- Mesh.hh, 1377
- MeshCSG
 - gazebo::common::MeshCSG, 668
- MeshCSG.hh, 1378
 - GPtrArray, 1380
 - GtsSurface, 1380
- MeshFromSDF
 - Messages, 64
- MeshLoader
 - gazebo::common::MeshLoader, 669
- MeshLoader.hh, 1380
- MeshManager.hh, 1381
- MeshShape
 - gazebo::physics::MeshShape, 676
- MeshShape.hh, 1383
- MeshShapePtr
 - gazebo::physics, 123
- MessagePtr
 - gazebo::transport, 136
- Messages, 57
 - Convert, 59–62
 - CreateRequest, 62
 - FogFromSDF, 62
 - GUIFromSDF, 63
 - GZ_REGISTER_STATIC_MSG, 59
 - GeometryFromSDF, 63
 - GetHeader, 63
 - Init, 63
 - LightFromSDF, 64
 - MeshFromSDF, 64
 - SceneFromSDF, 64
 - Set, 64–66
 - Stamp, 66
 - TrackVisualFromSDF, 66
 - VisualFromSDF, 67
- MicToNano
 - gazebo::common::Time, 1105
- MilToNano
 - gazebo::common::Time, 1105
- min
 - gazebo::math::Box, 185
 - Math, 54
- minScale
 - gazebo::rendering::SelectionObjPrivate, 905
- mobod
 - gazebo::physics::SimbodyJoint, 971
- mode
 - gazebo::rendering::SelectionObjPrivate, 905
- Model
 - gazebo::physics::Model, 682
- model
 - gazebo::physics::Joint, 567
 - gazebo::physics::JointControllerPrivate, 573
- Model.hh, 1384
- Model_V
 - gazebo::physics, 123
- modelCache
 - gazebo::common::ModelDatabasePrivate, 695
- modelDBUpdated
 - gazebo::common::ModelDatabasePrivate, 695
- ModelDatabase.hh, 1385
 - GZ_MODEL_DB_MANIFEST_FILENAME, 1387
 - GZ_MODEL_MANIFEST_FILENAME, 1387
- ModelDatabasePrivate.hh, 1387
- modelPathsFromEnv
 - gazebo::common::SystemPaths, 1097
- ModelPlugin
 - gazebo::ModelPlugin, 697
- ModelPluginPtr
 - gazebo, 103
- ModelPtr
 - gazebo::physics, 123
- ModelState
 - gazebo::physics::ModelState, 700
- ModelState.hh, 1388
- ModelState_M
 - gazebo::physics, 123
- ModelStdDesv
 - gazebo::sensors::WirelessTransmitter, 1238
- modelTransform
 - gazebo::common::SkeletonNode, 1043
- MouseEvent
 - gazebo::common::MouseEvent, 708
- MouseEvent.hh, 1389
- MovableText
 - gazebo::rendering::MovableText, 711
- MovableText.hh, 1390
- moveScale
 - gazebo::common::MouseEvent, 708
- MoveToPosition
 - gazebo::rendering::Camera, 215
 - gazebo::rendering::UserCamera, 1143
 - gazebo::rendering::Visual, 1211
- moveToPositionQueue
 - gazebo::rendering::CameraPrivate, 226
- MoveToPositions
 - gazebo::rendering::Camera, 216
 - gazebo::rendering::Visual, 1211
- MoveToVisual

- gazebo::rendering::UserCamera, 1144
- Moved
 - gazebo::rendering::WindowManager, 1227
- MovingWindowFilter
 - Common, 41
 - gazebo::common::MovingWindowFilter, 716
- MovingWindowFilter.hh, 1391
- MovingWindowFilterPrivate
 - Common, 41
- msg
 - gazebo::common::Console, 278
- MsgFactory.hh, 1392
- MsgFactoryFn
 - gazebo::msgs, 115
- msgs.hh, 1393
- MultiCameraSensor
 - gazebo::sensors::MultiCameraSensor, 720
- MultiCameraSensor.hh, 1396
- MultiCameraSensor_V
 - gazebo::sensors, 132
- MultiCameraSensorPtr
 - gazebo::sensors, 132
- MultiRayShape
 - gazebo::physics::MultiRayShape, 726
- MultiRayShape.hh, 1396
- MultiRayShapePtr
 - gazebo::physics, 123
- mustBeBaseLink
 - gazebo::physics::SimbodyLink, 980
- mustBreakLoopHere
 - gazebo::physics::SimbodyJoint, 971
- mutex
 - gazebo::rendering::ContactVisualPrivate, 295
 - gazebo::rendering::LaserVisualPrivate, 588
 - gazebo::rendering::SonarVisualPrivate, 1054
 - gazebo::rendering::TransmitterVisualPrivate, 1134
 - gazebo::rendering::WrenchVisualPrivate, 1262
- myMaterialName
 - gazebo::rendering::VisualPrivate, 1223
- NAN_D
 - Math, 56
- NAN_I
 - Math, 56
- NEmpty
 - gazebo::sensors::WirelessTransmitter, 1238
- NO_BUTTON
 - gazebo::common::MouseEvent, 707
- NO_EVENT
 - gazebo::common::KeyEvent, 584
 - gazebo::common::MouseEvent, 708
- NODE
 - gazebo::common::SkeletonNode, 1037
- NONE
 - gazebo::rendering::RenderEngine, 856
 - gazebo::sensors::Noise, 749
- NObstacle
 - gazebo::sensors::WirelessTransmitter, 1239
- NRealGen
 - gazebo::math, 113
- NSleep
 - gazebo::common::Time, 1106
- NULL
 - CommonTypes.hh, 1295
- name
 - gazebo::common::Animation, 157
 - gazebo::common::Material, 647
 - gazebo::common::NodeAnimation, 741
 - gazebo::common::SkeletonAnimation, 1035
 - gazebo::common::SkeletonNode, 1043
 - gazebo::physics::State, 1072
 - gazebo::rendering::Camera, 223
 - gazebo::rendering::VisualPrivate, 1223
- near
 - gazebo::rendering::GpuLaser, 476
- NeedsUpdate
 - gazebo::sensors::Sensor, 916
- NewContact
 - gazebo::physics::ContactManager, 285
- newData
 - gazebo::rendering::Camera, 223
- newImageFrame
 - gazebo::rendering::Camera, 223
- newLaserScans
 - gazebo::physics::MultiRayShape, 730
- NewMsg
 - gazebo::msgs::MsgFactory, 718
- NewNoiseModel
 - gazebo::sensors::NoiseFactory, 751
- NewPhysicsEngine
 - gazebo::physics::PhysicsFactory, 782
- NewSensor
 - gazebo::sensors::SensorFactory, 919
- Node
 - gazebo::transport::Node, 733
- node
 - gazebo::physics::Entity, 415
 - gazebo::physics::Gripper, 493
 - gazebo::physics::JointControllerPrivate, 573
 - gazebo::physics::PhysicsEngine, 780
 - gazebo::rendering::CameraPrivate, 226
 - gazebo::rendering::ContactVisualPrivate, 295
 - gazebo::rendering::LaserVisualPrivate, 588
 - gazebo::rendering::RFIDTagVisualPrivate, 866
 - gazebo::rendering::RFIDVisualPrivate, 868
 - gazebo::rendering::SonarVisualPrivate, 1054
 - gazebo::rendering::TransmitterVisualPrivate, 1134
 - gazebo::rendering::WrenchVisualPrivate, 1263

- gazebo::sensors::Sensor, 917
- Node.hh, 1397
- NodeAnimation
 - gazebo::common::NodeAnimation, 739
- NodeAssignment
 - gazebo::common::NodeAssignment, 742
- nodeIndex
 - gazebo::common::NodeAssignment, 742
- NodeMap
 - gazebo::common, 110
- NodeMapIter
 - gazebo::common, 110
- NodePtr
 - gazebo::transport, 136
- NodeTransform
 - gazebo::common::NodeTransform, 744
- nodes
 - gazebo::common::Skeleton, 1030
- Noise
 - gazebo::sensors::Noise, 749
- Noise.hh, 1398
- NoisePtr
 - gazebo::sensors, 132
- NoiseType
 - gazebo::sensors::Noise, 749
- noises
 - gazebo::sensors::Sensor, 918
- normal
 - gazebo::math::Plane, 790
 - gazebo::rendering::ContactVisualPrivate::Contact-Point, 286
- NormalRealDist
 - gazebo::math, 113
- Normalize
 - gazebo::math::Angle, 148
 - gazebo::math::Quaternion, 831
 - gazebo::math::Vector2d, 1150
 - gazebo::math::Vector2i, 1159
 - gazebo::math::Vector3, 1171
 - gazebo::math::Vector4, 1181
- normals
 - gazebo::physics::Contact, 282
- Notify
 - gazebo::util::LogRecord, 636
- notifyRenderSingleObject
 - gazebo::rendering::GpuLaser, 473
- nsec
 - gazebo::common::Time, 1120
- NumericAnimation
 - gazebo::common::NumericAnimation, 753
- NumericAnimationPtr
 - gazebo::common, 110
- NumericKeyFrame
 - gazebo::common::NumericKeyFrame, 754
- ORDER_MAX
 - STLLoader.hh, 1479
- OTHER
 - gazebo::sensors, 133
- offset
 - gazebo::physics::MultiRayShape, 730
- Ogre, 137
- ogre, 137
- ogre_gazebo.h, 1399
- ogrePathsFromEnv
 - gazebo::common::SystemPaths, 1097
- oldAction
 - gazebo::physics::Actor, 144
- onAnimationComplete
 - gazebo::rendering::Camera, 223
 - gazebo::rendering::VisualPrivate, 1223
- OnPhysicsMsg
 - gazebo::physics::DARTPhysics, 359
 - gazebo::physics::PhysicsEngine, 775
 - gazebo::physics::SimbodyPhysics, 992
- OnPoseChange
 - gazebo::physics::DARTCollision, 312
 - gazebo::physics::DARTLink, 345
 - gazebo::physics::Entity, 412
 - gazebo::physics::Link, 612
 - gazebo::physics::Model, 688
 - gazebo::physics::SimbodyCollision, 945
 - gazebo::physics::SimbodyLink, 978
 - gazebo::rendering::Light, 593
- OnRequest
 - gazebo::physics::DARTPhysics, 359
 - gazebo::physics::PhysicsEngine, 776
 - gazebo::physics::SimbodyPhysics, 993
- One
 - gazebo::math::Vector3, 1177
- Open
 - gazebo::util::LogPlay, 630
- OpenAL.hh, 1401
- OpenALSink
 - gazebo::util::OpenALSink, 758
- OpenALSinkPtr
 - gazebo::util, 137
- OpenALSource
 - gazebo::util::OpenALSource, 759
- OpenALSourcePtr
 - gazebo::util, 137
- operator<
 - gazebo::common::Time, 1113, 1114
 - gazebo::math::Angle, 150
- operator<<
 - gazebo::common::Color, 259
 - gazebo::common::Exception, 446
 - gazebo::common::Material, 647
 - gazebo::common::Time, 1120

- gazebo::common::Timer, 1122
- gazebo::math::Angle, 152
- gazebo::math::Box, 185
- gazebo::math::Matrix3, 653
- gazebo::math::Matrix4, 660
- gazebo::math::Pose, 804
- gazebo::math::Quaternion, 837
- gazebo::math::Vector2d, 1155
- gazebo::math::Vector2i, 1164
- gazebo::math::Vector3, 1177
- gazebo::math::Vector4, 1186
- gazebo::physics::CollisionState, 249
- gazebo::physics::Inertial, 538
- gazebo::physics::JointState, 578
- gazebo::physics::LinkState, 625
- gazebo::physics::ModelState, 706
- gazebo::physics::WorldState, 1259
- operator<=
 - gazebo::common::Time, 1114, 1115
 - gazebo::math::Angle, 150
- operator>
 - gazebo::common::Time, 1117
 - gazebo::math::Angle, 151
- operator>>
 - gazebo::common::Color, 259
 - gazebo::common::Time, 1120
 - gazebo::math::Angle, 152
 - gazebo::math::Pose, 805
 - gazebo::math::Quaternion, 837
 - gazebo::math::Vector2d, 1155
 - gazebo::math::Vector2i, 1164
 - gazebo::math::Vector3, 1177
 - gazebo::math::Vector4, 1187
- operator>=
 - gazebo::common::Time, 1118
 - gazebo::math::Angle, 151
- operator*
 - gazebo::common::Color, 254
 - gazebo::common::NodeTransform, 745, 746
 - gazebo::common::Time, 1107
 - gazebo::math::Angle, 148
 - gazebo::math::Matrix3, 650–652
 - gazebo::math::Matrix4, 657
 - gazebo::math::Pose, 801
 - gazebo::math::Quaternion, 832
 - gazebo::math::Vector2d, 1151
 - gazebo::math::Vector2i, 1159
 - gazebo::math::Vector3, 1172, 1176
 - gazebo::math::Vector4, 1182
- operator*=
 - gazebo::common::Color, 254
 - gazebo::common::Time, 1108
 - gazebo::math::Angle, 148
 - gazebo::math::Quaternion, 832
 - gazebo::math::Vector2d, 1151
 - gazebo::math::Vector2i, 1160
 - gazebo::math::Vector3, 1172
 - gazebo::math::Vector4, 1183
- operator()
 - gazebo::common::FileLogger, 448
 - gazebo::common::Logger, 627, 628
 - gazebo::common::NodeTransform, 745
 - gazebo::event::EventT, 437–439
- operator+
 - gazebo::common::Color, 255
 - gazebo::common::Time, 1108, 1109
 - gazebo::math::Angle, 149
 - gazebo::math::Box, 183
 - gazebo::math::Matrix3, 651
 - gazebo::math::Pose, 802
 - gazebo::math::Quaternion, 833
 - gazebo::math::Vector2d, 1152
 - gazebo::math::Vector2i, 1160
 - gazebo::math::Vector3, 1173
 - gazebo::math::Vector4, 1183
 - gazebo::physics::CollisionState, 248
 - gazebo::physics::Inertial, 534
 - gazebo::physics::JointState, 577
 - gazebo::physics::JointWrench, 582
 - gazebo::physics::LinkState, 623
 - gazebo::physics::ModelState, 704
 - gazebo::physics::WorldState, 1257
- operator+=
 - gazebo::common::Color, 255
 - gazebo::common::Time, 1109, 1110
 - gazebo::math::Angle, 149
 - gazebo::math::Box, 184
 - gazebo::math::Pose, 802
 - gazebo::math::Quaternion, 833
 - gazebo::math::Vector2d, 1152
 - gazebo::math::Vector2i, 1160
 - gazebo::math::Vector3, 1173
 - gazebo::math::Vector4, 1183
 - gazebo::physics::Inertial, 535
- operator-
 - gazebo::common::Color, 255, 256
 - gazebo::common::Time, 1110
 - gazebo::math::Angle, 149
 - gazebo::math::Box, 184
 - gazebo::math::Matrix3, 651
 - gazebo::math::Pose, 802
 - gazebo::math::Quaternion, 833
 - gazebo::math::Vector2d, 1152
 - gazebo::math::Vector2i, 1161
 - gazebo::math::Vector3, 1173
 - gazebo::math::Vector4, 1184
 - gazebo::physics::CollisionState, 248
 - gazebo::physics::JointState, 578

- gazebo::physics::JointWrench, 582
- gazebo::physics::LinkState, 623
- gazebo::physics::ModelState, 705
- gazebo::physics::State, 1070
- gazebo::physics::WorldState, 1257
- operator-=
 - gazebo::common::Color, 256
 - gazebo::common::Time, 1111
 - gazebo::math::Angle, 149
 - gazebo::math::Pose, 803
 - gazebo::math::Quaternion, 834
 - gazebo::math::Vector2d, 1152
 - gazebo::math::Vector2i, 1161
 - gazebo::math::Vector3, 1173
 - gazebo::math::Vector4, 1184
- operator/
 - gazebo::common::Color, 256
 - gazebo::common::Time, 1111, 1112
 - gazebo::math::Angle, 150
 - gazebo::math::Vector2d, 1153
 - gazebo::math::Vector2i, 1161, 1162
 - gazebo::math::Vector3, 1174
 - gazebo::math::Vector4, 1184
- operator/=
 - gazebo::common::Color, 257
 - gazebo::common::Time, 1112, 1113
 - gazebo::math::Angle, 150
 - gazebo::math::Vector2d, 1153
 - gazebo::math::Vector2i, 1162
 - gazebo::math::Vector3, 1174
 - gazebo::math::Vector4, 1185
- operator=
 - gazebo::common::Color, 257
 - gazebo::common::PID, 786
 - gazebo::common::Time, 1115
 - gazebo::math::Box, 184
 - gazebo::math::Matrix4, 657, 658
 - gazebo::math::Plane, 790
 - gazebo::math::Pose, 803
 - gazebo::math::Quaternion, 834
 - gazebo::math::Vector2d, 1154
 - gazebo::math::Vector2i, 1163
 - gazebo::math::Vector3, 1175
 - gazebo::math::Vector4, 1185, 1186
 - gazebo::physics::CollisionState, 249
 - gazebo::physics::Contact, 281
 - gazebo::physics::Inertial, 535
 - gazebo::physics::JointState, 578
 - gazebo::physics::JointWrench, 582
 - gazebo::physics::LinkState, 624
 - gazebo::physics::ModelState, 705
 - gazebo::physics::State, 1071
 - gazebo::physics::WorldState, 1258
- operator==
 - gazebo::common::Color, 257
 - gazebo::common::Time, 1116
 - gazebo::math::Angle, 151
 - gazebo::math::Box, 184
 - gazebo::math::Matrix3, 651
 - gazebo::math::Matrix4, 658
 - gazebo::math::Pose, 803
 - gazebo::math::Quaternion, 834
 - gazebo::math::Vector2d, 1154
 - gazebo::math::Vector2i, 1163
 - gazebo::math::Vector3, 1175
 - gazebo::math::Vector4, 1186
 - gazebo::physics::Base, 178
- operator[]
 - gazebo::common::Color, 257
 - gazebo::math::Matrix3, 651
 - gazebo::math::Matrix4, 658
 - gazebo::math::Vector2d, 1154
 - gazebo::math::Vector2i, 1163
 - gazebo::math::Vector3, 1175
 - gazebo::math::Vector4, 1186
- OrbitViewController
 - gazebo::rendering::OrbitViewController, 764
- orbitViewController
 - gazebo::rendering::UserCameraPrivate, 1147
- OrbitViewController.hh, 1401
- origMaterialName
 - gazebo::rendering::VisualPrivate, 1223
- PHONG
 - gazebo::common::Material, 642
- PID
 - gazebo::common::PID, 784
- PID.hh, 1409
- PIXEL_FORMAT_COUNT
 - gazebo::common::Image, 516
- PLANE_SHAPE
 - gazebo::physics::Base, 173
- POINTS
 - gazebo::common::SubMesh, 1076
- PRESS
 - gazebo::common::KeyEvent, 584
 - gazebo::common::MouseEvent, 708
- Param_V
 - gazebo::common, 110
- parent
 - gazebo::common::SkeletonNode, 1044
 - gazebo::physics::Base, 180
 - gazebo::rendering::VisualPrivate, 1223
 - gazebo::rendering::WireBoxPrivate, 1229
- parentAnchorPose
 - gazebo::physics::Joint, 567
- parentEntity
 - gazebo::physics::Entity, 415

- gazebo::sensors::WirelessTransceiver, 1235
- parentId
 - gazebo::sensors::Sensor, 918
- parentLink
 - gazebo::physics::Joint, 567
- parentName
 - gazebo::sensors::Sensor, 918
- ParseArgs
 - gazebo::Server, 927
- parseFloat
 - Math, 54
- parseInt
 - Math, 55
- pathLength
 - gazebo::physics::Actor, 144
- Pause
 - gazebo::util::OpenALSource, 761
- pause
 - gazebo::event::Events, 431
- pause_incoming
 - Transport, 96
- pause_world
 - Classes for physics and dynamics, 73
- pause_worlds
 - Classes for physics and dynamics, 74
- PauseIncoming
 - gazebo::transport::TopicManager, 1127
- PhysicsEngine
 - gazebo::physics::PhysicsEngine, 770
- PhysicsEngine.hh, 1402
- PhysicsEnginePtr
 - gazebo::physics, 123
- PhysicsFactory.hh, 1403
- PhysicsFactoryFn
 - Classes for physics and dynamics, 72
- PhysicsIface.hh, 1404
- physicsInitialized
 - gazebo::physics::SimbodyJoint, 971
 - gazebo::physics::SimbodyLink, 980
- physicsSub
 - gazebo::physics::PhysicsEngine, 780
- PhysicsTypes.hh, 1406
 - GZ_ALL_COLLIDE, 1409
 - GZ_FIXED_COLLIDE, 1409
 - GZ_GHOST_COLLIDE, 1409
 - GZ_NONE_COLLIDE, 1409
 - GZ_SENSOR_COLLIDE, 1409
- physicsUpdateMutex
 - gazebo::physics::PhysicsEngine, 780
- Pi
 - gazebo::math::Angle, 153
- PixelFormat
 - gazebo::common::Image, 516
- PixelFormatNames
 - Common, 46
- PlaceOnEntity
 - gazebo::physics::Entity, 412
- PlaceOnNearestEntityBelow
 - gazebo::physics::Entity, 412
- placeable
 - gazebo::physics::Collision, 245
- Plane
 - gazebo::math::Plane, 789
- Plane.hh, 1410
- PlaneShape
 - gazebo::physics::PlaneShape, 792
- PlaneShape.hh, 1411
- Play
 - gazebo::physics::Actor, 142
 - gazebo::util::OpenALSource, 761
- playStartTime
 - gazebo::physics::Actor, 144
- Plugin.hh, 1412
 - GZ_REGISTER_MODEL_PLUGIN, 1414
 - GZ_REGISTER_SENSOR_PLUGIN, 1414
 - GZ_REGISTER_SYSTEM_PLUGIN, 1415
 - GZ_REGISTER_VISUAL_PLUGIN, 1415
 - GZ_REGISTER_WORLD_PLUGIN, 1415
- pluginPathsFromEnv
 - gazebo::common::SystemPaths, 1097
- PluginT
 - gazebo::PluginT, 795
- PluginType
 - Common, 40
- plugins
 - gazebo::rendering::VisualPrivate, 1223
 - gazebo::sensors::Sensor, 918
- pointSize
 - gazebo::common::Material, 647
- points
 - gazebo::math::RotationSpline, 875
 - gazebo::math::Spline, 1067
 - gazebo::rendering::ContactVisualPrivate, 295
 - gazebo::rendering::TransmitterVisualPrivate, 1134
- pos
 - gazebo::common::MouseEvent, 708
 - gazebo::math::Pose, 805
- posPids
 - gazebo::physics::JointControllerPrivate, 573
- Pose
 - gazebo::math::Pose, 799
- pose
 - gazebo::sensors::Sensor, 918
- Pose.hh, 1416
- Pose2Transform
 - gazebo::physics::SimbodyPhysics, 993
- PoseAnimation
 - gazebo::common::PoseAnimation, 806

- PoseAnimationPtr
 - gazebo::common, 110
- PoseKeyFrame
 - gazebo::common::PoseKeyFrame, 809
- poseSub
 - gazebo::sensors::Sensor, 918
- positions
 - gazebo::physics::Contact, 282
 - gazebo::physics::JointControllerPrivate, 573
- PostRender
 - gazebo::rendering::Camera, 216
 - gazebo::rendering::DepthCamera, 386
 - gazebo::rendering::GpuLaser, 473
 - gazebo::rendering::UserCamera, 1144
- postRender
 - gazebo::event::Events, 431
- power
 - gazebo::sensors::WirelessTransceiver, 1235
- PreLoad
 - gazebo::Server, 928
- PreRender
 - gazebo::rendering::Scene, 892
- preRender
 - gazebo::event::Events, 431
- preRenderConnection
 - gazebo::rendering::VisualPrivate, 1223
- precision
 - gazebo::sensors::GaussianNoiseModel, 460
 - Math, 55
- PrepareHardwareBuffers
 - gazebo::rendering::DynamicRenderable, 403
- prepareProceduralPage
 - gazebo::rendering::DummyPageProvider, 396
- pressPos
 - gazebo::common::MouseEvent, 708
- prevAnimTime
 - gazebo::rendering::Camera, 223
 - gazebo::rendering::VisualPrivate, 1223
- prevAnimationTime
 - gazebo::physics::Entity, 415
- prevFrameTime
 - gazebo::physics::Actor, 144
- prevPos
 - gazebo::common::MouseEvent, 709
- prevUpdateTime
 - gazebo::physics::JointControllerPrivate, 573
- PrimitiveType
 - gazebo::common::SubMesh, 1076
- Print
 - gazebo::common::Exception, 446
 - gazebo::physics::Base, 178
- print_version
 - gazebo, 105
- PrintEntityTree
 - gazebo::physics::World, 1248
- PrintSceneGraph
 - gazebo::rendering::Scene, 892
- PrintSource
 - gazebo::common::NodeTransform, 746
- PrintTransforms
 - gazebo::common::Skeleton, 1029
- PrintUsage
 - gazebo::Server, 928
- printVersion
 - gazebo, 105
- ProcessIncoming
 - gazebo::transport::Node, 736
- ProcessMsg
 - gazebo::physics::BoxShape, 187
 - gazebo::physics::Collision, 243
 - gazebo::physics::CylinderShape, 300
 - gazebo::physics::HeightmapShape, 511
 - gazebo::physics::Inertial, 535
 - gazebo::physics::Link, 612
 - gazebo::physics::MeshShape, 677
 - gazebo::physics::Model, 689
 - gazebo::physics::MultiRayShape, 730
 - gazebo::physics::PlaneShape, 793
 - gazebo::physics::RayShape, 853
 - gazebo::physics::Shape, 935
 - gazebo::physics::SphereShape, 1056
 - gazebo::physics::SurfaceParams, 1091
- ProcessNodes
 - gazebo::transport::TopicManager, 1127
- ProcessPublishers
 - gazebo::transport::Node, 736
- ProcessWriteQueue
 - gazebo::transport::Connection, 270
- Projector
 - gazebo::rendering::Projector, 811
- Projector.hh, 1417
- provideFeedback
 - gazebo::physics::Joint, 567
- pub
 - gazebo::sensors::WirelessTransceiver, 1235
- Publication
 - gazebo::transport::Publication, 814
- Publication.hh, 1417
- PublicationPtr
 - gazebo::transport, 136
- PublicationTransport
 - gazebo::transport::PublicationTransport, 818
- PublicationTransport.hh, 1419
- PublicationTransportPtr
 - gazebo::transport, 136
- Publish
 - gazebo::transport::Node, 736
 - gazebo::transport::Publication, 816

- gazebo::transport::Publisher, 822
- gazebo::transport::TopicManager, 1127
- publish
 - Transport, 96
- PublishContacts
 - gazebo::physics::ContactManager, 285
- PublishModelPose
 - gazebo::physics::World, 1248
- PublishTask
 - gazebo::transport::PublishTask, 824
- Publisher
 - gazebo::transport::Publisher, 821
- publisher
 - gazebo::physics::ContactPublisher, 287
- Publisher.hh, 1420
- PublisherPtr
 - gazebo::transport, 136
- Purple
 - gazebo::common::Color, 260
- QuadToQuad
 - gazebo::physics::SimbodyPhysics, 993
- quantized
 - gazebo::sensors::GaussianNoiseModel, 461
- Quaternion
 - gazebo::math::Quaternion, 827, 828
- Quaternion.hh, 1421
- r
 - gazebo::common::Color, 260
- R_FLOAT16
 - gazebo::common::Image, 516
- R_FLOAT32
 - gazebo::common::Image, 516
- RAY
 - gazebo::sensors, 133
- RAY_SHAPE
 - gazebo::physics::Base, 172
- RELEASE
 - gazebo::common::KeyEvent, 584
 - gazebo::common::MouseEvent, 708
- RENDER_PATH_COUNT
 - gazebo::rendering::RenderEngine, 856
- RENDERING_LINE_LIST
 - gazebo::rendering, 128
- RENDERING_LINE_STRIP
 - gazebo::rendering, 128
- RENDERING_MESH_RESOURCE
 - gazebo::rendering, 128
- RENDERING_POINT_LIST
 - gazebo::rendering, 128
- RENDERING_TRIANGLE_FAN
 - gazebo::rendering, 128
- RENDERING_TRIANGLE_LIST
 - gazebo::rendering, 128
- RENDERING_TRIANGLE_STRIP
 - gazebo::rendering, 128
- REPLACE
 - gazebo::common::Material, 641
- RFIDSensor
 - gazebo::sensors::RFIDSensor, 860
- RFIDSensor.hh, 1430
- RFIDSensor_V
 - gazebo::sensors, 133
- RFIDSensorPtr
 - gazebo::sensors, 133
- RFIDTag
 - gazebo::sensors::RFIDTag, 863
- RFIDTag.hh, 1431
- RFIDTag_V
 - gazebo::sensors, 133
- RFIDTagPtr
 - gazebo::sensors, 133
- RFIDTagVisual
 - gazebo::rendering::RFIDTagVisual, 865
- RFIDTagVisual.hh, 1432
- RFIDTagVisualPrivate.hh, 1432
- RFIDTagVisualPtr
 - gazebo::rendering, 128
- RFIDVisual
 - gazebo::rendering::RFIDVisual, 867
- RFIDVisual.hh, 1433
- RFIDVisualPrivate.hh, 1433
- RFIDVisualPtr
 - gazebo::rendering, 128
- RGB_FLOAT16
 - gazebo::common::Image, 516
- RGB_FLOAT32
 - gazebo::common::Image, 516
- RGB_INT16
 - gazebo::common::Image, 516
- RGB_INT32
 - gazebo::common::Image, 516
- RGB_INT8
 - gazebo::common::Image, 516
- RGBA
 - gazebo::common::Color, 252
- RGBA_INT8
 - gazebo::common::Image, 516
- RIGHT
 - gazebo::common::MouseEvent, 707
- ROT
 - gazebo::rendering::SelectionObj, 901
- ROT_X
 - gazebo::rendering::SelectionObj, 901
- ROT_Y
 - gazebo::rendering::SelectionObj, 901
- ROT_Z
 - gazebo::rendering::SelectionObj, 901

- ROTATE
 - gazebo::common::NodeTransform, 744
- RTShaderSystem.hh, 1437
- Radian
 - gazebo::math::Angle, 151
- Raise
 - gazebo::rendering::Heightmap, 503
- Rand.hh, 1422
- rangeCountRatio
 - gazebo::sensors::GpuRaySensor, 487
- rangeElem
 - gazebo::physics::MultiRayShape, 731
 - gazebo::sensors::GpuRaySensor, 488
- RawCallbackHelper
 - gazebo::transport::RawCallbackHelper, 841
- rawNW
 - gazebo::common::Skeleton, 1030
- RawNodeAnim
 - gazebo::common, 110
- RawNodeWeights
 - gazebo::common, 110
- RawSkeletonAnim
 - gazebo::common, 110
- rawTransforms
 - gazebo::common::SkeletonNode, 1044
- rayCountRatio
 - gazebo::rendering::GpuLaser, 476
- rayElem
 - gazebo::physics::MultiRayShape, 731
- rayFans
 - gazebo::rendering::LaserVisualPrivate, 588
- RaySensor
 - gazebo::sensors::RaySensor, 844
- RaySensor.hh, 1424
- RaySensor_V
 - gazebo::sensors, 132
- RaySensorPtr
 - gazebo::sensors, 132
- RayShape
 - gazebo::physics::RayShape, 851
- RayShape.hh, 1425
- RayShapePtr
 - gazebo::physics, 123
- rays
 - gazebo::physics::MultiRayShape, 731
- Read
 - gazebo::transport::Connection, 270
- ReadCallback
 - gazebo::transport::Connection, 266
- ReadPixelBuffer
 - gazebo::rendering::Camera, 216
- realTime
 - gazebo::common::UpdateInfo, 1137
 - gazebo::physics::State, 1072
- realTimeUpdateRate
 - gazebo::physics::PhysicsEngine, 780
- RecalcTangents
 - gazebo::math::RotationSpline, 874
 - gazebo::math::Spline, 1066
- RecalculateMatrix
 - gazebo::common::NodeTransform, 746
- RecalculateNormals
 - gazebo::common::Mesh, 666
 - gazebo::common::SubMesh, 1082
- receiveMutex
 - gazebo::rendering::CameraPrivate, 227
- receivedMsg
 - gazebo::rendering::ContactVisualPrivate, 295
 - gazebo::rendering::LaserVisualPrivate, 588
 - gazebo::rendering::SonarVisualPrivate, 1054
 - gazebo::rendering::TransmitterVisualPrivate, 1134
 - gazebo::rendering::WrenchVisualPrivate, 1263
- Red
 - gazebo::common::Color, 260
- referenceBody
 - gazebo::physics::GearboxJoint, 464
- referencePose
 - gazebo::sensors::WirelessTransceiver, 1235
- RegisterAll
 - gazebo::physics::PhysicsFactory, 782
 - gazebo::sensors::SensorFactory, 920
- RegisterMsg
 - gazebo::msgs::MsgFactory, 718
- RegisterPhysicsEngine
 - gazebo::physics::PhysicsFactory, 782
- RegisterSensor
 - gazebo::sensors::SensorFactory, 920
- RegisterTopicNamespace
 - gazebo::transport::ConnectionManager, 274
 - gazebo::transport::TopicManager, 1128
- relativeEndPos
 - gazebo::physics::RayShape, 854
- relativeStartPos
 - gazebo::physics::RayShape, 854
- Remove
 - gazebo::util::LogRecord, 636
- remove_scene
 - Rendering, 86
- remove_sensor
 - Sensors, 90
- remove_sensors
 - Sensors, 91
- remove_worlds
 - Classes for physics and dynamics, 74
- RemoveCallback
 - gazebo::transport::Node, 736
- RemoveCamera
 - gazebo::rendering::Scene, 893

- RemoveChild
 - gazebo::physics::Base, 178
 - gazebo::physics::Link, 612
 - gazebo::physics::Model, 689
- RemoveChildJoint
 - gazebo::physics::Link, 612
- RemoveChildren
 - gazebo::physics::Base, 178
- RemoveCollision
 - gazebo::physics::Link, 612
- RemoveConnection
 - gazebo::transport::ConnectionManager, 274
- RemoveFilter
 - gazebo::physics::ContactManager, 285
- RemoveNode
 - gazebo::transport::TopicManager, 1128
- RemoveParentJoint
 - gazebo::physics::Link, 612
- RemovePlugin
 - gazebo::physics::World, 1249
 - gazebo::rendering::Visual, 1211
- RemoveProjectors
 - gazebo::rendering::Scene, 893
- RemovePublisher
 - gazebo::transport::Publication, 816
- RemoveScene
 - gazebo::rendering::RenderEngine, 858
 - gazebo::rendering::RTShaderSystem, 878
- removeScene
 - gazebo::rendering::Events, 434
- RemoveSensor
 - gazebo::sensors::SensorManager, 923
- RemoveSensors
 - gazebo::sensors::SensorManager, 923
- RemoveShadows
 - gazebo::rendering::RTShaderSystem, 879
- RemoveSubscription
 - gazebo::transport::Publication, 817
- RemoveTransport
 - gazebo::transport::Publication, 817
- RemoveVisual
 - gazebo::rendering::Scene, 893
- Render
 - gazebo::rendering::Camera, 216
- render
 - gazebo::event::Events, 432
- RenderEngine.hh, 1426
- RenderEvents.hh, 1426
- RenderImpl
 - gazebo::rendering::Camera, 217
- RenderOpType
 - gazebo::rendering, 128
- RenderPathType
 - gazebo::rendering::RenderEngine, 856
- renderPeriod
 - gazebo::rendering::CameraPrivate, 227
- renderTarget
 - gazebo::rendering::Camera, 223
- renderTexture
 - gazebo::rendering::Camera, 223
- RenderTypes.hh, 1428
 - GZ_VISIBILITY_ALL, 1430
 - GZ_VISIBILITY_GUI, 1430
 - GZ_VISIBILITY_SELECTABLE, 1430
 - GZ_VISIBILITY_SELECTION, 1430
- Rendering, 83
 - create_scene, 85
 - fini, 85
 - get_scene, 85
 - init, 85
 - load, 86
 - remove_scene, 86
- RenderingIface.hh, 1427
- request
 - Transport, 97
- requestNoReply
 - Transport, 97
- requestPub
 - gazebo::physics::Entity, 415
- requestSub
 - gazebo::physics::PhysicsEngine, 780
- requests
 - gazebo::rendering::Camera, 223
- Rescale
 - gazebo::common::Image, 519
- Reset
 - gazebo::common::Color, 258
 - gazebo::common::PID, 786
 - gazebo::common::SkeletonNode, 1041
 - gazebo::math::Pose, 803
 - gazebo::ModelPlugin, 697
 - gazebo::physics::Base, 178, 179
 - gazebo::physics::Contact, 281
 - gazebo::physics::DARTJoint, 335
 - gazebo::physics::DARTPhysics, 359
 - gazebo::physics::Entity, 412
 - gazebo::physics::Inertial, 535
 - gazebo::physics::Joint, 560
 - gazebo::physics::JointController, 571
 - gazebo::physics::Link, 613
 - gazebo::physics::Model, 689
 - gazebo::physics::PhysicsEngine, 776
 - gazebo::physics::SimbodyJoint, 967
 - gazebo::physics::SimbodyPhysics, 994
 - gazebo::physics::World, 1249
 - gazebo::SensorPlugin, 925
 - gazebo::SystemPlugin, 1099
 - gazebo::VisualPlugin, 1219

- gazebo::WorldPlugin, 1253
- ResetCount
 - gazebo::physics::ContactManager, 286
- ResetEntities
 - gazebo::physics::World, 1249
- ResetLastUpdateTime
 - gazebo::sensors::Sensor, 916
- ResetLastUpdateTimes
 - gazebo::sensors::SensorManager, 923
- ResetPhysicsStates
 - gazebo::physics::Link, 613
- ResetTime
 - gazebo::physics::World, 1249
- Resize
 - gazebo::rendering::GUIOverlay, 497
 - gazebo::rendering::UserCamera, 1144
 - gazebo::rendering::WindowManager, 1227
- responsePub
 - gazebo::physics::PhysicsEngine, 781
- RestoreSimbodyState
 - gazebo::physics::SimbodyHingeJoint, 959
 - gazebo::physics::SimbodyJoint, 967
 - gazebo::physics::SimbodyLink, 978
- RestoreState
 - gazebo::physics::DARTModel, 352
- Rewind
 - gazebo::util::OpenALSource, 761
- rfidSub
 - gazebo::rendering::RFIDTagVisualPrivate, 866
 - gazebo::rendering::RFIDVisualPrivate, 869
- ribbonTrail
 - gazebo::rendering::VisualPrivate, 1223
- Road, 870
 - gazebo::physics::Road, 870
- Road.hh, 1434
- Road2d
 - gazebo::rendering::Road2d, 871
- Road2d.hh, 1435
- RoadPtr
 - gazebo::physics, 123
- root
 - gazebo::common::Skeleton, 1030
 - gazebo::rendering::RenderEngine, 859
- rot
 - gazebo::math::Pose, 805
- rotVisual
 - gazebo::rendering::SelectionObjPrivate, 905
- rotXVisual
 - gazebo::rendering::SelectionObjPrivate, 905
- rotYVisual
 - gazebo::rendering::SelectionObjPrivate, 905
- rotZVisual
 - gazebo::rendering::SelectionObjPrivate, 906
- Rotate
 - gazebo::physics::Inertial, 535
- rotate
 - gazebo::common::PoseKeyFrame, 810
- RotatePitch
 - gazebo::rendering::Camera, 217
- RotatePositionAboutOrigin
 - gazebo::math::Pose, 803
- RotateVector
 - gazebo::math::Quaternion, 834
- RotateVectorReverse
 - gazebo::math::Quaternion, 834
- RotateYaw
 - gazebo::rendering::Camera, 217
- rotationNode
 - gazebo::rendering::ArrowVisualPrivate, 160
- RotationSpline
 - gazebo::math::RotationSpline, 872
- RotationSpline.hh, 1436
- Round
 - gazebo::math::Pose, 804
 - gazebo::math::Quaternion, 835
 - gazebo::math::Vector3, 1176
- roundUpPowerOfTwo
 - Math, 55
- rttImageSetCount
 - gazebo::rendering::GUIOverlayPrivate, 498
- Run
 - gazebo::Master, 638
 - gazebo::physics::World, 1249
 - gazebo::Server, 928
 - gazebo::transport::ConnectionManager, 274
- run
 - gazebo, 105
 - Transport, 98
- run_once
 - Sensors, 91
- run_threads
 - Sensors, 91
- run_world
 - Classes for physics and dynamics, 74
- run_worlds
 - Classes for physics and dynamics, 74
- RunBlocking
 - gazebo::physics::World, 1249
- RunOnce
 - gazebo::Master, 638
- RunThread
 - gazebo::Master, 638
- RunThreads
 - gazebo::sensors::SensorManager, 923
- runWorld
 - gazebo, 105
- SCALE

- gazebo::common::NodeTransform, 744
- gazebo::rendering::SelectionObj, 901
- SCALE_X
 - gazebo::rendering::SelectionObj, 901
- SCALE_Y
 - gazebo::rendering::SelectionObj, 901
- SCALE_Z
 - gazebo::rendering::SelectionObj, 901
- SCREW_JOINT
 - gazebo::physics::Base, 172
- SCROLL
 - gazebo::common::MouseEvent, 708
- SELECTION_NONE
 - gazebo::rendering::SelectionObj, 901
- SENSOR_COLLISION
 - gazebo::physics::Base, 173
- SENSOR_PLUGIN
 - Common, 41
- SHADE_COUNT
 - gazebo::common::Material, 642
- SHAPE
 - gazebo::physics::Base, 172
- SLIDER_JOINT
 - gazebo::physics::Base, 172
- SM2Profile
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 1046
- SPHERE_SHAPE
 - gazebo::physics::Base, 173
- SSLM_NormalMapLightingObjectSpace
 - gazebo::rendering::RTShaderSystem, 877
- SSLM_NormalMapLightingTangentSpace
 - gazebo::rendering::RTShaderSystem, 877
- SSLM_PerPixelLighting
 - gazebo::rendering::RTShaderSystem, 877
- SSLM_PerVertexLighting
 - gazebo::rendering::RTShaderSystem, 877
- STDERR
 - gazebo::common::Logger, 627
- STDOUT
 - gazebo::common::Logger, 627
- STLloader
 - gazebo::common::STLloader, 1073
- STLloader.hh, 1477
 - COR3_MAX, 1479
 - FACE_MAX, 1479
 - LINE_MAX_LEN, 1479
 - ORDER_MAX, 1479
- STOP_CFM
 - gazebo::physics::Joint, 547
- STOP_ERP
 - gazebo::physics::Joint, 547
- SUSPENSION_CFM
 - gazebo::physics::Joint, 547
- SUSPENSION_ERP
 - gazebo::physics::Joint, 547
- SYSTEM_PLUGIN
 - Common, 41
- samples
 - gazebo::common::MovingWindowFilterPrivate, 716
- Save
 - gazebo::physics::World, 1250
- saveCount
 - gazebo::rendering::Camera, 223
- SaveFrame
 - gazebo::rendering::Camera, 217
 - gazebo::sensors::CameraSensor, 231
 - gazebo::sensors::DepthCameraSensor, 389
 - gazebo::sensors::MultiCameraSensor, 722
- saveFrameBuffer
 - gazebo::rendering::Camera, 223
- SavePNG
 - gazebo::common::Image, 519
- SaveSimbodyState
 - gazebo::physics::SimbodyHingeJoint, 959
 - gazebo::physics::SimbodyJoint, 967
 - gazebo::physics::SimbodyLink, 978
- Scale
 - gazebo::common::Mesh, 666
 - gazebo::common::NodeAnimation, 741
 - gazebo::common::Skeleton, 1029
 - gazebo::common::SkeletonAnimation, 1034
 - gazebo::common::SubMesh, 1082
 - gazebo::math::Quaternion, 835
- scale
 - gazebo::physics::Entity, 415
 - gazebo::physics::Shape, 935
 - gazebo::rendering::VisualPrivate, 1223
- scaleVisual
 - gazebo::rendering::SelectionObjPrivate, 906
- ScaleXAxis
 - gazebo::rendering::AxisVisual, 164
- scaleXVisual
 - gazebo::rendering::SelectionObjPrivate, 906
- ScaleYAxis
 - gazebo::rendering::AxisVisual, 165
- scaleYVisual
 - gazebo::rendering::SelectionObjPrivate, 906
- ScaleZAxis
 - gazebo::rendering::AxisVisual, 165
- scaleZVisual
 - gazebo::rendering::SelectionObjPrivate, 906
- scanElem
 - gazebo::physics::MultiRayShape, 731
 - gazebo::sensors::GpuRaySensor, 488
- Scene
 - gazebo::rendering::Scene, 883
- scene

- gazebo::rendering::Camera, 224
- gazebo::rendering::VisualPrivate, 1224
- gazebo::sensors::Sensor, 918
- Scene.hh, 1438
- SceneFromSDF
 - Messages, 64
- sceneNode
 - gazebo::rendering::Camera, 224
 - gazebo::rendering::ContactVisualPrivate::Contact-Point, 286
 - gazebo::rendering::VisualPrivate, 1224
- ScenePtr
 - gazebo::rendering, 128
- scopedName
 - gazebo::rendering::Camera, 224
- scopedUniqueName
 - gazebo::rendering::Camera, 224
- screenshotPath
 - gazebo::rendering::Camera, 224
- ScrewJoint
 - gazebo::physics::ScrewJoint, 897
- ScrewJoint.hh, 1439
- scriptLength
 - gazebo::physics::Actor, 144
- scroll
 - gazebo::common::MouseEvent, 709
- sdf
 - gazebo::physics::Base, 180
 - gazebo::physics::PhysicsEngine, 781
 - gazebo::rendering::Camera, 224
 - gazebo::rendering::VisualPrivate, 1224
 - gazebo::sensors::Sensor, 918
- sec
 - gazebo::common::Time, 1120
- SecToNano
 - gazebo::common::Time, 1119
- SelectVisual
 - gazebo::rendering::Scene, 893
- selectedVis
 - gazebo::rendering::SelectionObjPrivate, 906
- selectionBuffer
 - gazebo::rendering::UserCameraPrivate, 1147
- SelectionMode
 - gazebo::rendering::SelectionObj, 901
- SelectionObj
 - gazebo::rendering::SelectionObj, 901
- SelectionObj.hh, 1440
- SelectionObjPrivate.hh, 1442
- SelectionObjPtr
 - gazebo::rendering, 128
- SendMessage
 - gazebo::transport::Publisher, 822
- Sensor
 - gazebo::sensors::Sensor, 911
- Sensor.hh, 1442
- Sensor_V
 - gazebo::sensors, 133
- SensorCategory
 - gazebo::sensors, 133
- SensorFactor, 919
- SensorFactory.hh, 1443
- SensorFactoryFn
 - gazebo::sensors, 133
- SensorManager.hh, 1444
- SensorPlugin
 - gazebo::SensorPlugin, 925
- SensorPluginPtr
 - gazebo, 103
- SensorPtr
 - gazebo::sensors, 133
- SensorTypes.hh, 1447
- Sensors, 87
 - create_sensor, 89
 - disable, 89
 - enable, 90
 - fini, 90
 - GZ_REGISTER_STATIC_SENSOR, 89
 - get_sensor, 90
 - init, 90
 - load, 90
 - remove_sensor, 90
 - remove_sensors, 91
 - run_once, 91
 - run_threads, 91
 - stop, 91
- SensorsIface.hh, 1445
- SensorsInitialized
 - gazebo::sensors::SensorManager, 924
- Server
 - gazebo::Server, 926
- Server.hh, 1449
- Set
 - gazebo::common::Color, 258
 - gazebo::common::NodeTransform, 746
 - gazebo::common::Time, 1119
 - gazebo::math::Matrix4, 659
 - gazebo::math::Plane, 790
 - gazebo::math::Pose, 804
 - gazebo::math::Quaternion, 835
 - gazebo::math::Vector2d, 1155
 - gazebo::math::Vector2i, 1163
 - gazebo::math::Vector3, 1176
 - gazebo::math::Vector4, 1186
 - Messages, 64–66
- SetActive
 - gazebo::sensors::DepthCameraSensor, 389
 - gazebo::sensors::Sensor, 916
- SetAltitude

- gazebo::physics::DARTPlaneShape, 362
- gazebo::physics::PlaneShape, 793
- gazebo::physics::SimbodyPlaneShape, 999
- SetAmbient
 - gazebo::common::Material, 644
 - gazebo::rendering::Visual, 1211
- SetAmbientColor
 - gazebo::rendering::Scene, 893
- SetAnchor
 - gazebo::physics::DARTJoint, 335
 - gazebo::physics::DARTScrewJoint, 369
 - gazebo::physics::Joint, 560
 - gazebo::physics::SimbodyJoint, 967
- SetAngle
 - gazebo::physics::Joint, 561
- SetAngleMax
 - gazebo::sensors::GpuRaySensor, 486
- SetAngleMin
 - gazebo::sensors::GpuRaySensor, 486
- SetAngularAccel
 - gazebo::physics::Link, 613
 - gazebo::physics::Model, 689
- SetAngularDamping
 - gazebo::physics::DARTLink, 345
 - gazebo::physics::Link, 613
 - gazebo::physics::SimbodyLink, 978
- SetAngularVel
 - gazebo::physics::DARTLink, 345
 - gazebo::physics::Link, 613
 - gazebo::physics::Model, 689
 - gazebo::physics::SimbodyLink, 978
- SetAnimation
 - gazebo::physics::Entity, 412
- SetAspectRatio
 - gazebo::rendering::Camera, 218
- SetAttenuation
 - gazebo::rendering::Light, 593
- SetAttribute
 - gazebo::physics::DARTJoint, 335
 - gazebo::physics::Joint, 561
 - gazebo::physics::SimbodyJoint, 967
 - gazebo::physics::SimbodyScrewJoint, 1007
- SetAutoCalculate
 - gazebo::math::RotationSpline, 874
 - gazebo::math::Spline, 1066
- SetAutoDisable
 - gazebo::physics::DARTLink, 345
 - gazebo::physics::Link, 613
 - gazebo::physics::Model, 689
 - gazebo::physics::SimbodyLink, 978
- SetAutoDisableFlag
 - gazebo::physics::PhysicsEngine, 776
- SetAxis
 - gazebo::physics::DARTBallJoint, 306
 - gazebo::physics::DARTHinge2Joint, 321
 - gazebo::physics::DARTHingeJoint, 326
 - gazebo::physics::DARTScrewJoint, 369
 - gazebo::physics::DARTSliderJoint, 374
 - gazebo::physics::DARTUniversalJoint, 382
 - gazebo::physics::Joint, 561
 - gazebo::physics::SimbodyBallJoint, 940
 - gazebo::physics::SimbodyHinge2Joint, 954
 - gazebo::physics::SimbodyHingeJoint, 959
 - gazebo::physics::SimbodyJoint, 968
 - gazebo::physics::SimbodyScrewJoint, 1007
 - gazebo::physics::SimbodySliderJoint, 1013
 - gazebo::physics::SimbodyUniversalJoint, 1019
- SetAxisMaterial
 - gazebo::rendering::AxisVisual, 165
- SetBackgroundColor
 - gazebo::rendering::Scene, 893
- SetBasePath
 - gazebo::util::LogRecord, 636
- SetBaseline
 - gazebo::rendering::MovableText, 713
- SetBindShapeTransform
 - gazebo::common::Skeleton, 1029
- SetBlendFactors
 - gazebo::common::Material, 645
- SetBlendMode
 - gazebo::common::Material, 645
- SetCallbackId
 - gazebo::transport::Subscriber, 1087
- SetCamera
 - gazebo::rendering::WindowManager, 1227
 - gazebo::sensors::ImageGaussianNoiseModel, 522
 - gazebo::sensors::Noise, 750
- SetCameraCount
 - gazebo::rendering::GpuLaser, 473
- SetCanonicalLink
 - gazebo::physics::Entity, 413
- SetCaptureData
 - gazebo::rendering::Camera, 218
- SetCaptureDataOnce
 - gazebo::rendering::Camera, 218
- SetCastShadows
 - gazebo::rendering::Light, 593
 - gazebo::rendering::Visual, 1211
- SetCategoryBits
 - gazebo::physics::Collision, 243
 - gazebo::physics::DARTCollision, 312
 - gazebo::physics::SimbodyCollision, 946
- SetCellCount
 - gazebo::rendering::Grid, 491
- SetCellLength
 - gazebo::rendering::Grid, 491
- SetCharHeight
 - gazebo::rendering::MovableText, 713

- SetClipDist
 - gazebo::rendering::Camera, 218
- SetCmd
 - gazebo::common::PID, 786
- SetCmdMax
 - gazebo::common::PID, 786
- SetCmdMin
 - gazebo::common::PID, 787
- SetCoG
 - gazebo::physics::Inertial, 535, 536
- SetCol
 - gazebo::math::Matrix3, 652
- SetCollideBits
 - gazebo::physics::Collision, 243
 - gazebo::physics::DARTCollision, 313
 - gazebo::physics::SimbodyCollision, 946
- SetCollideMode
 - gazebo::physics::Link, 614
 - gazebo::physics::Model, 690
- SetCollision
 - gazebo::physics::Collision, 243
- SetCollisionShape
 - gazebo::physics::SimbodyCollision, 946
- SetColor
 - gazebo::rendering::DynamicLines, 399
 - gazebo::rendering::Grid, 491
 - gazebo::rendering::MovableText, 713
- SetComponent
 - gazebo::common::NodeTransform, 746
- SetContactMaxCorrectingVel
 - gazebo::physics::PhysicsEngine, 776
- SetContactSurfaceLayer
 - gazebo::physics::PhysicsEngine, 776
- SetContactsEnabled
 - gazebo::physics::Collision, 243
- SetCosHorzFOV
 - gazebo::rendering::GpuLaser, 473
- SetCosVertFOV
 - gazebo::rendering::GpuLaser, 474
- SetCustomNoiseCallback
 - gazebo::sensors::Noise, 751
- SetDARTCollisionShape
 - gazebo::physics::DARTCollision, 313
- SetDARTParentJoint
 - gazebo::physics::DARTLink, 346
- SetDGain
 - gazebo::common::PID, 787
- SetDamping
 - gazebo::physics::DARTJoint, 335
 - gazebo::physics::Joint, 562
 - gazebo::physics::SimbodyJoint, 968
- SetDepthTarget
 - gazebo::rendering::DepthCamera, 386
- SetDepthWrite
 - gazebo::common::Material, 645
- SetDiffuse
 - gazebo::common::Material, 645
 - gazebo::rendering::Visual, 1212
- SetDiffuseColor
 - gazebo::rendering::Light, 593
- SetDirection
 - gazebo::rendering::Light, 593
- SetDirtyPose
 - gazebo::physics::SimbodyLink, 979
- SetDistance
 - gazebo::rendering::OrbitViewController, 766
- SetEffortLimit
 - gazebo::physics::Joint, 562
- SetElevationReference
 - gazebo::common::SphericalCoordinates, 1061
- SetEmissive
 - gazebo::common::Material, 645
 - gazebo::rendering::LaserVisual, 587
 - gazebo::rendering::Visual, 1212
- SetEnabled
 - gazebo::physics::DARTLink, 346
 - gazebo::physics::Link, 614
 - gazebo::physics::Model, 690
 - gazebo::physics::SimbodyLink, 979
 - gazebo::rendering::ContactVisual, 293
 - gazebo::rendering::Projector, 812
 - gazebo::rendering::ViewController, 1195
 - gazebo::rendering::WrenchVisual, 1260
- SetFarClip
 - gazebo::rendering::GpuLaser, 474
- SetFiducial
 - gazebo::physics::RayShape, 853
- SetFile
 - gazebo::common::AudioDecoder, 163
- SetFocalPoint
 - gazebo::rendering::OrbitViewController, 766
 - gazebo::rendering::UserCamera, 1144
- SetFog
 - gazebo::rendering::Scene, 893
- SetFontName
 - gazebo::rendering::MovableText, 713
- SetForce
 - gazebo::physics::DARTJoint, 335
 - gazebo::physics::DARTLink, 346
 - gazebo::physics::Joint, 562
 - gazebo::physics::Link, 614
 - gazebo::physics::SimbodyJoint, 968
 - gazebo::physics::SimbodyLink, 979
- SetForceImpl
 - gazebo::physics::DARTBallJoint, 306
 - gazebo::physics::DARTHinge2Joint, 321
 - gazebo::physics::DARTHingeJoint, 326
 - gazebo::physics::DARTJoint, 336

- gazebo::physics::DARTScrewJoint, 369
- gazebo::physics::DARTSliderJoint, 375
- gazebo::physics::DARTUniversalJoint, 382
- gazebo::physics::SimbodyBallJoint, 940
- gazebo::physics::SimbodyHinge2Joint, 954
- gazebo::physics::SimbodyHingeJoint, 959
- gazebo::physics::SimbodyJoint, 968
- gazebo::physics::SimbodyScrewJoint, 1007
- gazebo::physics::SimbodySliderJoint, 1013
- gazebo::physics::SimbodyUniversalJoint, 1019
- SetFromABGR
 - gazebo::common::Color, 258
- SetFromARGB
 - gazebo::common::Color, 258
- SetFromAxes
 - gazebo::math::Matrix3, 652
- SetFromAxis
 - gazebo::math::Matrix3, 652
 - gazebo::math::Quaternion, 835
- SetFromBGRA
 - gazebo::common::Color, 258
- SetFromData
 - gazebo::common::Image, 520
- SetFromDegree
 - gazebo::math::Angle, 152
- SetFromEuler
 - gazebo::math::Quaternion, 836
- SetFromHSV
 - gazebo::common::Color, 258
- SetFromRGBA
 - gazebo::common::Color, 259
- SetFromRadian
 - gazebo::math::Angle, 152
- SetFromYUV
 - gazebo::common::Color, 259
- SetGain
 - gazebo::util::OpenALSource, 761
- SetGearboxRatio
 - gazebo::physics::GearboxJoint, 463
- SetGlobal
 - gazebo::rendering::SelectionObj, 902
- SetGravity
 - gazebo::physics::DARTPhysics, 359
 - gazebo::physics::PhysicsEngine, 777
 - gazebo::physics::SimbodyPhysics, 994
- SetGravityMode
 - gazebo::physics::DARTLink, 346
 - gazebo::physics::Link, 614
 - gazebo::physics::Model, 690
 - gazebo::physics::SimbodyLink, 979
- SetGrid
 - gazebo::rendering::Scene, 894
- SetHFOV
 - gazebo::rendering::Camera, 218
- SetHandle
 - gazebo::common::SkeletonNode, 1041
- SetHeadingOffset
 - gazebo::common::SphericalCoordinates, 1061
- SetHeight
 - gazebo::rendering::Grid, 491
- SetHighStop
 - gazebo::physics::DARTBallJoint, 306
 - gazebo::physics::DARTJoint, 336
 - gazebo::physics::Joint, 562
 - gazebo::physics::SimbodyBallJoint, 940
 - gazebo::physics::SimbodyJoint, 969
 - gazebo::physics::SimbodyScrewJoint, 1008
- SetHighlighted
 - gazebo::rendering::Visual, 1212
- SetHorzFOV
 - gazebo::rendering::GpuLaser, 474
- SetHorzHalfAngle
 - gazebo::rendering::GpuLaser, 474
- SetIGain
 - gazebo::common::PID, 787
- SetIMax
 - gazebo::common::PID, 787
- SetIMin
 - gazebo::common::PID, 787
- SetIXX
 - gazebo::physics::Inertial, 537
- SetIXY
 - gazebo::physics::Inertial, 537
- SetIXZ
 - gazebo::physics::Inertial, 537
- SetIYY
 - gazebo::physics::Inertial, 537
- SetIYZ
 - gazebo::physics::Inertial, 537
- SetIZZ
 - gazebo::physics::Inertial, 537
- SetId
 - gazebo::common::SkeletonNode, 1041
 - gazebo::rendering::Visual, 1212
- SetImageHeight
 - gazebo::rendering::Camera, 218
- SetImageSize
 - gazebo::rendering::Camera, 219
- SetImageWidth
 - gazebo::rendering::Camera, 219
- SetIndexCount
 - gazebo::common::SubMesh, 1082
- SetInertiaMatrix
 - gazebo::physics::Inertial, 536
- SetInertial
 - gazebo::physics::Link, 614
- SetInitialRelativePose
 - gazebo::physics::Entity, 413

- SetInitialTransform
 - gazebo::common::SkeletonNode, 1042
- SetInverseBindTransform
 - gazebo::common::SkeletonNode, 1042
- SetIsHorizontal
 - gazebo::rendering::GpuLaser, 474
- SetJointAnimation
 - gazebo::physics::Model, 690
- SetJointPosition
 - gazebo::physics::JointController, 571
 - gazebo::physics::Model, 690
- SetJointPositions
 - gazebo::physics::JointController, 571
 - gazebo::physics::Model, 691
- SetKinematic
 - gazebo::physics::DARTLink, 346
 - gazebo::physics::Link, 614
- SetLaserRetro
 - gazebo::physics::Collision, 244
 - gazebo::physics::Link, 615
 - gazebo::physics::Model, 691
- SetLatching
 - gazebo::transport::CallbackHelper, 194
- SetLatitudeReference
 - gazebo::common::SphericalCoordinates, 1061
- SetLength
 - gazebo::common::Animation, 156
 - gazebo::physics::CylinderShape, 300
 - gazebo::physics::RayShape, 853
- SetLightType
 - gazebo::rendering::Light, 593
- SetLighting
 - gazebo::common::Material, 646
 - gazebo::rendering::Visual, 1212
- SetLineWidth
 - gazebo::rendering::Grid, 491
- SetLinearAccel
 - gazebo::physics::Link, 615
 - gazebo::physics::Model, 691
- SetLinearDamping
 - gazebo::physics::DARTLink, 347
 - gazebo::physics::Link, 615
 - gazebo::physics::SimbodyLink, 979
- SetLinearVel
 - gazebo::physics::DARTLink, 347
 - gazebo::physics::Link, 615
 - gazebo::physics::Model, 691
 - gazebo::physics::SimbodyLink, 979
- SetLinkStatic
 - gazebo::physics::DARTLink, 347
 - gazebo::physics::Link, 615
 - gazebo::physics::SimbodyLink, 980
- SetLinkWorldPose
 - gazebo::physics::Model, 691, 692
- SetLocallyAdvertised
 - gazebo::transport::Publication, 817
- SetLongitudeReference
 - gazebo::common::SphericalCoordinates, 1061
- SetLoop
 - gazebo::util::OpenALSource, 762
- SetLowStop
 - gazebo::physics::DARTBallJoint, 307
 - gazebo::physics::DARTJoint, 336
 - gazebo::physics::Joint, 563
 - gazebo::physics::SimbodyBallJoint, 941
 - gazebo::physics::SimbodyJoint, 969
 - gazebo::physics::SimbodyScrewJoint, 1008
- SetLowerLimit
 - gazebo::physics::Joint, 563
- SetMOI
 - gazebo::physics::Inertial, 538
- SetMass
 - gazebo::physics::Inertial, 538
- SetMaterial
 - gazebo::rendering::Visual, 1212
- SetMaterialIndex
 - gazebo::common::SubMesh, 1082
- SetMaxContacts
 - gazebo::physics::Collision, 244
 - gazebo::physics::PhysicsEngine, 777
- SetMaxForce
 - gazebo::physics::DARTBallJoint, 307
 - gazebo::physics::DARTHinge2Joint, 322
 - gazebo::physics::DARTHingeJoint, 327
 - gazebo::physics::DARTScrewJoint, 370
 - gazebo::physics::DARTSliderJoint, 375
 - gazebo::physics::DARTUniversalJoint, 382
 - gazebo::physics::Joint, 563
 - gazebo::physics::SimbodyBallJoint, 941
 - gazebo::physics::SimbodyHinge2Joint, 954
 - gazebo::physics::SimbodyHingeJoint, 959
 - gazebo::physics::SimbodyScrewJoint, 1008
 - gazebo::physics::SimbodySliderJoint, 1013
 - gazebo::physics::SimbodyUniversalJoint, 1020
- SetMaxStepSize
 - gazebo::physics::PhysicsEngine, 777
- SetMesh
 - gazebo::physics::MeshShape, 677
- setMinimalComms
 - Transport, 98
- SetMode
 - gazebo::rendering::SelectionObj, 902
- SetModel
 - gazebo::physics::Joint, 563
- SetModelTransform
 - gazebo::common::SkeletonNode, 1042
- SetMuPrimary
 - gazebo::physics::FrictionPyramid, 456

- SetMuSecondary
 - gazebo::physics::FrictionPyramid, 457
- SetName
 - gazebo::common::Mesh, 666
 - gazebo::common::NodeAnimation, 741
 - gazebo::common::SkeletonAnimation, 1034
 - gazebo::common::SkeletonNode, 1042
 - gazebo::common::SubMesh, 1082
 - gazebo::physics::Base, 179
 - gazebo::physics::Entity, 413
 - gazebo::physics::State, 1071
 - gazebo::rendering::Camera, 219
 - gazebo::rendering::Light, 594
 - gazebo::rendering::Visual, 1213
- SetNearClip
 - gazebo::rendering::GpuLaser, 474
- SetNode
 - gazebo::transport::Publisher, 823
- SetNormal
 - gazebo::common::SubMesh, 1082
 - gazebo::physics::PlaneShape, 793
- SetNormalCount
 - gazebo::common::SubMesh, 1083
- SetNormalMap
 - gazebo::rendering::Visual, 1213
- SetNumVertAttached
 - gazebo::common::Skeleton, 1030
- SetOperationType
 - gazebo::rendering::DynamicRenderable, 403
- SetPGain
 - gazebo::common::PID, 787
- SetParam
 - gazebo::physics::DARTJoint, 336
 - gazebo::physics::DARTPhysics, 359
 - gazebo::physics::Joint, 564
 - gazebo::physics::PhysicsEngine, 777
 - gazebo::physics::SimbodyJoint, 969
 - gazebo::physics::SimbodyPhysics, 994
 - gazebo::physics::SimbodyScrewJoint, 1008
- SetParams
 - gazebo::Server, 928
- SetParent
 - gazebo::common::SkeletonNode, 1042
 - gazebo::physics::Base, 179
 - gazebo::sensors::Sensor, 916
- SetPath
 - gazebo::common::Mesh, 666
- SetPaused
 - gazebo::physics::World, 1250
 - gazebo::util::LogRecord, 636
- SetPerPixelLighting
 - gazebo::rendering::RTShaderSystem, 879
- SetPitch
 - gazebo::util::OpenALSource, 762
- SetPoint
 - gazebo::rendering::DynamicLines, 399
- SetPointSize
 - gazebo::common::Material, 646
- SetPoints
 - DART Physics, 78
 - gazebo::physics::RayShape, 853
 - gazebo::physics::SimbodyRayShape, 1001
- SetPose
 - gazebo::rendering::Visual, 1213
 - gazebo::util::OpenALSink, 758
 - gazebo::util::OpenALSource, 762
- SetPosition
 - gazebo::rendering::Light, 594
 - gazebo::rendering::Visual, 1213
- SetPositionTarget
 - gazebo::physics::JointController, 571
- SetPrevMsg
 - gazebo::transport::Publication, 817
- SetPrimitiveType
 - gazebo::common::SubMesh, 1083
- SetProvideFeedback
 - gazebo::physics::Joint, 564
- SetPublication
 - gazebo::transport::Publisher, 823
- SetPublishData
 - gazebo::physics::Link, 616
- SetQuiet
 - gazebo::common::Console, 278
- SetRadius
 - gazebo::physics::CylinderShape, 300
 - gazebo::physics::DARTSphereShape, 377
 - gazebo::physics::SimbodySphereShape, 1015
 - gazebo::physics::SphereShape, 1056
- SetRange
 - gazebo::rendering::Light, 594
- SetRangeCount
 - gazebo::rendering::GpuLaser, 475
- SetRayCountRatio
 - gazebo::rendering::GpuLaser, 475
- SetRealTime
 - gazebo::physics::LinkState, 624
 - gazebo::physics::ModelState, 705
 - gazebo::physics::State, 1071
 - gazebo::physics::WorldState, 1258
- SetRealTimeUpdateRate
 - gazebo::physics::PhysicsEngine, 778
- SetReferencePose
 - gazebo::sensors::ImuSensor, 528
- SetRelativePose
 - gazebo::physics::Entity, 413
- SetRenderRate
 - gazebo::rendering::Camera, 219
- SetRenderTarget

- gazebo::rendering::Camera, 219
- gazebo::rendering::UserCamera, 1144
- SetRetro
 - gazebo::physics::RayShape, 854
- SetRibbonTrail
 - gazebo::rendering::Visual, 1213
- SetRootNode
 - gazebo::common::Skeleton, 1030
- SetRotation
 - gazebo::common::PoseKeyFrame, 809
 - gazebo::rendering::Visual, 1214
- SetSID
 - gazebo::common::NodeTransform, 746
- SetSORPGSIters
 - gazebo::physics::PhysicsEngine, 778
- SetSORPGSPreconIters
 - gazebo::physics::PhysicsEngine, 779
- SetSORPGSW
 - gazebo::physics::PhysicsEngine, 779
- SetSaveFramePathname
 - gazebo::rendering::Camera, 219
- SetSaveable
 - gazebo::physics::Base, 179
- SetScale
 - gazebo::common::Mesh, 667
 - gazebo::common::SubMesh, 1083
 - gazebo::math::Matrix4, 659
 - gazebo::physics::BoxShape, 187
 - gazebo::physics::Collision, 244
 - gazebo::physics::CylinderShape, 300
 - gazebo::physics::HeightmapShape, 511
 - gazebo::physics::Link, 616
 - gazebo::physics::MeshShape, 677
 - gazebo::physics::Model, 692
 - gazebo::physics::MultiRayShape, 730
 - gazebo::physics::PlaneShape, 794
 - gazebo::physics::RayShape, 854
 - gazebo::physics::Shape, 935
 - gazebo::physics::SphereShape, 1057
 - gazebo::rendering::Visual, 1214
- SetScene
 - gazebo::rendering::Camera, 220
 - gazebo::rendering::Visual, 1214
- SetSceneNode
 - gazebo::rendering::Camera, 220
- SetSeed
 - gazebo::math::Rand, 839
 - gazebo::physics::DARTPhysics, 360
 - gazebo::physics::PhysicsEngine, 778
 - gazebo::physics::SimbodyPhysics, 995
- SetSelected
 - gazebo::physics::Base, 179
 - gazebo::physics::Link, 616
 - gazebo::rendering::Light, 594
- setSelectedEntity
 - gazebo::event::Events, 432
- SetSelfCollide
 - gazebo::physics::DARTLink, 347
 - gazebo::physics::Link, 616
 - gazebo::physics::SimbodyLink, 980
- SetShadeMode
 - gazebo::common::Material, 646
- SetShaderType
 - gazebo::rendering::Visual, 1214
- SetShadowsEnabled
 - gazebo::rendering::Scene, 894
- SetShape
 - gazebo::physics::Collision, 244
- SetShininess
 - gazebo::common::Material, 646
- SetShowOnTop
 - gazebo::rendering::MovableText, 714
- SetSimTime
 - gazebo::physics::LinkState, 624
 - gazebo::physics::ModelState, 705
 - gazebo::physics::State, 1071
 - gazebo::physics::World, 1250
 - gazebo::physics::WorldState, 1258
- SetSize
 - gazebo::physics::BoxShape, 188
 - gazebo::physics::CylinderShape, 300
 - gazebo::physics::DARTBoxShape, 309
 - gazebo::physics::DARTCylinderShape, 315
 - gazebo::physics::PlaneShape, 794
 - gazebo::physics::SimbodyBoxShape, 943
 - gazebo::physics::SimbodyCylinderShape, 948
- SetSkeleton
 - gazebo::common::Mesh, 667
- SetSkeletonPose
 - gazebo::rendering::Visual, 1214
- SetSkyXMode
 - gazebo::rendering::Scene, 894
- SetSourceValues
 - gazebo::common::NodeTransform, 746, 747
- SetSpaceWidth
 - gazebo::rendering::MovableText, 714
- SetSpecular
 - gazebo::common::Material, 646
 - gazebo::rendering::Visual, 1214
- SetSpecularColor
 - gazebo::rendering::Light, 594
- SetSpotFalloff
 - gazebo::rendering::Light, 594
- SetSpotInnerAngle
 - gazebo::rendering::Light, 595
- SetSpotOuterAngle
 - gazebo::rendering::Light, 595
- SetState

- gazebo::physics::Collision, 244
- gazebo::physics::Joint, 564
- gazebo::physics::Link, 616
- gazebo::physics::Model, 692
- gazebo::physics::World, 1250
- gazebo::rendering::SelectionObj, 903
- SetStatic
 - gazebo::physics::Entity, 413
- SetStiffness
 - gazebo::physics::DARTJoint, 337
 - gazebo::physics::Joint, 564
 - gazebo::physics::SimbodyJoint, 969
- SetStiffnessDamping
 - gazebo::physics::DARTJoint, 337
 - gazebo::physics::Joint, 564
 - gazebo::physics::SimbodyJoint, 970
- SetStopDissipation
 - gazebo::physics::Joint, 565
- SetStopStiffness
 - gazebo::physics::Joint, 565
- SetSubMeshCenter
 - gazebo::common::SubMesh, 1083
- SetSurfaceType
 - gazebo::common::SphericalCoordinates, 1062
- SetTargetRealTimeFactor
 - gazebo::physics::PhysicsEngine, 779
- SetTension
 - gazebo::math::Spline, 1066
- SetTexCoord
 - gazebo::common::SubMesh, 1083
- SetTexCoordCount
 - gazebo::common::SubMesh, 1083
- SetText
 - gazebo::rendering::MovableText, 714
- SetTextAlignment
 - gazebo::rendering::MovableText, 714
- SetTexture
 - gazebo::rendering::Projector, 812
- SetTextureImage
 - gazebo::common::Material, 646
- SetThreadPitch
 - gazebo::physics::DARTScrewJoint, 370
 - gazebo::physics::ScrewJoint, 898, 899
 - gazebo::physics::SimbodyScrewJoint, 1009
- SetTime
 - gazebo::common::Animation, 156
- SetTolIdentity
 - gazebo::math::Quaternion, 836
- SetToMax
 - gazebo::math::Vector3, 1176
- SetToMin
 - gazebo::math::Vector3, 1176
- SetToWallTime
 - gazebo::common::Time, 1119
- SetTorque
 - gazebo::physics::DARTLink, 347
 - gazebo::physics::Link, 616
 - gazebo::physics::SimbodyLink, 980
- SetTransform
 - gazebo::common::SkeletonNode, 1042
- SetTranslate
 - gazebo::math::Matrix4, 659
- SetTranslation
 - gazebo::common::PoseKeyFrame, 809
- SetTransparency
 - gazebo::common::Material, 647
 - gazebo::rendering::Visual, 1215
- SetTransparent
 - gazebo::rendering::Scene, 894
- SetType
 - gazebo::common::NodeTransform, 747
 - gazebo::common::SkeletonNode, 1043
- SetUpdateRate
 - gazebo::sensors::Sensor, 916
- SetUpperLimit
 - gazebo::physics::Joint, 565
- SetUseSDFPose
 - gazebo::rendering::UserCamera, 1145
- SetUserData
 - gazebo::rendering::Grid, 492
- SetValue
 - gazebo::common::NumericKeyFrame, 755
- SetVelocity
 - gazebo::physics::DARTBallJoint, 307
 - gazebo::physics::DARTHinge2Joint, 322
 - gazebo::physics::DARTHingeJoint, 327
 - gazebo::physics::DARTScrewJoint, 370
 - gazebo::physics::DARTSliderJoint, 375
 - gazebo::physics::DARTUniversalJoint, 382
 - gazebo::physics::Joint, 565
 - gazebo::physics::SimbodyBallJoint, 941
 - gazebo::physics::SimbodyHinge2Joint, 954
 - gazebo::physics::SimbodyHingeJoint, 960
 - gazebo::physics::SimbodyScrewJoint, 1009
 - gazebo::physics::SimbodySliderJoint, 1014
 - gazebo::physics::SimbodyUniversalJoint, 1020
 - gazebo::util::OpenALSink, 758
 - gazebo::util::OpenALSOURCE, 762
- SetVelocityTarget
 - gazebo::physics::JointController, 572
- SetVertFOV
 - gazebo::rendering::GpuLaser, 475
- SetVertHalfAngle
 - gazebo::rendering::GpuLaser, 475
- SetVertex
 - gazebo::common::SubMesh, 1084
- SetVertexCount
 - gazebo::common::SubMesh, 1084

- SetVerticalAngleMax
 - gazebo::sensors::GpuRaySensor, 486
- SetVerticalAngleMin
 - gazebo::sensors::GpuRaySensor, 487
- SetViewController
 - gazebo::rendering::UserCamera, 1145
- SetViewportDimensions
 - gazebo::rendering::UserCamera, 1145
- SetVisibilityFlags
 - gazebo::rendering::Visual, 1215
- SetVisible
 - gazebo::rendering::Scene, 894
 - gazebo::rendering::Visual, 1215
 - gazebo::rendering::WireBox, 1229
- SetWallTime
 - gazebo::physics::LinkState, 624
 - gazebo::physics::ModelState, 706
 - gazebo::physics::State, 1072
 - gazebo::physics::WorldState, 1258
- SetWindowId
 - gazebo::rendering::Camera, 220
- SetWindowSize
 - Common, 45
- SetWireframe
 - gazebo::rendering::Heightmap, 503
 - gazebo::rendering::Scene, 895
 - gazebo::rendering::Visual, 1215
- SetWorld
 - gazebo::physics::Base, 180
 - gazebo::physics::WorldState, 1258
- SetWorldCFM
 - gazebo::physics::PhysicsEngine, 779
- SetWorldERP
 - gazebo::physics::PhysicsEngine, 779
- SetWorldPose
 - gazebo::physics::Entity, 413
 - gazebo::rendering::Camera, 220
 - gazebo::rendering::UserCamera, 1145
 - gazebo::rendering::Visual, 1215
- SetWorldPosition
 - gazebo::rendering::Camera, 220
 - gazebo::rendering::Visual, 1216
- SetWorldRotation
 - gazebo::rendering::Camera, 220
 - gazebo::rendering::Visual, 1216
- SetWorldTwist
 - gazebo::physics::Entity, 414
- setupClient
 - gazebo, 105
- setupServer
 - gazebo, 105
- ShadeMode
 - gazebo::common::Material, 641
- shadeMode
 - gazebo::common::Material, 648
- ShadeModeStr
 - gazebo::common::Material, 648
- shaftNode
 - gazebo::rendering::ArrowVisualPrivate, 160
- Shape
 - gazebo::physics::Shape, 934
- shape
 - gazebo::physics::Collision, 245
- Shape.hh, 1450
- ShapePtr
 - gazebo::physics, 123
- shift
 - gazebo::common::MouseEvent, 709
- shininess
 - gazebo::common::Material, 648
- Show
 - gazebo::rendering::GUIOverlay, 497
- ShowBoundingBox
 - gazebo::rendering::Visual, 1216
- ShowCOM
 - gazebo::rendering::Visual, 1216
- ShowCOMs
 - gazebo::rendering::Scene, 895
- ShowClouds
 - gazebo::rendering::Scene, 895
- ShowCollision
 - gazebo::rendering::Visual, 1216
- ShowCollisions
 - gazebo::rendering::Scene, 895
- ShowContacts
 - gazebo::rendering::Scene, 895
- ShowJoints
 - gazebo::rendering::Scene, 895
 - gazebo::rendering::Visual, 1216
- ShowRotation
 - gazebo::rendering::ArrowVisual, 159
 - gazebo::rendering::AxisVisual, 165
- ShowSkeleton
 - gazebo::rendering::Visual, 1217
- ShowVisual
 - gazebo::rendering::Light, 595
- ShowWireframe
 - gazebo::rendering::Camera, 220
- Shutdown
 - gazebo::transport::Connection, 270
- shutdown
 - gazebo, 106
- sid
 - gazebo::common::NodeTransform, 747
- sigInt
 - gazebo::event::Events, 432
- Signal
 - gazebo::event::EventT, 440–442

signalPropagationSub
 gazebo::rendering::TransmitterVisualPrivate, 1134
 signaled
 gazebo::event::EventPrivate, 421
 SimTK, 137
 simTime
 gazebo::common::UpdateInfo, 1137
 gazebo::physics::State, 1072
 SimTimeEventHandler
 gazebo::sensors::SimTimeEventHandler, 1021
 Simbody Physics, 81
 simbody_inc.h, 1451
 SimbodyBallJoint
 gazebo::physics::SimbodyBallJoint, 937
 SimbodyBallJoint.hh, 1451
 SimbodyBoxShape
 gazebo::physics::SimbodyBoxShape, 942
 SimbodyBoxShape.hh, 1452
 SimbodyCollision
 gazebo::physics::SimbodyCollision, 945
 SimbodyCollision.hh, 1452
 SimbodyCollisionPtr
 gazebo::physics, 123
 SimbodyCylinderShape
 gazebo::physics::SimbodyCylinderShape, 947
 SimbodyCylinderShape.hh, 1453
 SimbodyHeightmapShape
 gazebo::physics::SimbodyHeightmapShape, 949
 SimbodyHeightmapShape.hh, 1453
 SimbodyHinge2Joint
 gazebo::physics::SimbodyHinge2Joint, 952
 SimbodyHinge2Joint.hh, 1454
 SimbodyHingeJoint
 gazebo::physics::SimbodyHingeJoint, 957
 SimbodyHingeJoint.hh, 1455
 SimbodyJoint
 gazebo::physics::SimbodyJoint, 962
 SimbodyJoint.hh, 1455
 SimbodyLink
 gazebo::physics::SimbodyLink, 974
 SimbodyLink.hh, 1456
 SimbodyLinkPtr
 gazebo::physics, 123
 SimbodyMeshShape
 gazebo::physics::SimbodyMeshShape, 982
 SimbodyMeshShape.hh, 1457
 SimbodyModel
 gazebo::physics::SimbodyModel, 983
 SimbodyModel.hh, 1457
 SimbodyModelPtr
 gazebo::physics, 123
 SimbodyMultiRayShape
 gazebo::physics::SimbodyMultiRayShape, 986
 SimbodyMultiRayShape.hh, 1458
 SimbodyPhysics
 gazebo::physics::SimbodyPhysics, 989
 simbodyPhysics
 gazebo::physics::SimbodyJoint, 971
 SimbodyPhysics.hh, 1458
 simbodyPhysicsInitialized
 gazebo::physics::SimbodyPhysics, 997
 SimbodyPhysicsPtr
 gazebo::physics, 123
 simbodyPhysicsStepped
 gazebo::physics::SimbodyPhysics, 997
 SimbodyPlaneShape
 gazebo::physics::SimbodyPlaneShape, 998
 SimbodyPlaneShape.hh, 1459
 SimbodyRayShape
 gazebo::physics::SimbodyRayShape, 1001
 SimbodyRayShape.hh, 1460
 SimbodyRayShapePtr
 gazebo::physics, 123
 SimbodyScrewJoint
 gazebo::physics::SimbodyScrewJoint, 1004
 SimbodyScrewJoint.hh, 1460
 SimbodySliderJoint
 gazebo::physics::SimbodySliderJoint, 1011
 SimbodySliderJoint.hh, 1461
 SimbodySphereShape
 gazebo::physics::SimbodySphereShape, 1015
 SimbodySphereShape.hh, 1462
 SimbodyTypes.hh, 1462
 SimbodyUniversalJoint
 gazebo::physics::SimbodyUniversalJoint, 1017
 SimbodyUniversalJoint.hh, 1464
 SingletonT
 ~SingletonT, 1024
 Instance, 1024
 SingletonT, 1024
 SingletonT, 1024
 SingletonT< T >, 1022
 SingletonT.hh, 1464
 size
 gazebo::math::Plane, 790
 skelAnimation
 gazebo::physics::Actor, 144
 skelNodesMap
 gazebo::physics::Actor, 144
 Skeleton
 gazebo::common::Skeleton, 1026
 skeleton
 gazebo::physics::Actor, 144
 gazebo::rendering::VisualPrivate, 1224
 Skeleton.hh, 1465
 SkeletonAnimation
 gazebo::common::SkeletonAnimation, 1032
 SkeletonAnimation.hh, 1467

- SkeletonNode
 - gazebo::common::SkeletonNode, 1037, 1038
- SkeletonNodeType
 - gazebo::common::SkeletonNode, 1037
- skinFile
 - gazebo::physics::Actor, 144
- skinScale
 - gazebo::physics::Actor, 145
- SkyX, 137
- SkyXMode
 - gazebo::rendering::Scene, 883
- skyx
 - gazebo::rendering::Scene, 896
- slaveMobods
 - gazebo::physics::SimbodyLink, 980
- slaveWelds
 - gazebo::physics::SimbodyLink, 980
- Sleep
 - gazebo::common::Time, 1119
- Slerp
 - gazebo::math::Quaternion, 836
- SliderJoint
 - gazebo::physics::SliderJoint, 1045
- SliderJoint.hh, 1468
- Smooth
 - gazebo::rendering::Heightmap, 503
- SnapVisualToNearestBelow
 - gazebo::rendering::Scene, 896
- sonarMsg
 - gazebo::rendering::SonarVisualPrivate, 1054
- sonarRay
 - gazebo::rendering::SonarVisualPrivate, 1054
- SonarSensor
 - gazebo::sensors::SonarSensor, 1048
- SonarSensor.hh, 1469
- SonarSensorPtr
 - gazebo::sensors, 133
- sonarSub
 - gazebo::rendering::SonarVisualPrivate, 1054
- SonarVisual
 - gazebo::rendering::SonarVisual, 1052
- SonarVisual.hh, 1470
- SonarVisualPrivate.hh, 1471
- SonarVisualPtr
 - gazebo::rendering, 128
- source
 - gazebo::common::NodeTransform, 747
- specular
 - gazebo::common::Material, 648
- SpeedOfLight
 - gazebo::common, 110
- SphereShape
 - gazebo::physics::SphereShape, 1056
- SphereShape.hh, 1471
- SphereShapePtr
 - gazebo::physics, 123
- SphericalCoordinates
 - gazebo::common::SphericalCoordinates, 1059
- SphericalCoordinates.hh, 1472
- SphericalCoordinatesPrivate.hh, 1473
- SphericalCoordinatesPtr
 - gazebo::common, 110
- SphericalFromLocal
 - gazebo::common::SphericalCoordinates, 1062
- Spline
 - gazebo::math::Spline, 1064
- Spline.hh, 1475
- SplitHeights
 - gazebo::rendering::Heightmap, 504
- spring
 - gazebo::physics::SimbodyJoint, 971
- springReferencePosition
 - gazebo::physics::Joint, 567
- Squad
 - gazebo::math::Quaternion, 836
- ssaolInstance
 - gazebo::rendering::CameraPrivate, 227
- Stamp
 - Messages, 66
- Start
 - Common, 45
 - gazebo::common::Timer, 1122
 - gazebo::util::DiagnosticTimer, 395
 - gazebo::util::LogRecord, 637
- startCacheMutex
 - gazebo::common::ModelDatabasePrivate, 695
- startDelay
 - gazebo::physics::Actor, 145
- StartRead
 - gazebo::transport::Connection, 270
- startTime
 - gazebo::physics::TrajectoryInfo, 1130
- StartTimer
 - gazebo::util::DiagnosticManager, 393
- State
 - gazebo::physics::State, 1069
- state
 - gazebo::rendering::SelectionObjPrivate, 906
- State.hh, 1476
- staticGeom
 - gazebo::rendering::VisualPrivate, 1224
- stdDev
 - gazebo::sensors::GaussianNoiseModel, 461
- Step
 - gazebo::physics::World, 1250
 - gazebo::util::LogPlay, 631
- step
 - gazebo::event::Events, 432

- StepWorld
 - gazebo::physics::World, 1250
- stiffnessCoefficient
 - gazebo::physics::Joint, 567
- Stop
 - gazebo::common::Timer, 1122
 - gazebo::Master, 639
 - gazebo::physics::Actor, 142
 - gazebo::physics::World, 1251
 - gazebo::sensors::SensorManager, 924
 - gazebo::Server, 928
 - gazebo::transport::ConnectionManager, 274
 - gazebo::transport::IOManager, 541
 - gazebo::util::DiagnosticTimer, 395
 - gazebo::util::LogRecord, 637
 - gazebo::util::OpenALSource, 763
- stop
 - gazebo, 106
 - gazebo::common::ModelDatabasePrivate, 696
 - gazebo::event::Events, 432
 - Sensors, 91
 - Transport, 98
- stop_world
 - Classes for physics and dynamics, 74
- stop_worlds
 - Classes for physics and dynamics, 74
- StopAnimation
 - gazebo::physics::Entity, 414
 - gazebo::physics::Model, 692
- StopRead
 - gazebo::transport::Connection, 270
- StopTimer
 - gazebo::util::DiagnosticManager, 393
- StrStr_M
 - gazebo::common, 110
- stream
 - gazebo::common::FileLogger::Buffer, 190
- StripSceneName
 - gazebo::rendering::Scene, 896
- StripWorldName
 - gazebo::physics::World, 1251
- SubMesh
 - gazebo::common::SubMesh, 1076
- SubNodeMap
 - gazebo::transport::TopicManager, 1124
- subSampling
 - gazebo::physics::HeightmapShape, 511
- submesh
 - gazebo::physics::MeshShape, 678
- Subscribe
 - gazebo::transport::ConnectionManager, 274
 - gazebo::transport::Node, 736, 737
 - gazebo::transport::TopicManager, 1128
- SubscribeOptions
 - gazebo::transport::SubscribeOptions, 1085
- SubscribeOptions.hh, 1479
- Subscriber
 - gazebo::transport::Subscriber, 1087
- Subscriber.hh, 1480
- SubscriberPtr
 - gazebo::transport, 136
- SubscriptionTransport
 - gazebo::transport::SubscriptionTransport, 1089
- SubscriptionTransport.hh, 1481
- SubscriptionTransportPtr
 - gazebo::transport, 136
- sum
 - gazebo::common::MovingWindowFilterPrivate, 716
- surface
 - gazebo::physics::Collision, 245
- SurfaceParams
 - gazebo::physics::SurfaceParams, 1091
- SurfaceParams.hh, 1482
- SurfaceParamsPtr
 - gazebo::physics, 123
- SurfaceType
 - gazebo::common::SphericalCoordinates, 1058
- surfaceType
 - gazebo::common::SphericalCoordinatesPrivate, 1063
- sync
 - gazebo::common::FileLogger::Buffer, 190
 - gazebo::common::Logger::Buffer, 189
- system
 - gazebo::physics::SimbodyPhysics, 997
- system.hh, 1484
 - GAZEBO_HIDDEN, 1484
 - GAZEBO_VISIBLE, 1484
- SystemPaths.hh, 1484
 - GetCurrentDir, 1486
 - LINUX, 1486
- SystemPlugin
 - gazebo::SystemPlugin, 1098
- SystemPluginPtr
 - gazebo, 103
- TPtr
 - gazebo::PluginT, 795
- TRANS
 - gazebo::rendering::SelectionObj, 901
- TRANS_X
 - gazebo::rendering::SelectionObj, 901
- TRANS_Y
 - gazebo::rendering::SelectionObj, 901
- TRANS_Z
 - gazebo::rendering::SelectionObj, 901
- TRANSLATE
 - gazebo::common::NodeTransform, 744

- TRIANGLES
 - gazebo::common::SubMesh, 1076
- TRIFANS
 - gazebo::common::SubMesh, 1076
- TRISTRIPS
 - gazebo::common::SubMesh, 1076
- tangents
 - gazebo::math::RotationSpline, 875
 - gazebo::math::Spline, 1067
- targetRealTimeFactor
 - gazebo::physics::PhysicsEngine, 781
- tension
 - gazebo::math::Spline, 1067
- texImage
 - gazebo::common::Material, 648
- texture
 - gazebo::rendering::VideoVisualPrivate, 1192
- textureHeight
 - gazebo::rendering::Camera, 224
- textureWidth
 - gazebo::rendering::Camera, 224
- threadPitch
 - gazebo::physics::ScrewJoint, 899
- Time
 - gazebo::common::Time, 1103, 1104
- time
 - gazebo::common::KeyFrame, 585
 - gazebo::physics::Contact, 282
 - gazebo::sensors::SimTimeEvent, 1021
- Time.hh, 1486
- timePos
 - gazebo::common::Animation, 157
- Timer
 - gazebo::common::Timer, 1121
- Timer.hh, 1487
- Toggle
 - gazebo::rendering::Projector, 812
- ToggleShowVisual
 - gazebo::rendering::Light, 595
- ToggleShowWireframe
 - gazebo::rendering::Camera, 221
- ToggleVisible
 - gazebo::rendering::Visual, 1217
- TopicManager.hh, 1488
- topicName
 - gazebo::rendering::ContactVisualPrivate, 295
- TrackVisual
 - gazebo::rendering::Camera, 221
- TrackVisualFromSDF
 - Messages, 66
- TrackVisualImpl
 - gazebo::rendering::Camera, 221
 - gazebo::rendering::UserCamera, 1146
- trackVisualPID
 - gazebo::rendering::CameraPrivate, 227
- trackVisualPitchPID
 - gazebo::rendering::CameraPrivate, 227
- trackVisualYawPID
 - gazebo::rendering::CameraPrivate, 227
- trackedVisual
 - gazebo::rendering::CameraPrivate, 227
- tracker
 - gazebo::physics::SimbodyPhysics, 997
- trajInfo
 - gazebo::physics::Actor, 145
- trajectories
 - gazebo::physics::Actor, 145
- TrajectoryInfo
 - gazebo::physics::TrajectoryInfo, 1130
- transVisual
 - gazebo::rendering::SelectionObjPrivate, 906
- transXVisual
 - gazebo::rendering::SelectionObjPrivate, 906
- transYVisual
 - gazebo::rendering::SelectionObjPrivate, 906
- transZVisual
 - gazebo::rendering::SelectionObjPrivate, 906
- transform
 - gazebo::common::NodeTransform, 747
 - gazebo::common::SkeletonNode, 1044
- Transform2Pose
 - gazebo::physics::SimbodyPhysics, 995
- TransformAffine
 - gazebo::math::Matrix4, 659
- TransformType
 - gazebo::common::NodeTransform, 744
- Translate
 - gazebo::common::Mesh, 667
 - gazebo::common::SubMesh, 1084
 - gazebo::rendering::Camera, 221
- translate
 - gazebo::common::PoseKeyFrame, 810
- translated
 - gazebo::physics::TrajectoryInfo, 1130
- TransmitterVisual
 - gazebo::rendering::TransmitterVisual, 1132
- TransmitterVisual.hh, 1489
- TransmitterVisualPrivate.hh, 1490
- transparency
 - gazebo::common::Material, 648
 - gazebo::rendering::VisualPrivate, 1224
- Transport, 92
 - CallbackHelperPtr, 94
 - clear_buffers, 94
 - connectToMaster, 94
 - fini, 94
 - get_master_uri, 94
 - get_topic_namespaces, 95

- getAdvertisedTopics, 95
- getMinimalComms, 95
- getTopicMsgType, 95
- init, 96
- is_stopped, 96
- pause_incoming, 96
- publish, 96
- request, 97
- requestNoReply, 97
- run, 98
- setMinimalComms, 98
- stop, 98
- waitForNamespaces, 98
- TransportInterface.hh, 1490
- TransportTypes.hh, 1493
- TriggerUpdate
 - gazebo::transport::ConnectionManager, 274
- TwoPi
 - gazebo::math::Angle, 153
- type
 - gazebo::common::KeyEvent, 584
 - gazebo::common::Logger::Buffer, 189
 - gazebo::common::MouseEvent, 709
 - gazebo::common::NodeTransform, 747
 - gazebo::common::SkeletonNode, 1044
 - gazebo::physics::TrajectoryInfo, 1130
 - gazebo::PluginT, 796
- typeString
 - gazebo::rendering::ViewController, 1196
- UIntGen
 - gazebo::math, 113
- UNION
 - gazebo::common::MeshCSG, 668
- UNIVERSAL_JOINT
 - gazebo::physics::Base, 172
- UNKNOWN_PIXEL_FORMAT
 - gazebo::common::Image, 516
- URealGen
 - gazebo::math, 113
- Unadvertise
 - gazebo::transport::ConnectionManager, 274
 - gazebo::transport::TopicManager, 1128
- UniformIntDist
 - gazebo::math, 113
- UniformRealDist
 - gazebo::math, 113
- UnitX
 - gazebo::math::Vector3, 1177
- UnitY
 - gazebo::math::Vector3, 1177
- UnitZ
 - gazebo::math::Vector3, 1178
- UniversalJoint
 - gazebo::physics::UniversalJoint, 1136
- UniversalJoint.hh, 1494
- unloadProceduralPage
 - gazebo::rendering::DummyPageProvider, 396
- unprepareProceduralPage
 - gazebo::rendering::DummyPageProvider, 396
- Unsubscribe
 - gazebo::transport::ConnectionManager, 275
 - gazebo::transport::Subscriber, 1087
 - gazebo::transport::TopicManager, 1129
- Update
 - Common, 45
 - DART Physics, 78
 - gazebo::common::PID, 788
 - gazebo::physics::Actor, 142
 - gazebo::physics::Base, 180
 - gazebo::physics::DARTMeshShape, 349
 - gazebo::physics::DARTModel, 352
 - gazebo::physics::Joint, 566
 - gazebo::physics::JointController, 572
 - gazebo::physics::Link, 617
 - gazebo::physics::MeshShape, 678
 - gazebo::physics::Model, 692
 - gazebo::physics::MultiRayShape, 730
 - gazebo::physics::RayShape, 854
 - gazebo::physics::SimbodyRayShape, 1001
 - gazebo::rendering::Camera, 222
 - gazebo::rendering::DynamicLines, 400
 - gazebo::rendering::FPSViewController, 455
 - gazebo::rendering::GUIOverlay, 497
 - gazebo::rendering::MovableText, 714
 - gazebo::rendering::OrbitViewController, 766
 - gazebo::rendering::TransmitterVisual, 1132
 - gazebo::rendering::UserCamera, 1146
 - gazebo::rendering::ViewController, 1195
 - gazebo::rendering::Visual, 1217
 - gazebo::sensors::Sensor, 916
 - gazebo::sensors::SensorManager, 924
- update
 - gazebo::sensors::ForceTorqueSensor, 453
 - gazebo::sensors::SonarSensor, 1051
- updateCacheCompleteCondition
 - gazebo::common::ModelDatabasePrivate, 696
- updateCacheCondition
 - gazebo::common::ModelDatabasePrivate, 696
- updateCacheThread
 - gazebo::common::ModelDatabasePrivate, 696
- UpdateChildrenTransforms
 - gazebo::common::SkeletonNode, 1043
- UpdateCollision
 - gazebo::physics::DARTPhysics, 360
 - gazebo::physics::PhysicsEngine, 780
 - gazebo::physics::SimbodyPhysics, 996
- updateDirtyPoseFromDARTTransformation

- gazebo::physics::DARTLink, 348
- UpdateFromMsg
 - gazebo::rendering::Light, 595
 - gazebo::rendering::Visual, 1217
- UpdateImpl
 - gazebo::sensors::CameraSensor, 231
 - gazebo::sensors::ContactSensor, 291
 - gazebo::sensors::DepthCameraSensor, 390
 - gazebo::sensors::ForceTorqueSensor, 452
 - gazebo::sensors::GpsSensor, 466
 - gazebo::sensors::GpuRaySensor, 487
 - gazebo::sensors::ImuSensor, 528
 - gazebo::sensors::MultiCameraSensor, 723
 - gazebo::sensors::RaySensor, 848
 - gazebo::sensors::RFIDSensor, 861
 - gazebo::sensors::RFIDTag, 864
 - gazebo::sensors::Sensor, 917
 - gazebo::sensors::SonarSensor, 1050
 - gazebo::sensors::WirelessTransmitter, 1238
- UpdateInfo.hh, 1495
- UpdateMass
 - gazebo::physics::Link, 617
- updateMutex
 - gazebo::common::ModelDatabasePrivate, 696
- UpdateParameters
 - gazebo::physics::Actor, 143
 - gazebo::physics::Base, 180
 - gazebo::physics::Collision, 245
 - gazebo::physics::Entity, 414
 - gazebo::physics::Inertial, 538
 - gazebo::physics::Joint, 566
 - gazebo::physics::Link, 617
 - gazebo::physics::Model, 692
- UpdateParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 1046
- updateParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 932
- UpdateParamsForCompositeMap
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 1046
- updatePeriod
 - gazebo::sensors::Sensor, 918
- UpdatePhysics
 - gazebo::physics::DARTPhysics, 361
 - gazebo::physics::PhysicsEngine, 780
 - gazebo::physics::SimbodyPhysics, 996
- UpdatePoint
 - gazebo::math::RotationSpline, 874
 - gazebo::math::Spline, 1067
- UpdatePublications
 - gazebo::transport::TopicManager, 1129
- UpdateRays
 - Bullet Physics, 80
 - gazebo::physics::MultiRayShape, 730
 - gazebo::physics::SimbodyMultiRayShape, 986
- UpdateShaders
 - gazebo::rendering::RTShaderSystem, 879
- UpdateSize
 - gazebo::rendering::SelectionObj, 903
- UpdateStateSDF
 - gazebo::physics::World, 1251
- UpdateSurface
 - gazebo::physics::Link, 617
- updateVpParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 932
- updatedLinks
 - gazebo::physics::JointControllerPrivate, 573
- upperLimit
 - gazebo::physics::Joint, 567
- useCFMDamping
 - gazebo::physics::Joint, 568
- useRTShader
 - gazebo::rendering::VisualPrivate, 1224
- UserCamera
 - gazebo::rendering::UserCamera, 1140
- UserCamera.hh, 1496
- UserCameraPrivate.hh, 1496
- UserCameraPtr
 - gazebo::rendering, 128
- UtilTypes.hh, 1497
- Utility, 99
 - DIAG_TIMER_LAP, 99
 - DIAG_TIMER_START, 99
 - DIAG_TIMER_STOP, 99
- V_ABOVE
 - gazebo::rendering::MovableText, 711
- V_BELOW
 - gazebo::rendering::MovableText, 711
- VEL
 - gazebo::physics::Joint, 547
- VERTEX
 - gazebo::rendering::RenderEngine, 856
- VISUAL
 - gazebo::physics::Base, 172
- VISUAL_PLUGIN
 - Common, 41
- valHistory
 - gazebo::common::MovingWindowFilterPrivate, 717
- vallter
 - gazebo::common::MovingWindowFilterPrivate, 717
- valWindowSize
 - gazebo::common::MovingWindowFilterPrivate, 717
- Valid
 - gazebo::common::Image, 520

- ValidateIP
 - gazebo::transport::Connection, 270
- value
 - gazebo::common::NumericKeyFrame, 755
- variance
 - Math, 55
- Vec3ToVector3
 - gazebo::physics::SimbodyPhysics, 996
- Vector2d
 - gazebo::math::Vector2d, 1149
- Vector2d.hh, 1498
- Vector2i
 - gazebo::math::Vector2i, 1158
- Vector2i.hh, 1499
- Vector3
 - gazebo::math::Vector3, 1167, 1168
- Vector3.hh, 1500
- Vector3ToVec3
 - gazebo::physics::SimbodyPhysics, 996
- Vector4
 - gazebo::math::Vector4, 1180, 1181
- Vector4.hh, 1500
- vectorLink
 - gazebo::rendering::TransmitterVisualPrivate, 1134
- velPids
 - gazebo::physics::JointControllerPrivate, 574
- velocities
 - gazebo::physics::JointControllerPrivate, 574
- velocityLimit
 - gazebo::physics::Joint, 568
- VertAlign
 - gazebo::rendering::MovableText, 711
- vertElem
 - gazebo::physics::MultiRayShape, 731
 - gazebo::sensors::GpuRaySensor, 488
- vertHalfAngle
 - gazebo::rendering::GpuLaser, 476
- vertRangeCount
 - gazebo::sensors::GpuRaySensor, 488
- vertRayCount
 - gazebo::sensors::GpuRaySensor, 488
- vertSize
 - gazebo::physics::HeightmapShape, 512
- vertexBufferCapacity
 - gazebo::rendering::DynamicRenderable, 404
- vertexIndex
 - gazebo::common::NodeAssignment, 742
- vfov
 - gazebo::rendering::GpuLaser, 476
- Video
 - gazebo::common::Video, 1188
- video
 - gazebo::rendering::VideoVisualPrivate, 1192
- Video.hh, 1502
- VideoVisual
 - gazebo::rendering::VideoVisual, 1190
- VideoVisual.hh, 1503
- VideoVisualPrivate.hh, 1503
- ViewController
 - gazebo::rendering::ViewController, 1194
- viewController
 - gazebo::rendering::UserCameraPrivate, 1147
- ViewController.hh, 1504
- viewport
 - gazebo::rendering::Camera, 224
- visPub
 - gazebo::physics::Entity, 415
- visible
 - gazebo::rendering::VisualPrivate, 1224
- visitRenderables
 - gazebo::rendering::MovableText, 714
- Visual
 - gazebo::rendering::Visual, 1202
- Visual.hh, 1505
- VisualFromSDF
 - Messages, 67
- visualId
 - gazebo::physics::Actor, 145
- visualIdCount
 - gazebo::rendering::VisualPrivate, 1224
- visualMsg
 - gazebo::physics::Entity, 415
- visualName
 - gazebo::physics::Actor, 145
- VisualPlugin
 - gazebo::VisualPlugin, 1218
- VisualPluginPtr
 - gazebo, 103
- VisualPrivate.hh, 1506
- VisualPtr
 - gazebo::rendering, 128
- visuals
 - gazebo::physics::Link, 618
- Visuals_M
 - gazebo::physics::Link, 601
- w
 - gazebo::math::Quaternion, 837
 - gazebo::math::Vector4, 1187
- WORLD_PLUGIN
 - Common, 40
- WaitForConnection
 - gazebo::transport::Publisher, 823
- waitForNamespaces
 - Transport, 98
- wallTime
 - gazebo::physics::State, 1072
- warn

- gazebo::common::Console, 279
- weight
 - gazebo::common::NodeAssignment, 742
- White
 - gazebo::common::Color, 260
- width
 - gazebo::rendering::VideoVisualPrivate, 1192
- windowId
 - gazebo::rendering::Camera, 224
- WindowManager
 - gazebo::rendering::WindowManager, 1225
- WindowManager.hh, 1507
- WindowManagerPtr
 - gazebo::rendering, 128
- WireBox
 - gazebo::rendering::WireBox, 1228
- WireBox.hh, 1508
- WireBoxPrivate.hh, 1509
- WirelessReceiver
 - gazebo::sensors::WirelessReceiver, 1231
- WirelessReceiver.hh, 1509
- WirelessReceiver_V
 - gazebo::sensors, 133
- WirelessReceiverPtr
 - gazebo::sensors, 133
- WirelessTransceiver
 - gazebo::sensors::WirelessTransceiver, 1233
- WirelessTransceiver.hh, 1510
- WirelessTransceiver_V
 - gazebo::sensors, 133
- WirelessTransceiverPtr
 - gazebo::sensors, 133
- WirelessTransmitter
 - gazebo::sensors::WirelessTransmitter, 1237
- WirelessTransmitter.hh, 1511
- WirelessTransmitter_V
 - gazebo::sensors, 133
- WirelessTransmitterPtr
 - gazebo::sensors, 133
- World
 - gazebo::physics::World, 1242
- world
 - gazebo::physics::Base, 180
 - gazebo::physics::Contact, 282
 - gazebo::physics::PhysicsEngine, 781
 - gazebo::physics::SimbodyJoint, 971
 - gazebo::sensors::Sensor, 918
- World.hh, 1512
- worldCreated
 - gazebo::event::Events, 432
- worldName
 - gazebo::common::UpdateInfo, 1137
- WorldPlugin
 - gazebo::WorldPlugin, 1252
- WorldPluginPtr
 - gazebo, 103
- WorldPtr
 - gazebo::physics, 123
- WorldState
 - gazebo::physics::WorldState, 1255
- WorldState.hh, 1513
- worldUpdateBegin
 - gazebo::event::Events, 432
- worldUpdateEnd
 - gazebo::event::Events, 432
- worlds_running
 - Classes for physics and dynamics, 75
- wrench
 - gazebo::physics::Contact, 282
 - gazebo::physics::Joint, 568
- wrenchMsg
 - gazebo::rendering::WrenchVisualPrivate, 1263
- wrenchSub
 - gazebo::rendering::WrenchVisualPrivate, 1263
- WrenchVisual
 - gazebo::rendering::WrenchVisual, 1260
- WrenchVisual.hh, 1515
- WrenchVisualPrivate.hh, 1515
- WrenchVisualPtr
 - gazebo::rendering, 128
- Write
 - gazebo::util::LogRecord, 637
- x
 - gazebo::math::Quaternion, 837
 - gazebo::math::Vector2d, 1155
 - gazebo::math::Vector2i, 1164
 - gazebo::math::Vector3, 1178
 - gazebo::math::Vector4, 1187
- X_POSITION
 - BVHLoader.hh, 1282
- X_ROTATION
 - BVHLoader.hh, 1282
- xAxis
 - gazebo::rendering::AxisVisualPrivate, 166
- xAxisMat
 - gazebo::rendering::SelectionObjPrivate, 907
- xAxisMatOverlay
 - gazebo::rendering::SelectionObjPrivate, 907
- xCB
 - gazebo::physics::SimbodyJoint, 971
- xPA
 - gazebo::physics::SimbodyJoint, 971
- y
 - gazebo::math::Quaternion, 837
 - gazebo::math::Vector2d, 1156
 - gazebo::math::Vector2i, 1164
 - gazebo::math::Vector3, 1178

- gazebo::math::Vector4, 1187
- Y_POSITION
 - BVHLoader.hh, 1282
- Y_ROTATION
 - BVHLoader.hh, 1282
- yAxis
 - gazebo::rendering::AxisVisualPrivate, 166
- yAxisMat
 - gazebo::rendering::SelectionObjPrivate, 907
- yAxisMatOverlay
 - gazebo::rendering::SelectionObjPrivate, 907
- Yellow
 - gazebo::common::Color, 260

- z
 - gazebo::math::Quaternion, 838
 - gazebo::math::Vector3, 1178
 - gazebo::math::Vector4, 1187
- Z_POSITION
 - BVHLoader.hh, 1282
- Z_ROTATION
 - BVHLoader.hh, 1282
- zAxis
 - gazebo::rendering::AxisVisualPrivate, 166
- zAxisMat
 - gazebo::rendering::SelectionObjPrivate, 907
- zAxisMatOverlay
 - gazebo::rendering::SelectionObjPrivate, 907
- ZERO
 - gazebo::math::Matrix3, 653
 - gazebo::math::Matrix4, 660
- Zero
 - gazebo::common::Time, 1120
 - gazebo::math::Angle, 153
 - gazebo::math::Pose, 805
 - gazebo::math::Vector3, 1178