

Gazebo
2.0.0

Generated by Doxygen 1.8.3.1

Tue Oct 8 2013 16:05:35

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	Hierarchical Index	9
5.1	Class Hierarchy	9
6	Class Index	15
6.1	Class List	15
7	File Index	25
7.1	File List	25
8	Module Documentation	31
8.1	Common	31
8.1.1	Detailed Description	35
8.1.2	Macro Definition Documentation	35
8.1.2.1	gzclr_end	35
8.1.2.2	gzclr_start	35
8.1.2.3	gzdbg	35
8.1.2.4	gzerr	35
8.1.2.5	gzlog	35
8.1.2.6	gzmsg	36
8.1.2.7	gzthrow	36
8.1.2.8	gzwarn	36

8.1.3	Enumeration Type Documentation	36
8.1.3.1	PluginType	36
8.1.4	Function Documentation	36
8.1.4.1	NullStream	36
8.1.4.2	add_search_path_suffix	37
8.1.4.3	ColorErr	37
8.1.4.4	ColorMsg	37
8.1.4.5	DownloadDependencies	37
8.1.4.6	find_file	37
8.1.4.7	find_file	38
8.1.4.8	find_file_path	38
8.1.4.9	Fini	38
8.1.4.10	GetDBConfig	38
8.1.4.11	GetModelConfig	38
8.1.4.12	GetModelFile	39
8.1.4.13	GetModelName	39
8.1.4.14	GetModelPath	39
8.1.4.15	GetModels	39
8.1.4.16	GetModels	40
8.1.4.17	GetQuiet	40
8.1.4.18	GetURI	40
8.1.4.19	HasModel	40
8.1.4.20	Init	40
8.1.4.21	IsInitialized	40
8.1.4.22	load	41
8.1.4.23	Log	41
8.1.4.24	SetQuiet	41
8.1.4.25	Start	41
8.1.5	Variable Documentation	41
8.1.5.1	PixelFormatNames	41
8.2	Events	43
8.2.1	Detailed Description	43
8.2.2	Function Documentation	43
8.2.2.1	~EventT	43
8.2.2.2	Connect	43
8.2.2.3	ConnectionCount	44
8.2.2.4	Disconnect	44

8.2.2.5	Disconnect	45
8.3	Math	46
8.3.1	Detailed Description	47
8.3.2	Function Documentation	48
8.3.2.1	clamp	48
8.3.2.2	equal	48
8.3.2.3	isnan	48
8.3.2.4	isnan	48
8.3.2.5	isPowerOfTwo	49
8.3.2.6	max	49
8.3.2.7	mean	49
8.3.2.8	min	49
8.3.2.9	parseFloat	50
8.3.2.10	parseInt	50
8.3.2.11	precision	50
8.3.2.12	variance	51
8.3.3	Variable Documentation	51
8.3.3.1	NAN_D	51
8.3.3.2	NAN_I	51
8.4	Messages	52
8.4.1	Detailed Description	54
8.4.2	Macro Definition Documentation	54
8.4.2.1	GZ_REGISTER_STATIC_MSG	54
8.4.3	Function Documentation	54
8.4.3.1	Convert	54
8.4.3.2	Convert	54
8.4.3.3	Convert	55
8.4.3.4	Convert	55
8.4.3.5	Convert	55
8.4.3.6	Convert	55
8.4.3.7	Convert	56
8.4.3.8	Convert	56
8.4.3.9	Convert	56
8.4.3.10	Convert	56
8.4.3.11	Convert	57
8.4.3.12	Convert	57
8.4.3.13	CreateRequest	57

8.4.3.14	FogFromSDF	57
8.4.3.15	GeometryFromSDF	58
8.4.3.16	GetHeader	58
8.4.3.17	GUIFromSDF	58
8.4.3.18	Init	58
8.4.3.19	LightFromSDF	59
8.4.3.20	MeshFromSDF	59
8.4.3.21	SceneFromSDF	59
8.4.3.22	Set	59
8.4.3.23	Set	60
8.4.3.24	Set	60
8.4.3.25	Set	60
8.4.3.26	Set	60
8.4.3.27	Set	60
8.4.3.28	Set	60
8.4.3.29	Set	61
8.4.3.30	Set	61
8.4.3.31	Stamp	61
8.4.3.32	Stamp	61
8.4.3.33	TrackVisualFromSDF	61
8.4.3.34	VisualFromSDF	62
8.5	Classes for physics and dynamics	63
8.5.1	Detailed Description	66
8.5.2	Macro Definition Documentation	66
8.5.2.1	GZ_REGISTER_PHYSICS_ENGINE	66
8.5.3	Typedef Documentation	67
8.5.3.1	PhysicsFactoryFn	67
8.5.4	Function Documentation	67
8.5.4.1	create_world	67
8.5.4.2	fini	67
8.5.4.3	get_world	67
8.5.4.4	getUniqueld	67
8.5.4.5	init_world	68
8.5.4.6	init_worlds	68
8.5.4.7	load	68
8.5.4.8	load_world	68
8.5.4.9	load_worlds	68

8.5.4.10	pause_world	68
8.5.4.11	pause_worlds	68
8.5.4.12	remove_worlds	69
8.5.4.13	run_world	69
8.5.4.14	run_worlds	69
8.5.4.15	stop_world	69
8.5.4.16	stop_worlds	69
8.5.4.17	worlds_running	69
8.5.5	Variable Documentation	69
8.5.5.1	EntityTypename	70
8.6	Simbody Physics	71
8.6.1	Detailed Description	72
8.7	Rendering	73
8.7.1	Detailed Description	76
8.7.2	Enumeration Type Documentation	76
8.7.2.1	SelectionMode	76
8.7.3	Function Documentation	76
8.7.3.1	SelectionObj	76
8.7.3.2	~SelectionObj	76
8.7.3.3	Attach	77
8.7.3.4	create_scene	77
8.7.3.5	Detach	77
8.7.3.6	fini	77
8.7.3.7	get_scene	77
8.7.3.8	GetMode	77
8.7.3.9	GetState	77
8.7.3.10	init	77
8.7.3.11	load	78
8.7.3.12	Load	78
8.7.3.13	remove_scene	78
8.7.3.14	SetGlobal	78
8.7.3.15	SetMode	78
8.7.3.16	SetMode	78
8.7.3.17	SetState	78
8.7.3.18	SetState	79
8.7.3.19	UpdateSize	79
8.8	Sensors	80

8.8.1	Detailed Description	81
8.8.2	Macro Definition Documentation	82
8.8.2.1	GZ_REGISTER_STATIC_SENSOR	82
8.8.3	Function Documentation	82
8.8.3.1	create_sensor	82
8.8.3.2	create_sensor	82
8.8.3.3	fini	82
8.8.3.4	get_sensor	83
8.8.3.5	init	83
8.8.3.6	load	83
8.8.3.7	remove_sensor	83
8.8.3.8	remove_sensors	83
8.8.3.9	run_once	84
8.8.3.10	run_threads	84
8.8.3.11	stop	84
8.9	Transport	85
8.9.1	Detailed Description	87
8.9.2	Typedef Documentation	87
8.9.2.1	CallbackHelperPtr	87
8.9.3	Function Documentation	87
8.9.3.1	clear_buffers	87
8.9.3.2	fini	87
8.9.3.3	get_master_uri	87
8.9.3.4	get_topic_namespaces	87
8.9.3.5	getAdvertisedTopics	88
8.9.3.6	getAdvertisedTopics	88
8.9.3.7	getMinimalComms	88
8.9.3.8	getTopicMsgType	88
8.9.3.9	init	88
8.9.3.10	is_stopped	89
8.9.3.11	pause_incoming	89
8.9.3.12	publish	89
8.9.3.13	request	89
8.9.3.14	requestNoReply	90
8.9.3.15	requestNoReply	90
8.9.3.16	run	90
8.9.3.17	setMinimalComms	90

8.9.3.18	stop	90
8.10	Utility	91
8.10.1	Detailed Description	91
8.10.2	Macro Definition Documentation	91
8.10.2.1	DIAG_TIMER_LAP	91
8.10.2.2	DIAG_TIMER_START	91
8.10.2.3	DIAG_TIMER_STOP	91
9	Namespace Documentation	93
9.1	boost Namespace Reference	93
9.2	gazebo Namespace Reference	93
9.2.1	Detailed Description	94
9.2.2	Typedef Documentation	95
9.2.2.1	GUIPluginPtr	95
9.2.2.2	ModelPluginPtr	95
9.2.2.3	SensorPluginPtr	95
9.2.2.4	SystemPluginPtr	95
9.2.2.5	VisualPluginPtr	95
9.2.2.6	WorldPluginPtr	95
9.2.3	Function Documentation	95
9.2.3.1	add_plugin	95
9.2.3.2	find_file	95
9.2.3.3	fini	95
9.2.3.4	init	95
9.2.3.5	load	95
9.2.3.6	print_version	95
9.2.3.7	run	95
9.2.3.8	stop	95
9.3	gazebo::common Namespace Reference	95
9.3.1	Detailed Description	98
9.3.2	Typedef Documentation	98
9.3.2.1	AnimationPtr	98
9.3.2.2	DiagnosticTimerPtr	98
9.3.2.3	NodeMap	98
9.3.2.4	NodeMapIter	98
9.3.2.5	NumericAnimationPtr	98
9.3.2.6	Param_V	98

9.3.2.7	PoseAnimationPtr	98
9.3.2.8	RawNodeAnim	98
9.3.2.9	RawNodeWeights	98
9.3.2.10	RawSkeletonAnim	98
9.3.2.11	SphericalCoordinatesPtr	98
9.3.2.12	StrStr_M	99
9.3.3	Variable Documentation	99
9.3.3.1	SpeedOfLight	99
9.4	gazebo::event Namespace Reference	99
9.4.1	Detailed Description	99
9.4.2	Typedef Documentation	99
9.4.2.1	Connection_V	99
9.4.2.2	ConnectionPtr	99
9.5	gazebo::math Namespace Reference	99
9.5.1	Detailed Description	102
9.5.2	Typedef Documentation	102
9.5.2.1	GeneratorType	102
9.5.2.2	NormalRealDist	102
9.5.2.3	NRealGen	102
9.5.2.4	UIntGen	102
9.5.2.5	UniformIntDist	102
9.5.2.6	UniformRealDist	102
9.5.2.7	URealGen	102
9.6	gazebo::msgs Namespace Reference	102
9.6.1	Detailed Description	104
9.6.2	Typedef Documentation	104
9.6.2.1	MsgFactoryFn	104
9.7	gazebo::physics Namespace Reference	104
9.7.1	Detailed Description	109
9.7.2	Typedef Documentation	109
9.7.2.1	Actor_V	109
9.7.2.2	ActorPtr	109
9.7.2.3	Base_V	109
9.7.2.4	BasePtr	109
9.7.2.5	BoxShapePtr	109
9.7.2.6	Collision_V	109
9.7.2.7	CollisionPtr	109

9.7.2.8	ContactPtr	109
9.7.2.9	CylinderShapePtr	109
9.7.2.10	EntityPtr	109
9.7.2.11	GripperPtr	109
9.7.2.12	HeightmapShapePtr	109
9.7.2.13	InertialPtr	109
9.7.2.14	Joint_V	110
9.7.2.15	JointController_V	110
9.7.2.16	JointControllerPtr	110
9.7.2.17	JointPtr	110
9.7.2.18	JointState_M	110
9.7.2.19	Link_V	110
9.7.2.20	LinkPtr	110
9.7.2.21	LinkState_M	110
9.7.2.22	MeshShapePtr	110
9.7.2.23	Model_V	110
9.7.2.24	ModelPtr	110
9.7.2.25	ModelState_M	110
9.7.2.26	MultiRayShapePtr	110
9.7.2.27	PhysicsEnginePtr	110
9.7.2.28	RayShapePtr	110
9.7.2.29	RoadPtr	110
9.7.2.30	ShapePtr	110
9.7.2.31	SimbodyCollisionPtr	110
9.7.2.32	SimbodyLinkPtr	110
9.7.2.33	SimbodyModelPtr	110
9.7.2.34	SimbodyPhysicsPtr	110
9.7.2.35	SimbodyRayShapePtr	110
9.7.2.36	SphereShapePtr	110
9.7.2.37	SurfaceParamsPtr	110
9.7.2.38	WorldPtr	110
9.8	gazebo::rendering Namespace Reference	111
9.8.1	Detailed Description	114
9.8.2	Typedef Documentation	114
9.8.2.1	ArrowVisualPtr	114
9.8.2.2	AxisVisualPtr	114
9.8.2.3	CameraPtr	114

9.8.2.4	CameraVisualPtr	114
9.8.2.5	COMVisualPtr	114
9.8.2.6	ContactVisualPtr	114
9.8.2.7	DepthCameraPtr	114
9.8.2.8	DynamicLinesPtr	114
9.8.2.9	GpuLaserPtr	114
9.8.2.10	JointVisualPtr	114
9.8.2.11	LaserVisualPtr	114
9.8.2.12	LightPtr	114
9.8.2.13	RFIDTagVisualPtr	114
9.8.2.14	RFIDVisualPtr	114
9.8.2.15	ScenePtr	114
9.8.2.16	SelectionObjPtr	114
9.8.2.17	SonarVisualPtr	114
9.8.2.18	UserCameraPtr	114
9.8.2.19	VisualPtr	114
9.8.2.20	WindowManagerPtr	115
9.8.2.21	WrenchVisualPtr	115
9.8.3	Enumeration Type Documentation	115
9.8.3.1	RenderOpType	115
9.9	gazebo::sensors Namespace Reference	115
9.9.1	Detailed Description	118
9.9.2	Typedef Documentation	118
9.9.2.1	CameraSensor_V	118
9.9.2.2	CameraSensorPtr	118
9.9.2.3	ContactSensor_V	118
9.9.2.4	ContactSensorPtr	118
9.9.2.5	DepthCameraSensor_V	118
9.9.2.6	DepthCameraSensorPtr	118
9.9.2.7	ForceTorqueSensorPtr	118
9.9.2.8	GpsSensorPtr	118
9.9.2.9	GpuRaySensor_V	118
9.9.2.10	GpuRaySensorPtr	118
9.9.2.11	ImuSensor_V	118
9.9.2.12	ImuSensorPtr	118
9.9.2.13	NoisePtr	118
9.9.2.14	RaySensor_V	118

9.9.2.15	RaySensorPtr	118
9.9.2.16	RFIDSensor_V	118
9.9.2.17	RFIDSensorPtr	118
9.9.2.18	RFIDTag_V	118
9.9.2.19	RFIDTagPtr	118
9.9.2.20	Sensor_V	118
9.9.2.21	SensorFactoryFn	118
9.9.2.22	SensorPtr	118
9.9.2.23	SonarSensorPtr	118
9.9.2.24	WirelessReceiver_V	119
9.9.2.25	WirelessReceiverPtr	119
9.9.2.26	WirelessTransceiver_V	119
9.9.2.27	WirelessTransceiverPtr	119
9.9.2.28	WirelessTransmitter_V	119
9.9.2.29	WirelessTransmitterPtr	119
9.9.3	Enumeration Type Documentation	119
9.9.3.1	SensorCategory	119
9.10	gazebo::transport Namespace Reference	119
9.10.1	Typedef Documentation	121
9.10.1.1	ConnectionPtr	121
9.10.1.2	MessagePtr	121
9.10.1.3	NodePtr	121
9.10.1.4	PublicationPtr	121
9.10.1.5	PublicationTransportPtr	121
9.10.1.6	PublisherPtr	121
9.10.1.7	SubscriberPtr	121
9.10.1.8	SubscriptionTransportPtr	121
9.11	gazebo::util Namespace Reference	121
9.11.1	Typedef Documentation	122
9.11.1.1	DiagnosticTimerPtr	122
9.11.1.2	OpenALSinkPtr	122
9.11.1.3	OpenALSinkSourcePtr	122
9.12	google Namespace Reference	122
9.13	google::protobuf Namespace Reference	122
9.14	google::protobuf::compiler Namespace Reference	122
9.15	google::protobuf::compiler::cpp Namespace Reference	123
9.16	Ogre Namespace Reference	123

9.17	ogre Namespace Reference	123
9.18	SimTK Namespace Reference	123
9.19	SkyX Namespace Reference	123
10	Class Documentation	125
10.1	gazebo::physics::Actor Class Reference	125
10.1.1	Detailed Description	127
10.1.2	Constructor & Destructor Documentation	127
10.1.2.1	Actor	127
10.1.2.2	~Actor	128
10.1.3	Member Function Documentation	128
10.1.3.1	Fini	128
10.1.3.2	GetSDF	128
10.1.3.3	Init	128
10.1.3.4	IsActive	128
10.1.3.5	Load	128
10.1.3.6	Play	128
10.1.3.7	Stop	129
10.1.3.8	Update	129
10.1.3.9	UpdateParameters	129
10.1.4	Member Data Documentation	129
10.1.4.1	active	129
10.1.4.2	autoStart	129
10.1.4.3	bonePosePub	129
10.1.4.4	interpolateX	129
10.1.4.5	lastPos	129
10.1.4.6	lastScriptTime	129
10.1.4.7	lastTraj	130
10.1.4.8	loop	130
10.1.4.9	mainLink	130
10.1.4.10	mesh	130
10.1.4.11	oldAction	130
10.1.4.12	pathLength	130
10.1.4.13	playStartTime	130
10.1.4.14	prevFrameTime	130
10.1.4.15	scriptLength	130
10.1.4.16	skelAnimation	130

10.1.4.17 skeleton	130
10.1.4.18 skelNodesMap	131
10.1.4.19 skinFile	131
10.1.4.20 skinScale	131
10.1.4.21 startDelay	131
10.1.4.22 trajectories	131
10.1.4.23 trajInfo	131
10.1.4.24 visualId	131
10.1.4.25 visualName	131
10.2 gazebo::math::Angle Class Reference	131
10.2.1 Detailed Description	133
10.2.2 Constructor & Destructor Documentation	133
10.2.2.1 Angle	133
10.2.2.2 Angle	133
10.2.2.3 Angle	133
10.2.2.4 ~Angle	133
10.2.3 Member Function Documentation	134
10.2.3.1 Degree	134
10.2.3.2 Normalize	134
10.2.3.3 operator!=	134
10.2.3.4 operator*	134
10.2.3.5 operator*	134
10.2.3.6 operator*=	134
10.2.3.7 operator+	135
10.2.3.8 operator+=	135
10.2.3.9 operator-	135
10.2.3.10 operator-=	135
10.2.3.11 operator/	136
10.2.3.12 operator/=	136
10.2.3.13 operator<	136
10.2.3.14 operator<=	136
10.2.3.15 operator==	137
10.2.3.16 operator>	137
10.2.3.17 operator>=	137
10.2.3.18 Radian	137
10.2.3.19 SetFromDegree	138
10.2.3.20 SetFromRadian	138

10.2.4	Friends And Related Function Documentation	138
10.2.4.1	operator<<	138
10.2.4.2	operator>>	138
10.2.5	Member Data Documentation	139
10.2.5.1	HalfPi	139
10.2.5.2	Pi	139
10.2.5.3	TwoPi	139
10.2.5.4	Zero	139
10.3	gazebo::common::Animation Class Reference	139
10.3.1	Detailed Description	140
10.3.2	Member Typedef Documentation	141
10.3.2.1	KeyFrame_V	141
10.3.3	Constructor & Destructor Documentation	141
10.3.3.1	Animation	141
10.3.3.2	~Animation	141
10.3.4	Member Function Documentation	141
10.3.4.1	AddTime	141
10.3.4.2	GetKeyFrame	141
10.3.4.3	GetKeyFrameCount	141
10.3.4.4	GetKeyFramesAtTime	142
10.3.4.5	GetLength	142
10.3.4.6	GetTime	142
10.3.4.7	SetLength	142
10.3.4.8	SetTime	142
10.3.5	Member Data Documentation	143
10.3.5.1	build	143
10.3.5.2	keyFrames	143
10.3.5.3	length	143
10.3.5.4	loop	143
10.3.5.5	name	143
10.3.5.6	timePos	143
10.4	gazebo::rendering::ArrowVisual Class Reference	143
10.4.1	Detailed Description	144
10.4.2	Constructor & Destructor Documentation	144
10.4.2.1	ArrowVisual	144
10.4.2.2	~ArrowVisual	145
10.4.3	Member Function Documentation	145

10.4.3.1	Load	145
10.4.3.2	ShowRotation	145
10.5	gazebo::common::AssertionInternalError Class Reference	145
10.5.1	Detailed Description	146
10.5.2	Constructor & Destructor Documentation	146
10.5.2.1	AssertionInternalError	146
10.5.2.2	~AssertionInternalError	147
10.6	gazebo::common::AudioDecoder Class Reference	147
10.6.1	Detailed Description	147
10.6.2	Constructor & Destructor Documentation	147
10.6.2.1	AudioDecoder	147
10.6.2.2	~AudioDecoder	147
10.6.3	Member Function Documentation	148
10.6.3.1	Decode	148
10.6.3.2	GetFile	148
10.6.3.3	GetSampleRate	148
10.6.3.4	SetFile	148
10.7	gazebo::rendering::AxisVisual Class Reference	149
10.7.1	Detailed Description	150
10.7.2	Constructor & Destructor Documentation	150
10.7.2.1	AxisVisual	150
10.7.2.2	~AxisVisual	150
10.7.3	Member Function Documentation	150
10.7.3.1	Load	150
10.7.3.2	ScaleXAxis	150
10.7.3.3	ScaleYAxis	150
10.7.3.4	ScaleZAxis	151
10.7.3.5	SetAxisMaterial	151
10.7.3.6	ShowRotation	151
10.8	gazebo::physics::BallJoint< T > Class Template Reference	151
10.8.1	Detailed Description	152
10.8.2	Constructor & Destructor Documentation	152
10.8.2.1	BallJoint	152
10.8.2.2	~BallJoint	152
10.8.3	Member Function Documentation	152
10.8.3.1	GetAngleCount	152
10.8.3.2	GetHighStop	152

10.8.3.3	GetLowStop	152
10.8.3.4	Load	152
10.8.3.5	SetAxis	153
10.8.3.6	SetHighStop	153
10.8.3.7	SetLowStop	153
10.9	gazebo::physics::Base Class Reference	153
10.9.1	Detailed Description	156
10.9.2	Member Enumeration Documentation	156
10.9.2.1	EntityType	156
10.9.3	Constructor & Destructor Documentation	157
10.9.3.1	Base	157
10.9.3.2	~Base	157
10.9.4	Member Function Documentation	157
10.9.4.1	AddChild	157
10.9.4.2	AddType	157
10.9.4.3	ComputeScopedName	157
10.9.4.4	Fini	158
10.9.4.5	GetByName	158
10.9.4.6	GetChild	158
10.9.4.7	GetChild	158
10.9.4.8	GetChildCount	158
10.9.4.9	GetId	159
10.9.4.10	GetName	159
10.9.4.11	GetParent	159
10.9.4.12	GetParentId	159
10.9.4.13	GetSaveable	159
10.9.4.14	GetScopedName	159
10.9.4.15	GetSDF	160
10.9.4.16	GetType	160
10.9.4.17	GetWorld	160
10.9.4.18	HasType	160
10.9.4.19	Init	160
10.9.4.20	IsSelected	161
10.9.4.21	Load	161
10.9.4.22	operator==	161
10.9.4.23	Print	161
10.9.4.24	RemoveChild	162

10.9.4.25 RemoveChild	162
10.9.4.26 RemoveChildren	162
10.9.4.27 Reset	162
10.9.4.28 Reset	162
10.9.4.29 SetName	162
10.9.4.30 SetParent	163
10.9.4.31 SetSaveable	163
10.9.4.32 SetSelected	163
10.9.4.33 SetWorld	163
10.9.4.34 Update	163
10.9.4.35 UpdateParameters	163
10.9.5 Member Data Documentation	164
10.9.5.1 children	164
10.9.5.2 childrenEnd	164
10.9.5.3 parent	164
10.9.5.4 sdf	164
10.9.5.5 world	164
10.10 gazebo::math::Box Class Reference	164
10.10.1 Detailed Description	165
10.10.2 Constructor & Destructor Documentation	165
10.10.2.1 Box	165
10.10.2.2 Box	166
10.10.2.3 Box	166
10.10.2.4 ~Box	166
10.10.3 Member Function Documentation	166
10.10.3.1 GetCenter	166
10.10.3.2 GetSize	166
10.10.3.3 GetXLength	166
10.10.3.4 GetYLength	167
10.10.3.5 GetZLength	167
10.10.3.6 Merge	167
10.10.3.7 operator+	167
10.10.3.8 operator+=	167
10.10.3.9 operator-	168
10.10.3.10 operator=	168
10.10.3.11 operator==	168
10.10.4 Friends And Related Function Documentation	168

10.10.4.1 operator<<	168
10.10.5 Member Data Documentation	169
10.10.5.1 max	169
10.10.5.2 min	169
10.11 gazebo::physics::BoxShape Class Reference	169
10.11.1 Detailed Description	171
10.11.2 Constructor & Destructor Documentation	171
10.11.2.1 BoxShape	171
10.11.2.2 ~BoxShape	171
10.11.3 Member Function Documentation	171
10.11.3.1 FillMsg	171
10.11.3.2 GetSize	171
10.11.3.3 Init	171
10.11.3.4 ProcessMsg	171
10.11.3.5 SetScale	172
10.11.3.6 SetSize	172
10.12 gazebo::common::BVHLoader Class Reference	172
10.12.1 Detailed Description	172
10.12.2 Constructor & Destructor Documentation	173
10.12.2.1 BVHLoader	173
10.12.2.2 ~BVHLoader	173
10.12.3 Member Function Documentation	173
10.12.3.1 Load	173
10.13 gazebo::transport::CallbackHelper Class Reference	173
10.13.1 Detailed Description	174
10.13.2 Constructor & Destructor Documentation	175
10.13.2.1 CallbackHelper	175
10.13.2.2 ~CallbackHelper	175
10.13.3 Member Function Documentation	175
10.13.3.1 GetId	175
10.13.3.2 GetLatching	175
10.13.3.3 GetMsgType	175
10.13.3.4 HandleData	175
10.13.3.5 HandleMessage	176
10.13.3.6 IsLocal	176
10.13.4 Member Data Documentation	176
10.13.4.1 latching	176

10.14gazebo::transport::CallbackHelperT< M > Class Template Reference	177
10.14.1 Detailed Description	177
10.14.2 Constructor & Destructor Documentation	177
10.14.2.1 CallbackHelperT	178
10.14.3 Member Function Documentation	178
10.14.3.1 GetMsgType	178
10.14.3.2 HandleData	178
10.14.3.3 HandleMessage	178
10.14.3.4 IsLocal	179
10.15gazebo::rendering::Camera Class Reference	179
10.15.1 Detailed Description	185
10.15.2 Constructor & Destructor Documentation	185
10.15.2.1 Camera	185
10.15.2.2 ~Camera	186
10.15.3 Member Function Documentation	186
10.15.3.1 AnimationComplete	186
10.15.3.2 AttachToVisual	186
10.15.3.3 AttachToVisual	186
10.15.3.4 AttachToVisualImpl	186
10.15.3.5 AttachToVisualImpl	187
10.15.3.6 AttachToVisualImpl	187
10.15.3.7 ConnectNewImageFrame	187
10.15.3.8 CreateRenderTexture	188
10.15.3.9 DisconnectNewImageFrame	188
10.15.3.10EnableSaveFrame	188
10.15.3.11Fini	188
10.15.3.12GetAspectRatio	188
10.15.3.13GetAvgFPS	189
10.15.3.14GetCameraToViewportRay	189
10.15.3.15GetCaptureData	189
10.15.3.16GetDirection	189
10.15.3.17GetFarClip	189
10.15.3.18GetFrameFilename	189
10.15.3.19GetHFOV	190
10.15.3.20GetImageByteSize	190
10.15.3.21GetImageByteSize	190
10.15.3.22GetImageData	190

10.15.3.23	GetImageDepth	190
10.15.3.24	GetImageFormat	191
10.15.3.25	GetImageHeight	191
10.15.3.26	GetImageWidth	191
10.15.3.27	GetInitialized	191
10.15.3.28	GetLastRenderWallTime	191
10.15.3.29	GetName	192
10.15.3.30	GetNearClip	192
10.15.3.31	GetOgreCamera	192
10.15.3.32	GetPitchNode	192
10.15.3.33	GetRenderRate	192
10.15.3.34	GetRenderTexture	192
10.15.3.35	GetRight	193
10.15.3.36	GetScene	193
10.15.3.37	GetSceneNode	193
10.15.3.38	GetScreenshotPath	193
10.15.3.39	GetTextureHeight	193
10.15.3.40	GetTextureWidth	193
10.15.3.41	GetTriangleCount	194
10.15.3.42	GetUp	194
10.15.3.43	GetVFOV	194
10.15.3.44	GetViewport	194
10.15.3.45	GetViewportHeight	194
10.15.3.46	GetViewportWidth	194
10.15.3.47	GetWindowId	195
10.15.3.48	GetWorldPointOnPlane	195
10.15.3.49	GetWorldPose	195
10.15.3.50	GetWorldPosition	195
10.15.3.51	GetWorldRotation	195
10.15.3.52	GetZValue	196
10.15.3.53	Init	196
10.15.3.54	IsAnimating	196
10.15.3.55	IsVisible	196
10.15.3.56	IsVisible	196
10.15.3.57	Load	197
10.15.3.58	Load	197
10.15.3.59	MoveToPosition	197

10.15.3.60	MoveToPositions	197
10.15.3.61	PostRender	197
10.15.3.62	ReadPixelBuffer	198
10.15.3.63	Render	198
10.15.3.64	RenderImpl	198
10.15.3.65	RotatePitch	198
10.15.3.66	RotateYaw	198
10.15.3.67	SaveFrame	198
10.15.3.68	SaveFrame	199
10.15.3.69	SetAspectRatio	199
10.15.3.70	SetCaptureData	199
10.15.3.71	SetCaptureDataOnce	199
10.15.3.72	SetClipDist	199
10.15.3.73	SetHFOV	200
10.15.3.74	SetImageHeight	200
10.15.3.75	SetImageSize	200
10.15.3.76	SetImageWidth	200
10.15.3.77	SetName	200
10.15.3.78	SetRenderRate	201
10.15.3.79	SetRenderTarget	201
10.15.3.80	SetSaveFramePathname	201
10.15.3.81	SetScene	201
10.15.3.82	SetSceneNode	201
10.15.3.83	SetWindowId	201
10.15.3.84	SetWorldPose	201
10.15.3.85	SetWorldPosition	202
10.15.3.86	SetWorldRotation	202
10.15.3.87	ShowWireframe	202
10.15.3.88	ToggleShowWireframe	202
10.15.3.89	TrackVisual	202
10.15.3.90	TrackVisualImpl	202
10.15.3.91	TrackVisualImpl	203
10.15.3.92	Translate	203
10.15.3.93	Update	203
10.15.4	Member Data Documentation	203
10.15.4.1	animState	203
10.15.4.2	bayerFrameBuffer	203

10.15.4.3 camera	203
10.15.4.4 captureData	203
10.15.4.5 captureDataOnce	204
10.15.4.6 connections	204
10.15.4.7 imageFormat	204
10.15.4.8 imageHeight	204
10.15.4.9 imageWidth	204
10.15.4.10initialized	204
10.15.4.11lastRenderWallTime	204
10.15.4.12name	204
10.15.4.13newData	204
10.15.4.14newImageFrame	204
10.15.4.15onAnimationComplete	204
10.15.4.16pitchNode	205
10.15.4.17prevAnimTime	205
10.15.4.18renderTarget	205
10.15.4.19renderTexture	205
10.15.4.20requests	205
10.15.4.21saveCount	205
10.15.4.22saveFrameBuffer	205
10.15.4.23scene	205
10.15.4.24sceneNode	205
10.15.4.25screenshotPath	205
10.15.4.26sdf	205
10.15.4.27textureHeight	205
10.15.4.28textureWidth	206
10.15.4.29viewport	206
10.15.4.30windowId	206
10.16gazebo::sensors::CameraSensor Class Reference	206
10.16.1 Detailed Description	207
10.16.2 Constructor & Destructor Documentation	207
10.16.2.1 CameraSensor	207
10.16.2.2 ~CameraSensor	207
10.16.3 Member Function Documentation	208
10.16.3.1 Fini	208
10.16.3.2 GetCamera	208
10.16.3.3 GetImageData	208

10.16.3.4	GetImageHeight	208
10.16.3.5	GetImageWidth	208
10.16.3.6	GetTopic	208
10.16.3.7	Init	209
10.16.3.8	IsActive	209
10.16.3.9	Load	209
10.16.3.10	Load	209
10.16.3.11	SaveFrame	209
10.16.3.12	UpdateImpl	210
10.17	gazebo::rendering::CameraVisual Class Reference	210
10.17.1	Detailed Description	211
10.17.2	Constructor & Destructor Documentation	211
10.17.2.1	CameraVisual	211
10.17.2.2	~CameraVisual	211
10.17.3	Member Function Documentation	211
10.17.3.1	Load	211
10.18	gazebo::common::ColladaLoader Class Reference	211
10.18.1	Detailed Description	212
10.18.2	Constructor & Destructor Documentation	212
10.18.2.1	ColladaLoader	212
10.18.2.2	~ColladaLoader	212
10.18.3	Member Function Documentation	212
10.18.3.1	Load	212
10.19	gazebo::physics::Collision Class Reference	213
10.19.1	Detailed Description	215
10.19.2	Constructor & Destructor Documentation	216
10.19.2.1	Collision	216
10.19.2.2	~Collision	216
10.19.3	Member Function Documentation	216
10.19.3.1	AddContact	216
10.19.3.2	FillMsg	216
10.19.3.3	Fini	216
10.19.3.4	GetBoundingBox	216
10.19.3.5	GetContactsEnabled	217
10.19.3.6	GetLaserRetro	217
10.19.3.7	GetLink	217
10.19.3.8	GetMaxContacts	217

10.19.3.9	GetModel	217
10.19.3.10	GetRelativeAngularAccel	217
10.19.3.11	GetRelativeAngularVel	218
10.19.3.12	GetRelativeLinearAccel	218
10.19.3.13	GetRelativeLinearVel	218
10.19.3.14	GetShape	218
10.19.3.15	GetShapeType	218
10.19.3.16	GetState	219
10.19.3.17	GetSurface	219
10.19.3.18	GetWorldAngularAccel	219
10.19.3.19	GetWorldAngularVel	219
10.19.3.20	GetWorldLinearAccel	219
10.19.3.21	GetWorldLinearVel	219
10.19.3.22	Init	220
10.19.3.23	IsPlaceable	220
10.19.3.24	Load	220
10.19.3.25	ProcessMsg	220
10.19.3.26	SetCategoryBits	220
10.19.3.27	SetCollideBits	220
10.19.3.28	SetCollision	221
10.19.3.29	SetContactsEnabled	221
10.19.3.30	SetLaserRetro	221
10.19.3.31	SetMaxContacts	221
10.19.3.32	SetScale	221
10.19.3.33	SetShape	222
10.19.3.34	SetState	222
10.19.3.35	UpdateParameters	222
10.19.4	Member Data Documentation	222
10.19.4.1	link	222
10.19.4.2	placeable	222
10.19.4.3	shape	222
10.20	gazebo::physics::CollisionState Class Reference	222
10.20.1	Detailed Description	224
10.20.2	Constructor & Destructor Documentation	224
10.20.2.1	CollisionState	224
10.20.2.2	CollisionState	224
10.20.2.3	CollisionState	224

10.20.2.4	~CollisionState	224
10.20.3	Member Function Documentation	224
10.20.3.1	FillSDF	224
10.20.3.2	GetPose	225
10.20.3.3	IsZero	225
10.20.3.4	Load	225
10.20.3.5	operator+	225
10.20.3.6	operator-	225
10.20.3.7	operator=	226
10.20.4	Friends And Related Function Documentation	226
10.20.4.1	operator<<	226
10.21	gazebo::common::Color Class Reference	226
10.21.1	Detailed Description	229
10.21.2	Member Typedef Documentation	229
10.21.2.1	ABGR	229
10.21.2.2	ARGB	229
10.21.2.3	BGRA	229
10.21.2.4	RGBA	229
10.21.3	Constructor & Destructor Documentation	229
10.21.3.1	Color	229
10.21.3.2	Color	229
10.21.3.3	Color	229
10.21.3.4	~Color	230
10.21.4	Member Function Documentation	230
10.21.4.1	GetAsABGR	230
10.21.4.2	GetAsARGB	230
10.21.4.3	GetAsBGRA	230
10.21.4.4	GetAsHSV	230
10.21.4.5	GetAsRGBA	230
10.21.4.6	GetAsYUV	231
10.21.4.7	operator!=	231
10.21.4.8	operator*	231
10.21.4.9	operator*	231
10.21.4.10	operator*= 10	231
10.21.4.11	operator+	232
10.21.4.12	operator+	232
10.21.4.13	operator+=	232

10.21.4.14operator-	232
10.21.4.15operator-	233
10.21.4.16operator==	233
10.21.4.17operator/	233
10.21.4.18operator/	233
10.21.4.19operator/=	234
10.21.4.20operator=	234
10.21.4.21operator==	234
10.21.4.22operator[]	234
10.21.4.23Reset	235
10.21.4.24Set	235
10.21.4.25SetFromABGR	235
10.21.4.26SetFromARGB	235
10.21.4.27SetFromBGRA	235
10.21.4.28SetFromHSV	236
10.21.4.29SetFromRGBA	236
10.21.4.30SetFromYUV	236
10.21.5 Friends And Related Function Documentation	236
10.21.5.1 operator<<	236
10.21.5.2 operator>>	236
10.21.6 Member Data Documentation	237
10.21.6.1 a	237
10.21.6.2 b	237
10.21.6.3 Black	237
10.21.6.4 Blue	237
10.21.6.5 g	237
10.21.6.6 Green	237
10.21.6.7 Purple	237
10.21.6.8 r	237
10.21.6.9 Red	237
10.21.6.10White	237
10.21.6.11Yellow	237
10.22gazebo::rendering::COMVisual Class Reference	237
10.22.1 Detailed Description	238
10.22.2 Constructor & Destructor Documentation	238
10.22.2.1 COMVisual	238
10.22.2.2 ~COMVisual	239

10.22.3 Member Function Documentation	239
10.22.3.1 Load	239
10.22.3.2 Load	239
10.23 gazebo::event::Connection Class Reference	239
10.23.1 Detailed Description	240
10.23.2 Constructor & Destructor Documentation	240
10.23.2.1 Connection	240
10.23.2.2 Connection	240
10.23.2.3 ~Connection	240
10.23.3 Member Function Documentation	240
10.23.3.1 GetId	240
10.24 gazebo::transport::Connection Class Reference	240
10.24.1 Detailed Description	242
10.24.2 Member Typedef Documentation	243
10.24.2.1 AcceptCallback	243
10.24.2.2 ReadCallback	243
10.24.3 Constructor & Destructor Documentation	243
10.24.3.1 Connection	243
10.24.3.2 ~Connection	243
10.24.4 Member Function Documentation	243
10.24.4.1 AsyncRead	243
10.24.4.2 Cancel	243
10.24.4.3 Connect	243
10.24.4.4 ConnectToShutdown	243
10.24.4.5 DisconnectShutdown	244
10.24.4.6 EnqueueMsg	244
10.24.4.7 EnqueueMsg	244
10.24.4.8 GetId	244
10.24.4.9 GetIPWhiteList	245
10.24.4.10 GetLocalAddress	245
10.24.4.11 GetLocalHostname	245
10.24.4.12 GetLocalPort	245
10.24.4.13 GetLocalURI	245
10.24.4.14 GetRemoteAddress	245
10.24.4.15 GetRemoteHostname	246
10.24.4.16 GetRemotePort	246
10.24.4.17 GetRemoteURI	246

10.24.4.18	IsOpen	246
10.24.4.19	Listen	246
10.24.4.20	ProcessWriteQueue	246
10.24.4.21	Read	247
10.24.4.22	Shutdown	247
10.24.4.23	StartRead	247
10.24.4.24	StopRead	247
10.24.4.25	ValidateIP	247
10.25	gazebo::transport::ConnectionManager Class Reference	247
10.25.1	Detailed Description	249
10.25.2	Member Function Documentation	249
10.25.2.1	Advertise	249
10.25.2.2	ConnectToRemoteHost	249
10.25.2.3	Fini	249
10.25.2.4	GetAllPublishers	249
10.25.2.5	GetTopicNamespaces	250
10.25.2.6	Init	250
10.25.2.7	IsRunning	250
10.25.2.8	RegisterTopicNamespace	250
10.25.2.9	RemoveConnection	250
10.25.2.10	Run	251
10.25.2.11	Stop	251
10.25.2.12	Subscribe	251
10.25.2.13	TriggerUpdate	251
10.25.2.14	Unadvertise	251
10.25.2.15	Unsubscribe	251
10.25.2.16	Unsubscribe	251
10.25.3	Member Data Documentation	252
10.25.3.1	eventConnections	252
10.26	gazebo::common::Console Class Reference	252
10.26.1	Detailed Description	253
10.27	gazebo::physics::Contact Class Reference	253
10.27.1	Detailed Description	254
10.27.2	Constructor & Destructor Documentation	254
10.27.2.1	Contact	254
10.27.2.2	Contact	254
10.27.2.3	~Contact	254

10.27.3 Member Function Documentation	254
10.27.3.1 DebugString	254
10.27.3.2 FillMsg	254
10.27.3.3 operator=	255
10.27.3.4 operator=	255
10.27.3.5 Reset	255
10.27.4 Member Data Documentation	255
10.27.4.1 collision1	255
10.27.4.2 collision2	255
10.27.4.3 count	255
10.27.4.4 depths	256
10.27.4.5 normals	256
10.27.4.6 positions	256
10.27.4.7 time	256
10.27.4.8 world	256
10.27.4.9 wrench	256
10.28gazebo::physics::ContactManager Class Reference	256
10.28.1 Detailed Description	257
10.28.2 Constructor & Destructor Documentation	257
10.28.2.1 ContactManager	257
10.28.2.2 ~ContactManager	257
10.28.3 Member Function Documentation	257
10.28.3.1 Clear	257
10.28.3.2 CreateFilter	257
10.28.3.3 CreateFilter	258
10.28.3.4 CreateFilter	258
10.28.3.5 GetContact	258
10.28.3.6 GetContactCount	258
10.28.3.7 GetContacts	258
10.28.3.8 Init	259
10.28.3.9 NewContact	259
10.28.3.10PublishContacts	259
10.28.3.11ResetCount	259
10.29gazebo::physics::ContactPublisher Class Reference	259
10.29.1 Detailed Description	260
10.29.2 Member Data Documentation	260
10.29.2.1 collisionNames	260

10.29.2.2 collisions	260
10.29.2.3 contacts	260
10.29.2.4 publisher	260
10.30 gazebo::sensors::ContactSensor Class Reference	260
10.30.1 Detailed Description	262
10.30.2 Constructor & Destructor Documentation	262
10.30.2.1 ContactSensor	262
10.30.2.2 ~ContactSensor	262
10.30.3 Member Function Documentation	262
10.30.3.1 Fini	262
10.30.3.2 GetCollisionContactCount	262
10.30.3.3 GetCollisionCount	262
10.30.3.4 GetCollisionName	263
10.30.3.5 GetContacts	263
10.30.3.6 GetContacts	264
10.30.3.7 Init	264
10.30.3.8 IsActive	264
10.30.3.9 Load	264
10.30.3.10 Load	264
10.30.3.11 UpdateImpl	265
10.31 gazebo::rendering::ContactVisual Class Reference	265
10.31.1 Detailed Description	266
10.31.2 Constructor & Destructor Documentation	266
10.31.2.1 ContactVisual	266
10.31.2.2 ~ContactVisual	266
10.31.3 Member Function Documentation	266
10.31.3.1 SetEnabled	266
10.32 gazebo::rendering::Conversions Class Reference	266
10.32.1 Detailed Description	267
10.32.2 Member Function Documentation	267
10.32.2.1 Convert	267
10.32.2.2 Convert	267
10.32.2.3 Convert	267
10.32.2.4 Convert	268
10.32.2.5 Convert	268
10.32.2.6 Convert	268
10.33 gazebo::physics::CylinderShape Class Reference	268

10.33.1 Detailed Description	270
10.33.2 Constructor & Destructor Documentation	270
10.33.2.1 CylinderShape	270
10.33.2.2 ~CylinderShape	270
10.33.3 Member Function Documentation	270
10.33.3.1 FillMsg	270
10.33.3.2 GetLength	270
10.33.3.3 GetRadius	271
10.33.3.4 Init	271
10.33.3.5 ProcessMsg	271
10.33.3.6 SetLength	271
10.33.3.7 SetRadius	271
10.33.3.8 SetScale	271
10.33.3.9 SetSize	272
10.34gazebo::rendering::DepthCamera Class Reference	272
10.34.1 Detailed Description	273
10.34.2 Constructor & Destructor Documentation	274
10.34.2.1 DepthCamera	274
10.34.2.2 ~DepthCamera	274
10.34.3 Member Function Documentation	274
10.34.3.1 ConnectNewDepthFrame	274
10.34.3.2 ConnectNewRGBPointCloud	274
10.34.3.3 CreateDepthTexture	275
10.34.3.4 DisconnectNewDepthFrame	275
10.34.3.5 DisconnectNewRGBPointCloud	275
10.34.3.6 Fini	275
10.34.3.7 GetDepthData	275
10.34.3.8 Init	275
10.34.3.9 Load	275
10.34.3.10Load	276
10.34.3.11PostRender	276
10.34.3.12SetDepthTarget	276
10.34.4 Member Data Documentation	276
10.34.4.1 depthTarget	276
10.34.4.2 depthTexture	276
10.34.4.3 depthViewport	276
10.35gazebo::sensors::DepthCameraSensor Class Reference	276

10.35.1 Constructor & Destructor Documentation	278
10.35.1.1 DepthCameraSensor	278
10.35.1.2 ~DepthCameraSensor	278
10.35.2 Member Function Documentation	278
10.35.2.1 Fini	278
10.35.2.2 GetDepthCamera	278
10.35.2.3 Init	278
10.35.2.4 Load	278
10.35.2.5 Load	278
10.35.2.6 SaveFrame	279
10.35.2.7 SetActive	279
10.35.2.8 UpdateImpl	279
10.36gazebo::util::DiagnosticManager Class Reference	279
10.36.1 Detailed Description	280
10.36.2 Member Function Documentation	281
10.36.2.1 GetLabel	281
10.36.2.2 GetLogPath	281
10.36.2.3 GetTime	281
10.36.2.4 GetTime	281
10.36.2.5 GetTimerCount	281
10.36.2.6 Init	282
10.36.2.7 Lap	282
10.36.2.8 StartTimer	282
10.36.2.9 StopTimer	282
10.37gazebo::util::DiagnosticTimer Class Reference	283
10.37.1 Detailed Description	283
10.37.2 Constructor & Destructor Documentation	283
10.37.2.1 DiagnosticTimer	283
10.37.2.2 ~DiagnosticTimer	284
10.37.3 Member Function Documentation	284
10.37.3.1 GetName	284
10.37.3.2 Lap	284
10.37.3.3 Start	284
10.37.3.4 Stop	284
10.38gazebo::rendering::DummyPageProvider Class Reference	284
10.38.1 Detailed Description	285
10.38.2 Member Function Documentation	285

10.38.2.1 loadProceduralPage	285
10.38.2.2 prepareProceduralPage	285
10.38.2.3 unloadProceduralPage	286
10.38.2.4 unprepareProceduralPage	286
10.39gazebo::rendering::DynamicLines Class Reference	286
10.39.1 Detailed Description	287
10.39.2 Constructor & Destructor Documentation	287
10.39.2.1 DynamicLines	287
10.39.2.2 ~DynamicLines	287
10.39.3 Member Function Documentation	287
10.39.3.1 AddPoint	288
10.39.3.2 AddPoint	288
10.39.3.3 Clear	288
10.39.3.4 GetMovableType	288
10.39.3.5 getMovableType	288
10.39.3.6 GetPoint	288
10.39.3.7 GetPointCount	289
10.39.3.8 SetColor	289
10.39.3.9 SetPoint	289
10.39.3.10Update	289
10.40gazebo::rendering::DynamicRenderable Class Reference	289
10.40.1 Detailed Description	291
10.40.2 Constructor & Destructor Documentation	291
10.40.2.1 DynamicRenderable	291
10.40.2.2 ~DynamicRenderable	291
10.40.3 Member Function Documentation	291
10.40.3.1 CreateVertexDeclaration	291
10.40.3.2 FillHardwareBuffers	291
10.40.3.3 getBoundingRadius	292
10.40.3.4 GetMovableType	292
10.40.3.5 GetOperationType	292
10.40.3.6 getSquaredViewDepth	292
10.40.3.7 Init	292
10.40.3.8 PrepareHardwareBuffers	293
10.40.3.9 SetOperationType	293
10.40.4 Member Data Documentation	293
10.40.4.1 indexBufferCapacity	293

10.40.4.2 vertexBufferCapacity	293
10.41 gazebo::physics::Entity Class Reference	293
10.41.1 Detailed Description	296
10.41.2 Constructor & Destructor Documentation	296
10.41.2.1 Entity	296
10.41.2.2 ~Entity	296
10.41.3 Member Function Documentation	297
10.41.3.1 Fini	297
10.41.3.2 GetBoundingBox	297
10.41.3.3 GetChildCollision	297
10.41.3.4 GetChildLink	297
10.41.3.5 GetCollisionBoundingBox	297
10.41.3.6 GetDirtyPose	298
10.41.3.7 GetInitialRelativePose	298
10.41.3.8 GetNearestEntityBelow	298
10.41.3.9 GetParentModel	298
10.41.3.10 GetRelativeAngularAccel	298
10.41.3.11 GetRelativeAngularVel	299
10.41.3.12 GetRelativeLinearAccel	299
10.41.3.13 GetRelativeLinearVel	299
10.41.3.14 GetRelativePose	299
10.41.3.15 GetWorldAngularAccel	299
10.41.3.16 GetWorldAngularVel	300
10.41.3.17 GetWorldLinearAccel	300
10.41.3.18 GetWorldLinearVel	300
10.41.3.19 GetWorldPose	300
10.41.3.20 IsCanonicalLink	300
10.41.3.21 IsStatic	301
10.41.3.22 Load	301
10.41.3.23 OnPoseChange	301
10.41.3.24 PlaceOnEntity	301
10.41.3.25 PlaceOnNearestEntityBelow	301
10.41.3.26 Reset	301
10.41.3.27 SetAnimation	302
10.41.3.28 SetAnimation	302
10.41.3.29 SetCanonicalLink	302
10.41.3.30 SetInitialRelativePose	302

10.41.3.31SetName	302
10.41.3.32SetRelativePose	302
10.41.3.33SetStatic	303
10.41.3.34SetWorldPose	303
10.41.3.35SetWorldTwist	303
10.41.3.36StopAnimation	303
10.41.3.37UpdateParameters	303
10.41.4 Member Data Documentation	304
10.41.4.1 animation	304
10.41.4.2 animationConnection	304
10.41.4.3 animationStartPose	304
10.41.4.4 connections	304
10.41.4.5 dirtyPose	304
10.41.4.6 node	304
10.41.4.7 parentEntity	304
10.41.4.8 prevAnimationTime	304
10.41.4.9 requestPub	304
10.41.4.10scale	304
10.41.4.11visPub	305
10.41.4.12visualMsg	305
10.42gazebo::event::Event Class Reference	305
10.42.1 Detailed Description	307
10.42.2 Constructor & Destructor Documentation	307
10.42.2.1 ~Event	307
10.42.3 Member Function Documentation	307
10.42.3.1 Disconnect	307
10.42.3.2 Disconnect	307
10.43gazebo::rendering::Events Class Reference	308
10.43.1 Detailed Description	308
10.43.2 Member Function Documentation	308
10.43.2.1 ConnectCreateScene	308
10.43.2.2 ConnectRemoveScene	309
10.43.2.3 DisconnectCreateScene	309
10.43.2.4 DisconnectRemoveScene	309
10.43.3 Member Data Documentation	309
10.43.3.1 createScene	309
10.43.3.2 removeScene	310

10.44gazebo::event::Events Class Reference	310
10.44.1 Detailed Description	312
10.44.2 Member Function Documentation	313
10.44.2.1 ConnectAddEntity	313
10.44.2.2 ConnectCreateEntity	313
10.44.2.3 ConnectDeleteEntity	313
10.44.2.4 ConnectDiagTimerStart	313
10.44.2.5 ConnectDiagTimerStop	314
10.44.2.6 ConnectPause	314
10.44.2.7 ConnectPostRender	314
10.44.2.8 ConnectPreRender	315
10.44.2.9 ConnectRender	315
10.44.2.10ConnectSetSelectedEntity	315
10.44.2.11ConnectSigInt	315
10.44.2.12ConnectStep	316
10.44.2.13ConnectStop	316
10.44.2.14ConnectWorldCreated	316
10.44.2.15ConnectWorldUpdateBegin	317
10.44.2.16ConnectWorldUpdateEnd	317
10.44.2.17DisconnectAddEntity	317
10.44.2.18DisconnectCreateEntity	317
10.44.2.19DisconnectDeleteEntity	318
10.44.2.20DisconnectDiagTimerStart	318
10.44.2.21DisconnectDiagTimerStop	318
10.44.2.22DisconnectPause	318
10.44.2.23DisconnectPostRender	318
10.44.2.24DisconnectPreRender	319
10.44.2.25DisconnectRender	319
10.44.2.26DisconnectSetSelectedEntity	319
10.44.2.27DisconnectSigInt	319
10.44.2.28DisconnectStep	319
10.44.2.29DisconnectStop	320
10.44.2.30DisconnectWorldCreated	320
10.44.2.31DisconnectWorldUpdateBegin	320
10.44.2.32DisconnectWorldUpdateEnd	320
10.44.3 Member Data Documentation	320
10.44.3.1 addEntity	320

10.44.3.2 deleteEntity	321
10.44.3.3 diagTimerStart	321
10.44.3.4 diagTimerStop	321
10.44.3.5 entityCreated	321
10.44.3.6 pause	321
10.44.3.7 postRender	321
10.44.3.8 preRender	321
10.44.3.9 render	321
10.44.3.10setSelectedEntity	321
10.44.3.11sigInt	322
10.44.3.12step	322
10.44.3.13stop	322
10.44.3.14worldCreated	322
10.44.3.15worldUpdateBegin	322
10.44.3.16worldUpdateEnd	322
10.45gazebo::event::EventT< T > Class Template Reference	322
10.45.1 Detailed Description	325
10.45.2 Member Function Documentation	325
10.45.2.1 operator()	325
10.45.2.2 operator()	325
10.45.2.3 operator()	325
10.45.2.4 operator()	325
10.45.2.5 operator()	326
10.45.2.6 operator()	326
10.45.2.7 operator()	326
10.45.2.8 operator()	326
10.45.2.9 operator()	327
10.45.2.10operator()	327
10.45.2.11operator()	328
10.45.2.12Signal	328
10.45.2.13Signal	328
10.45.2.14Signal	328
10.45.2.15Signal	328
10.45.2.16Signal	329
10.45.2.17Signal	329
10.45.2.18Signal	329
10.45.2.19Signal	330

10.45.2.20Signal	330
10.45.2.21Signal	330
10.45.2.22Signal	331
10.46gazebo::common::Exception Class Reference	331
10.46.1 Detailed Description	332
10.46.2 Constructor & Destructor Documentation	332
10.46.2.1 Exception	332
10.46.2.2 Exception	332
10.46.2.3 ~Exception	332
10.46.3 Member Function Documentation	332
10.46.3.1 GetErrorFile	333
10.46.3.2 GetErrorStr	333
10.46.3.3 Print	333
10.46.4 Friends And Related Function Documentation	333
10.46.4.1 operator<<	333
10.47gazebo::sensors::ForceTorqueSensor Class Reference	333
10.47.1 Detailed Description	335
10.47.2 Constructor & Destructor Documentation	335
10.47.2.1 ForceTorqueSensor	335
10.47.2.2 ~ForceTorqueSensor	335
10.47.3 Member Function Documentation	335
10.47.3.1 ConnectUpdate	335
10.47.3.2 DisconnectUpdate	335
10.47.3.3 Fini	336
10.47.3.4 GetForce	336
10.47.3.5 GetTopic	336
10.47.3.6 GetTorque	336
10.47.3.7 Init	336
10.47.3.8 IsActive	336
10.47.3.9 Load	337
10.47.3.10UpdateImpl	337
10.47.4 Member Data Documentation	337
10.47.4.1 update	337
10.48gazebo::rendering::FPSViewController Class Reference	337
10.48.1 Detailed Description	338
10.48.2 Constructor & Destructor Documentation	339
10.48.2.1 FPSViewController	339

10.48.2.2	~FPSViewController	339
10.48.3	Member Function Documentation	339
10.48.3.1	GetTypeString	339
10.48.3.2	HandleKeyPressEvent	339
10.48.3.3	HandleKeyReleaseEvent	339
10.48.3.4	HandleMouseEvent	339
10.48.3.5	Init	340
10.48.3.6	Update	340
10.49	google::protobuf::compiler::cpp::GazeboGenerator Class Reference	340
10.49.1	Detailed Description	341
10.49.2	Constructor & Destructor Documentation	341
10.49.2.1	GazeboGenerator	341
10.49.2.2	~GazeboGenerator	341
10.49.3	Member Function Documentation	341
10.49.3.1	Generate	341
10.50	gazebo::sensors::GpsSensor Class Reference	341
10.50.1	Detailed Description	342
10.50.2	Constructor & Destructor Documentation	342
10.50.2.1	GpsSensor	342
10.50.2.2	~GpsSensor	342
10.50.3	Member Function Documentation	342
10.50.3.1	Fini	342
10.50.3.2	GetAltitude	343
10.50.3.3	GetLatitude	343
10.50.3.4	GetLongitude	343
10.50.3.5	Init	343
10.50.3.6	Load	343
10.50.3.7	Load	343
10.50.3.8	UpdateImpl	343
10.51	gazebo::rendering::GpuLaser Class Reference	344
10.51.1	Detailed Description	346
10.51.2	Constructor & Destructor Documentation	347
10.51.2.1	GpuLaser	347
10.51.2.2	~GpuLaser	347
10.51.3	Member Function Documentation	347
10.51.3.1	ConnectNewLaserFrame	347
10.51.3.2	CreateLaserTexture	347

10.51.3.3 DisconnectNewLaserFrame	347
10.51.3.4 Fini	348
10.51.3.5 GetCameraCount	348
10.51.3.6 GetCosHorzFOV	348
10.51.3.7 GetCosVertFOV	348
10.51.3.8 GetFarClip	348
10.51.3.9 GetHorzFOV	348
10.51.3.10GetHorzHalfAngle	349
10.51.3.11GetLaserData	349
10.51.3.12GetNearClip	349
10.51.3.13GetRayCountRatio	349
10.51.3.14GetVertFOV	349
10.51.3.15GetVertHalfAngle	349
10.51.3.16Init	350
10.51.3.17IsHorizontal	350
10.51.3.18Load	350
10.51.3.19Load	350
10.51.3.20notifyRenderSingleObject	350
10.51.3.21PostRender	350
10.51.3.22SetCameraCount	350
10.51.3.23SetCosHorzFOV	350
10.51.3.24SetCosVertFOV	351
10.51.3.25SetFarClip	351
10.51.3.26SetHorzFOV	351
10.51.3.27SetHorzHalfAngle	351
10.51.3.28SetIsHorizontal	351
10.51.3.29SetNearClip	351
10.51.3.30SetRangeCount	352
10.51.3.31SetRayCountRatio	352
10.51.3.32SetVertFOV	352
10.51.3.33SetVertHalfAngle	352
10.51.4 Member Data Documentation	352
10.51.4.1 cameraCount	352
10.51.4.2 chfov	352
10.51.4.3 cvfov	352
10.51.4.4 far	353
10.51.4.5 hfov	353

10.51.4.6 horzHalfAngle	353
10.51.4.7 isHorizontal	353
10.51.4.8 near	353
10.51.4.9 rayCountRatio	353
10.51.4.10vertHalfAngle	353
10.51.4.11vfov	353
10.52gazebo::sensors::GpuRaySensor Class Reference	353
10.52.1 Constructor & Destructor Documentation	356
10.52.1.1 GpuRaySensor	356
10.52.1.2 ~GpuRaySensor	357
10.52.2 Member Function Documentation	357
10.52.2.1 ConnectNewLaserFrame	357
10.52.2.2 DisconnectNewLaserFrame	357
10.52.2.3 Fini	357
10.52.2.4 GetAngleMax	357
10.52.2.5 GetAngleMin	357
10.52.2.6 GetAngleResolution	357
10.52.2.7 GetCameraCount	358
10.52.2.8 GetCosHorzFOV	358
10.52.2.9 GetCosVertFOV	358
10.52.2.10GetFiducial	358
10.52.2.11GetHorzFOV	358
10.52.2.12GetHorzHalfAngle	359
10.52.2.13GetLaserCamera	359
10.52.2.14GetRange	359
10.52.2.15GetRangeCount	359
10.52.2.16GetRangeCountRatio	359
10.52.2.17GetRangeMax	360
10.52.2.18GetRangeMin	360
10.52.2.19GetRangeResolution	360
10.52.2.20GetRanges	360
10.52.2.21GetRayCount	360
10.52.2.22GetRayCountRatio	360
10.52.2.23GetRetro	361
10.52.2.24GetTopic	361
10.52.2.25GetVertFOV	361
10.52.2.26GetVertHalfAngle	361

10.52.2.27	GetVerticalAngleMax	361
10.52.2.28	GetVerticalAngleMin	362
10.52.2.29	GetVerticalRangeCount	362
10.52.2.30	GetVerticalRayCount	362
10.52.2.31	Init	362
10.52.2.32	IsActive	362
10.52.2.33	IsHorizontal	362
10.52.2.34	Load	363
10.52.2.35	Load	363
10.52.2.36	SetAngleMax	363
10.52.2.37	SetAngleMin	363
10.52.2.38	SetVerticalAngleMax	363
10.52.2.39	SetVerticalAngleMin	363
10.52.2.40	UpdateImpl	364
10.52.3	Member Data Documentation	364
10.52.3.1	cameraElem	364
10.52.3.2	horzElem	364
10.52.3.3	horzRangeCount	364
10.52.3.4	horzRayCount	364
10.52.3.5	rangeCountRatio	364
10.52.3.6	rangeElem	364
10.52.3.7	scanElem	364
10.52.3.8	vertElem	364
10.52.3.9	vertRangeCount	365
10.52.3.10	vertRayCount	365
10.53	gazebo::rendering::Grid Class Reference	365
10.53.1	Detailed Description	366
10.53.2	Constructor & Destructor Documentation	366
10.53.2.1	Grid	366
10.53.2.2	~Grid	366
10.53.3	Member Function Documentation	366
10.53.3.1	Enable	366
10.53.3.2	GetCellCount	366
10.53.3.3	GetCellLength	366
10.53.3.4	GetColor	367
10.53.3.5	GetHeight	367
10.53.3.6	GetLineWidth	367

10.53.3.7	GetSceneNode	367
10.53.3.8	Init	367
10.53.3.9	SetCellCount	367
10.53.3.10	SetCellLength	368
10.53.3.11	SetColor	368
10.53.3.12	SetHeight	368
10.53.3.13	SetLineWidth	368
10.53.3.14	SetUserData	368
10.54	gazebo::physics::Gripper Class Reference	368
10.54.1	Detailed Description	369
10.54.2	Constructor & Destructor Documentation	369
10.54.2.1	Gripper	369
10.54.2.2	~Gripper	369
10.54.3	Member Function Documentation	369
10.54.3.1	GetName	369
10.54.3.2	Init	370
10.54.3.3	IsAttached	370
10.54.3.4	Load	370
10.54.4	Member Data Documentation	370
10.54.4.1	node	370
10.55	gazebo::rendering::GUIOverlay Class Reference	370
10.55.1	Detailed Description	371
10.55.2	Constructor & Destructor Documentation	371
10.55.2.1	GUIOverlay	371
10.55.2.2	~GUIOverlay	371
10.55.3	Member Function Documentation	371
10.55.3.1	AttachCameraToImage	371
10.55.3.2	AttachCameraToImage	372
10.55.3.3	ButtonCallback	372
10.55.3.4	CreateWindow	372
10.55.3.5	HandleKeyPressEvent	373
10.55.3.6	HandleKeyReleaseEvent	373
10.55.3.7	HandleMouseEvent	373
10.55.3.8	Hide	373
10.55.3.9	Init	373
10.55.3.10	IsInitialized	374
10.55.3.11	LoadLayout	374

10.55.3.12	Resize	374
10.55.3.13	Show	374
10.55.3.14	Update	374
10.56	gazebo::rendering::GzTerrainMatGen Class Reference	374
10.56.1	Constructor & Destructor Documentation	375
10.56.1.1	GzTerrainMatGen	375
10.56.1.2	~GzTerrainMatGen	375
10.57	gazebo::rendering::Heightmap Class Reference	375
10.57.1	Detailed Description	376
10.57.2	Constructor & Destructor Documentation	377
10.57.2.1	Heightmap	377
10.57.2.2	~Heightmap	377
10.57.3	Member Function Documentation	377
10.57.3.1	Flatten	377
10.57.3.2	GetAvgHeight	377
10.57.3.3	GetHeight	377
10.57.3.4	GetImage	378
10.57.3.5	GetMouseHit	378
10.57.3.6	GetOgreTerrain	378
10.57.3.7	Load	378
10.57.3.8	LoadFromMsg	378
10.57.3.9	Lower	379
10.57.3.10	Raise	379
10.57.3.11	SetWireframe	379
10.57.3.12	Smooth	379
10.57.3.13	SplitHeights	380
10.57.4	Member Data Documentation	380
10.57.4.1	NumTerrainSubdivisions	380
10.58	gazebo::physics::HeightmapShape Class Reference	380
10.58.1	Detailed Description	382
10.58.2	Constructor & Destructor Documentation	382
10.58.2.1	HeightmapShape	382
10.58.2.2	~HeightmapShape	383
10.58.3	Member Function Documentation	383
10.58.3.1	FillMsg	383
10.58.3.2	GetHeight	383
10.58.3.3	GetImage	383

10.58.3.4	GetMaxHeight	383
10.58.3.5	GetMinHeight	383
10.58.3.6	GetPos	384
10.58.3.7	GetSize	384
10.58.3.8	GetSubSampling	384
10.58.3.9	GetURI	384
10.58.3.10	GetVertexCount	384
10.58.3.11	Init	384
10.58.3.12	Load	385
10.58.3.13	ProcessMsg	385
10.58.3.14	SetScale	385
10.58.4	Member Data Documentation	385
10.58.4.1	flipY	385
10.58.4.2	heights	385
10.58.4.3	img	385
10.58.4.4	subSampling	385
10.58.4.5	vertSize	386
10.59	gazebo::physics::Hinge2Joint< T > Class Template Reference	386
10.59.1	Detailed Description	386
10.59.2	Constructor & Destructor Documentation	387
10.59.2.1	Hinge2Joint	387
10.59.2.2	~Hinge2Joint	387
10.59.3	Member Function Documentation	387
10.59.3.1	GetAngleCount	387
10.59.3.2	Load	387
10.60	gazebo::physics::HingeJoint< T > Class Template Reference	387
10.60.1	Detailed Description	388
10.60.2	Constructor & Destructor Documentation	388
10.60.2.1	HingeJoint	388
10.60.2.2	~HingeJoint	389
10.60.3	Member Function Documentation	389
10.60.3.1	GetAngleCount	389
10.60.3.2	Init	389
10.60.3.3	Load	389
10.61	gazebo::common::Image Class Reference	389
10.61.1	Detailed Description	390
10.61.2	Member Enumeration Documentation	391

10.61.2.1 PixelFormat	391
10.61.3 Constructor & Destructor Documentation	391
10.61.3.1 Image	391
10.61.3.2 ~Image	391
10.61.4 Member Function Documentation	391
10.61.4.1 ConvertPixelFormat	392
10.61.4.2 GetAvgColor	392
10.61.4.3 GetBPP	392
10.61.4.4 GetData	392
10.61.4.5 GetFilename	392
10.61.4.6 GetHeight	393
10.61.4.7 GetMaxColor	393
10.61.4.8 GetPitch	393
10.61.4.9 GetPixel	393
10.61.4.10GetPixelFormat	393
10.61.4.11GetRGBData	393
10.61.4.12GetWidth	394
10.61.4.13Load	394
10.61.4.14Rescale	394
10.61.4.15SavePNG	394
10.61.4.16SetFromData	394
10.61.4.17Valid	395
10.62gazebo::sensors::ImuSensor Class Reference	395
10.62.1 Detailed Description	396
10.62.2 Constructor & Destructor Documentation	396
10.62.2.1 ImuSensor	396
10.62.2.2 ~ImuSensor	396
10.62.3 Member Function Documentation	396
10.62.3.1 Fini	396
10.62.3.2 GetAngularVelocity	397
10.62.3.3 GetImuMessage	397
10.62.3.4 GetLinearAcceleration	397
10.62.3.5 GetOrientation	397
10.62.3.6 Init	397
10.62.3.7 IsActive	397
10.62.3.8 Load	398
10.62.3.9 Load	398

10.62.3.10SetReferencePose	398
10.62.3.11UpdateImpl	398
10.63gazebo::physics::Inertial Class Reference	398
10.63.1 Detailed Description	400
10.63.2 Constructor & Destructor Documentation	401
10.63.2.1 Inertial	401
10.63.2.2 Inertial	401
10.63.2.3 Inertial	401
10.63.2.4 ~Inertial	401
10.63.3 Member Function Documentation	401
10.63.3.1 GetCoG	401
10.63.3.2 GetInertial	401
10.63.3.3 GetIXX	402
10.63.3.4 GetIXY	402
10.63.3.5 GetIXZ	402
10.63.3.6 GetIYY	402
10.63.3.7 GetIYZ	402
10.63.3.8 GetIZZ	402
10.63.3.9 GetMass	403
10.63.3.10GetMOI	403
10.63.3.11GetMOI	403
10.63.3.12GetPose	403
10.63.3.13GetPrincipalMoments	403
10.63.3.14GetProductsofInertia	403
10.63.3.15Load	404
10.63.3.16operator+	404
10.63.3.17operator+=	404
10.63.3.18operator=	404
10.63.3.19ProcessMsg	404
10.63.3.20Reset	405
10.63.3.21Rotate	405
10.63.3.22SetCoG	405
10.63.3.23SetCoG	405
10.63.3.24SetCoG	405
10.63.3.25SetCoG	406
10.63.3.26SetInertiaMatrix	406
10.63.3.27SetIXX	406

10.63.3.28	SetIXY	406
10.63.3.29	SetIXZ	406
10.63.3.30	SetIYY	406
10.63.3.31	SetIYZ	407
10.63.3.32	SetIZZ	407
10.63.3.33	SetMass	407
10.63.3.34	SetMOI	407
10.63.3.35	UpdateParameters	407
10.63.4	Friends And Related Function Documentation	407
10.63.4.1	operator<<	407
10.64	gazebo::common::InternalError Class Reference	408
10.64.1	Detailed Description	409
10.64.2	Constructor & Destructor Documentation	409
10.64.2.1	InternalError	409
10.64.2.2	InternalError	409
10.64.2.3	~InternalError	409
10.65	gazebo::transport::IOManager Class Reference	409
10.65.1	Detailed Description	410
10.65.2	Constructor & Destructor Documentation	410
10.65.2.1	IOManager	410
10.65.2.2	~IOManager	410
10.65.3	Member Function Documentation	410
10.65.3.1	DecCount	410
10.65.3.2	GetCount	410
10.65.3.3	GetIO	410
10.65.3.4	IncCount	410
10.65.3.5	Stop	411
10.66	gazebo::physics::Joint Class Reference	411
10.66.1	Detailed Description	414
10.66.2	Member Enumeration Documentation	415
10.66.2.1	Attribute	415
10.66.3	Constructor & Destructor Documentation	415
10.66.3.1	Joint	415
10.66.3.2	~Joint	415
10.66.4	Member Function Documentation	415
10.66.4.1	ApplyDamping	415
10.66.4.2	AreConnected	415

10.66.4.3 Attach	416
10.66.4.4 CacheForceTorque	416
10.66.4.5 CheckAndTruncateForce	416
10.66.4.6 ConnectJointUpdate	416
10.66.4.7 Detach	417
10.66.4.8 DisconnectJointUpdate	417
10.66.4.9 FillMsg	417
10.66.4.10GetAnchor	417
10.66.4.11GetAngle	417
10.66.4.12GetAngleCount	418
10.66.4.13GetAngleImpl	418
10.66.4.14GetAttribute	418
10.66.4.15GetChild	418
10.66.4.16GetDamping	419
10.66.4.17GetDampingCoefficient	419
10.66.4.18GetEffortLimit	419
10.66.4.19GetForce	419
10.66.4.20GetForceTorque	420
10.66.4.21GetGlobalAxis	420
10.66.4.22GetHighStop	420
10.66.4.23GetInertiaRatio	421
10.66.4.24GetJointLink	421
10.66.4.25GetLinkForce	421
10.66.4.26GetLinkTorque	422
10.66.4.27GetLocalAxis	422
10.66.4.28GetLowerLimit	422
10.66.4.29GetLowStop	422
10.66.4.30GetMaxForce	423
10.66.4.31GetParent	423
10.66.4.32GetUpperLimit	423
10.66.4.33GetVelocity	423
10.66.4.34GetVelocityLimit	424
10.66.4.35nit	424
10.66.4.36Load	424
10.66.4.37Load	424
10.66.4.38Reset	425
10.66.4.39SetAnchor	425

10.66.4.40	SetAngle	425
10.66.4.41	SetAttribute	425
10.66.4.42	SetAxis	426
10.66.4.43	SetDamping	426
10.66.4.44	SetForce	426
10.66.4.45	SetHighStop	427
10.66.4.46	SetLowStop	427
10.66.4.47	SetMaxForce	427
10.66.4.48	SetModel	427
10.66.4.49	SetProvideFeedback	427
10.66.4.50	SetState	428
10.66.4.51	SetVelocity	428
10.66.4.52	Update	428
10.66.4.53	UpdateParameters	428
10.66.5	Member Data Documentation	428
10.66.5.1	anchorLink	428
10.66.5.2	anchorPos	428
10.66.5.3	anchorPose	429
10.66.5.4	applyDamping	429
10.66.5.5	childLink	429
10.66.5.6	dampingCoefficient	429
10.66.5.7	effortLimit	429
10.66.5.8	inertiaRatio	429
10.66.5.9	lowerLimit	429
10.66.5.10	model	429
10.66.5.11	parentLink	429
10.66.5.12	provideFeedback	429
10.66.5.13	upperLimit	430
10.66.5.14	useCFMDamping	430
10.66.5.15	velocityLimit	430
10.66.5.16	wrench	430
10.67	Joint_TEST Class Reference	430
10.67.1	Constructor & Destructor Documentation	431
10.67.1.1	Joint_TEST	431
10.67.2	Member Function Documentation	431
10.67.2.1	ForceTorque1	431
10.67.2.2	ForceTorque2	432

10.67.2.3	GetForceTorqueWithAppliedForce	432
10.67.2.4	JointCreationDestructionTest	432
10.67.2.5	JointTorqueTest	432
10.67.2.6	SetUp	432
10.67.2.7	SpawnJoint	432
10.67.2.8	SpawnJoint	433
10.67.2.9	SpawnJointRotational	433
10.67.2.10	SpawnJointRotationalWorld	433
10.67.2.11	SpawnJointTypes	433
10.67.3	Member Data Documentation	434
10.67.3.1	jointType	434
10.67.3.2	physicsEngine	434
10.68	gazebo::physics::JointController Class Reference	434
10.68.1	Detailed Description	434
10.68.2	Constructor & Destructor Documentation	435
10.68.2.1	JointController	435
10.68.3	Member Function Documentation	435
10.68.3.1	AddJoint	435
10.68.3.2	Reset	435
10.68.3.3	SetJointPosition	435
10.68.3.4	SetJointPosition	435
10.68.3.5	SetJointPositions	435
10.68.3.6	Update	436
10.69	gazebo::physics::JointState Class Reference	436
10.69.1	Detailed Description	437
10.69.2	Constructor & Destructor Documentation	437
10.69.2.1	JointState	437
10.69.2.2	JointState	437
10.69.2.3	JointState	438
10.69.2.4	JointState	438
10.69.2.5	~JointState	438
10.69.3	Member Function Documentation	438
10.69.3.1	FillSDF	438
10.69.3.2	GetAngle	438
10.69.3.3	GetAngleCount	439
10.69.3.4	GetAngles	439
10.69.3.5	IsZero	439

10.69.3.6 Load	439
10.69.3.7 Load	439
10.69.3.8 operator+	439
10.69.3.9 operator-	440
10.69.3.10operator=	440
10.69.4 Friends And Related Function Documentation	440
10.69.4.1 operator<<	440
10.70gazebo::rendering::JointVisual Class Reference	441
10.70.1 Detailed Description	441
10.70.2 Constructor & Destructor Documentation	441
10.70.2.1 JointVisual	441
10.70.2.2 ~JointVisual	442
10.70.3 Member Function Documentation	442
10.70.3.1 Load	442
10.71gazebo::physics::JointWrench Class Reference	442
10.71.1 Detailed Description	443
10.71.2 Member Function Documentation	443
10.71.2.1 operator+	443
10.71.2.2 operator-	443
10.71.2.3 operator=	443
10.71.3 Member Data Documentation	443
10.71.3.1 body1Force	444
10.71.3.2 body1Torque	444
10.71.3.3 body2Force	444
10.71.3.4 body2Torque	444
10.72gazebo::common::KeyEvent Class Reference	444
10.72.1 Detailed Description	445
10.72.2 Member Enumeration Documentation	445
10.72.2.1 EventType	445
10.72.3 Constructor & Destructor Documentation	445
10.72.3.1 KeyEvent	445
10.72.4 Member Data Documentation	445
10.72.4.1 key	445
10.72.4.2 type	445
10.73gazebo::common::KeyFrame Class Reference	445
10.73.1 Detailed Description	446
10.73.2 Constructor & Destructor Documentation	446

10.73.2.1 KeyFrame	446
10.73.2.2 ~KeyFrame	446
10.73.3 Member Function Documentation	447
10.73.3.1 GetTime	447
10.73.4 Member Data Documentation	447
10.73.4.1 time	447
10.74 gazebo::rendering::LaserVisual Class Reference	447
10.74.1 Detailed Description	448
10.74.2 Constructor & Destructor Documentation	448
10.74.2.1 LaserVisual	448
10.74.2.2 ~LaserVisual	448
10.74.3 Member Function Documentation	448
10.74.3.1 SetEmissive	448
10.75 gazebo::rendering::Light Class Reference	448
10.75.1 Detailed Description	450
10.75.2 Constructor & Destructor Documentation	450
10.75.2.1 Light	450
10.75.2.2 ~Light	450
10.75.3 Member Function Documentation	450
10.75.3.1 FillMsg	450
10.75.3.2 GetDiffuseColor	451
10.75.3.3 GetDirection	451
10.75.3.4 GetName	451
10.75.3.5 GetPosition	451
10.75.3.6 GetSpecularColor	451
10.75.3.7 GetType	451
10.75.3.8 Load	452
10.75.3.9 Load	452
10.75.3.10 LoadFromMsg	452
10.75.3.11 OnPoseChange	452
10.75.3.12 SetAttenuation	452
10.75.3.13 SetCastShadows	452
10.75.3.14 SetDiffuseColor	452
10.75.3.15 SetDirection	453
10.75.3.16 SetLightType	453
10.75.3.17 SetName	453
10.75.3.18 SetPosition	453

10.75.3.19	SetRange	453
10.75.3.20	SetSelected	453
10.75.3.21	SetSpecularColor	454
10.75.3.22	SetSpotFalloff	454
10.75.3.23	SetSpotInnerAngle	454
10.75.3.24	SetSpotOuterAngle	454
10.75.3.25	ShowVisual	454
10.75.3.26	ToggleShowVisual	454
10.75.3.27	UpdateFromMsg	455
10.76	gazebo::physics::Link Class Reference	455
10.76.1	Detailed Description	460
10.76.2	Member Typedef Documentation	460
10.76.2.1	Visuals_M	460
10.76.3	Constructor & Destructor Documentation	460
10.76.3.1	Link	460
10.76.3.2	~Link	460
10.76.4	Member Function Documentation	460
10.76.4.1	AddChildJoint	460
10.76.4.2	AddForce	460
10.76.4.3	AddForceAtRelativePosition	461
10.76.4.4	AddForceAtWorldPosition	461
10.76.4.5	AddParentJoint	461
10.76.4.6	AddRelativeForce	461
10.76.4.7	AddRelativeTorque	461
10.76.4.8	AddTorque	462
10.76.4.9	AttachStaticModel	462
10.76.4.10	ConnectEnabled	462
10.76.4.11	DetachAllStaticModels	462
10.76.4.12	DetachStaticModel	462
10.76.4.13	DisconnectEnabled	462
10.76.4.14	FillMsg	463
10.76.4.15	Fin	463
10.76.4.16	GetAngularDamping	463
10.76.4.17	GetBoundingBox	463
10.76.4.18	GetChildJoints	463
10.76.4.19	GetChildJointsLinks	463
10.76.4.20	GetCollision	464

10.76.4.21	GetCollision	464
10.76.4.22	GetCollisions	464
10.76.4.23	GetEnabled	464
10.76.4.24	GetGravityMode	464
10.76.4.25	GetInertial	465
10.76.4.26	GetKinematic	465
10.76.4.27	GetLinearDamping	465
10.76.4.28	GetModel	465
10.76.4.29	GetParentJoints	465
10.76.4.30	GetParentJointsLinks	465
10.76.4.31	GetRelativeAngularAccel	466
10.76.4.32	GetRelativeAngularVel	466
10.76.4.33	GetRelativeForce	466
10.76.4.34	GetRelativeLinearAccel	466
10.76.4.35	GetRelativeLinearVel	466
10.76.4.36	GetRelativeTorque	467
10.76.4.37	GetSelfCollide	467
10.76.4.38	GetSensorCount	467
10.76.4.39	GetSensorName	467
10.76.4.40	GetWorldAngularAccel	467
10.76.4.41	GetWorldCoGLinearVel	468
10.76.4.42	GetWorldCoGPose	468
10.76.4.43	GetWorldForce	468
10.76.4.44	GetWorldLinearAccel	468
10.76.4.45	GetWorldLinearVel	468
10.76.4.46	GetWorldLinearVel	469
10.76.4.47	GetWorldTorque	469
10.76.4.48	Init	469
10.76.4.49	Load	469
10.76.4.50	OnPoseChange	470
10.76.4.51	ProcessMsg	470
10.76.4.52	RemoveChild	470
10.76.4.53	RemoveChildJoint	470
10.76.4.54	RemoveCollision	470
10.76.4.55	RemoveParentJoint	470
10.76.4.56	Reset	470
10.76.4.57	ResetPhysicsStates	471

10.76.4.58SetAngularAccel	471
10.76.4.59SetAngularDamping	471
10.76.4.60SetAngularVel	471
10.76.4.61SetAutoDisable	471
10.76.4.62SetCollideMode	471
10.76.4.63SetEnabled	472
10.76.4.64SetForce	472
10.76.4.65SetGravityMode	472
10.76.4.66SetInertial	472
10.76.4.67SetKinematic	472
10.76.4.68SetLaserRetro	473
10.76.4.69SetLinearAccel	473
10.76.4.70SetLinearDamping	473
10.76.4.71SetLinearVel	473
10.76.4.72SetLinkStatic	473
10.76.4.73SetPublishData	473
10.76.4.74SetScale	474
10.76.4.75SetSelected	474
10.76.4.76SetSelfCollide	474
10.76.4.77SetState	474
10.76.4.78SetTorque	474
10.76.4.79Update	475
10.76.4.80UpdateMass	475
10.76.4.81UpdateParameters	475
10.76.4.82UpdateSurface	475
10.76.5 Member Data Documentation	475
10.76.5.1 angularAccel	475
10.76.5.2 attachedModelsOffset	475
10.76.5.3 cgVisuals	475
10.76.5.4 inertial	475
10.76.5.5 linearAccel	476
10.76.5.6 visuals	476
10.77gazebo::physics::LinkState Class Reference	476
10.77.1 Detailed Description	477
10.77.2 Constructor & Destructor Documentation	478
10.77.2.1 LinkState	478
10.77.2.2 LinkState	478

10.77.2.3 LinkState	478
10.77.2.4 LinkState	478
10.77.2.5 ~LinkState	478
10.77.3 Member Function Documentation	478
10.77.3.1 FillSDF	478
10.77.3.2 GetAcceleration	479
10.77.3.3 GetCollisionState	479
10.77.3.4 GetCollisionState	479
10.77.3.5 GetCollisionStateCount	480
10.77.3.6 GetCollisionStates	480
10.77.3.7 GetPose	480
10.77.3.8 GetVelocity	480
10.77.3.9 GetWrench	480
10.77.3.10 IsZero	480
10.77.3.11 Load	481
10.77.3.12 Load	481
10.77.3.13 operator+	481
10.77.3.14 operator-	481
10.77.3.15 operator=	482
10.77.3.16 SetRealTime	482
10.77.3.17 SetSimTime	482
10.77.3.18 SetWallTime	482
10.77.4 Friends And Related Function Documentation	482
10.77.4.1 operator<<	483
10.78 gazebo::util::LogPlay Class Reference	483
10.78.1 Member Function Documentation	484
10.78.1.1 GetChunk	484
10.78.1.2 GetChunkCount	484
10.78.1.3 GetEncoding	484
10.78.1.4 GetGazeboVersion	485
10.78.1.5 GetHeader	485
10.78.1.6 GetLogVersion	485
10.78.1.7 GetRandSeed	485
10.78.1.8 IsOpen	485
10.78.1.9 Open	485
10.78.1.10 Step	486
10.79 Logplay Class Reference	486

10.79.1 Detailed Description	486
10.80 gazebo::util::LogRecord Class Reference	486
10.80.1 Detailed Description	488
10.80.2 Member Function Documentation	488
10.80.2.1 Add	488
10.80.2.2 Fini	489
10.80.2.3 GetBasePath	489
10.80.2.4 GetBufferSize	489
10.80.2.5 GetEncoding	489
10.80.2.6 GetFilename	489
10.80.2.7 GetFileSize	489
10.80.2.8 GetFirstUpdate	490
10.80.2.9 GetPaused	490
10.80.2.10 GetRunning	490
10.80.2.11 GetRunTime	490
10.80.2.12 Init	490
10.80.2.13 IsReadyToStart	491
10.80.2.14 Notify	491
10.80.2.15 Remove	491
10.80.2.16 SetBasePath	491
10.80.2.17 SetPaused	491
10.80.2.18 Start	492
10.80.2.19 Stop	492
10.80.2.20 Write	492
10.81 gazebo::physics::MapShape Class Reference	492
10.81.1 Detailed Description	494
10.81.2 Constructor & Destructor Documentation	494
10.81.2.1 MapShape	494
10.81.2.2 ~MapShape	494
10.81.3 Member Function Documentation	494
10.81.3.1 FillMsg	494
10.81.3.2 GetGranularity	494
10.81.3.3 GetHeight	495
10.81.3.4 GetScale	495
10.81.3.5 GetThreshold	495
10.81.3.6 GetURI	495
10.81.3.7 Init	495

10.81.3.8 Load	495
10.81.3.9 ProcessMsg	496
10.81.3.10SetScale	496
10.81.3.11Update	496
10.82gazebo::Master Class Reference	496
10.82.1 Detailed Description	497
10.82.2 Constructor & Destructor Documentation	497
10.82.2.1 Master	497
10.82.2.2 ~Master	497
10.82.3 Member Function Documentation	497
10.82.3.1 Fini	497
10.82.3.2 Init	497
10.82.3.3 Run	497
10.82.3.4 RunOnce	497
10.82.3.5 RunThread	498
10.82.3.6 Stop	498
10.83gazebo::common::Material Class Reference	498
10.83.1 Detailed Description	500
10.83.2 Member Enumeration Documentation	500
10.83.2.1 BlendMode	500
10.83.2.2 ShadeMode	501
10.83.3 Constructor & Destructor Documentation	501
10.83.3.1 Material	501
10.83.3.2 ~Material	501
10.83.3.3 Material	501
10.83.4 Member Function Documentation	501
10.83.4.1 GetAmbient	501
10.83.4.2 GetBlendFactors	501
10.83.4.3 GetBlendMode	502
10.83.4.4 GetDepthWrite	502
10.83.4.5 GetDiffuse	502
10.83.4.6 GetEmissive	502
10.83.4.7 GetLighting	502
10.83.4.8 GetName	502
10.83.4.9 GetPointSize	503
10.83.4.10GetShadeMode	503
10.83.4.11GetShininess	503

10.83.4.12	GetSpecular	503
10.83.4.13	GetTextureImage	503
10.83.4.14	GetTransparency	503
10.83.4.15	SetAmbient	504
10.83.4.16	SetBlendFactors	504
10.83.4.17	SetBlendMode	504
10.83.4.18	SetDepthWrite	504
10.83.4.19	SetDiffuse	504
10.83.4.20	SetEmissive	504
10.83.4.21	SetLighting	505
10.83.4.22	SetPointSize	505
10.83.4.23	SetShadeMode	505
10.83.4.24	SetShininess	505
10.83.4.25	SetSpecular	505
10.83.4.26	SetTextureImage	505
10.83.4.27	SetTextureImage	506
10.83.4.28	SetTransparency	506
10.83.5	Friends And Related Function Documentation	506
10.83.5.1	operator<<	506
10.83.6	Member Data Documentation	506
10.83.6.1	ambient	506
10.83.6.2	blendMode	506
10.83.6.3	BlendModeStr	506
10.83.6.4	diffuse	506
10.83.6.5	emissive	506
10.83.6.6	name	506
10.83.6.7	pointSize	507
10.83.6.8	shadeMode	507
10.83.6.9	ShadeModeStr	507
10.83.6.10	shininess	507
10.83.6.11	specular	507
10.83.6.12	texImage	507
10.83.6.13	transparency	507
10.84	gazebo::math::Matrix3 Class Reference	507
10.84.1	Detailed Description	508
10.84.2	Constructor & Destructor Documentation	508
10.84.2.1	Matrix3	508

10.84.2.2 Matrix3	509
10.84.2.3 Matrix3	509
10.84.2.4 ~Matrix3	509
10.84.3 Member Function Documentation	509
10.84.3.1 operator*	509
10.84.3.2 operator*	509
10.84.3.3 operator+	510
10.84.3.4 operator-	510
10.84.3.5 operator==	510
10.84.3.6 operator[]	510
10.84.3.7 operator[]	510
10.84.3.8 SetCol	511
10.84.3.9 SetFromAxes	511
10.84.3.10SetFromAxis	511
10.84.4 Friends And Related Function Documentation	511
10.84.4.1 operator*	511
10.84.4.2 operator<<	511
10.84.5 Member Data Documentation	512
10.84.5.1 m	512
10.85gazebo::math::Matrix4 Class Reference	512
10.85.1 Detailed Description	513
10.85.2 Constructor & Destructor Documentation	514
10.85.2.1 Matrix4	514
10.85.2.2 Matrix4	514
10.85.2.3 Matrix4	514
10.85.2.4 ~Matrix4	514
10.85.3 Member Function Documentation	514
10.85.3.1 GetAsPose	514
10.85.3.2 GetEulerRotation	515
10.85.3.3 GetRotation	515
10.85.3.4 GetTranslation	515
10.85.3.5 Inverse	515
10.85.3.6 IsAffine	515
10.85.3.7 operator*	515
10.85.3.8 operator*	516
10.85.3.9 operator*	516
10.85.3.10operator=	516

10.85.3.11operator=	516
10.85.3.12operator==	517
10.85.3.13operator[]	517
10.85.3.14operator[]	517
10.85.3.15Set	517
10.85.3.16SetScale	518
10.85.3.17SetTranslate	518
10.85.3.18TransformAffine	518
10.85.4 Friends And Related Function Documentation	518
10.85.4.1 operator<<	519
10.85.5 Member Data Documentation	519
10.85.5.1 IDENTITY	519
10.85.5.2 m	519
10.85.5.3 ZERO	519
10.86gazebo::common::Mesh Class Reference	519
10.86.1 Detailed Description	521
10.86.2 Constructor & Destructor Documentation	521
10.86.2.1 Mesh	521
10.86.2.2 ~Mesh	521
10.86.3 Member Function Documentation	521
10.86.3.1 AddMaterial	521
10.86.3.2 AddSubMesh	521
10.86.3.3 Center	521
10.86.3.4 FillArrays	522
10.86.3.5 GenSphericalTexCoord	522
10.86.3.6 GetAABB	522
10.86.3.7 GetIndexCount	522
10.86.3.8 GetMaterial	522
10.86.3.9 GetMaterialCount	523
10.86.3.10GetMax	523
10.86.3.11GetMin	523
10.86.3.12GetName	523
10.86.3.13GetNormalCount	523
10.86.3.14GetPath	523
10.86.3.15GetSkeleton	524
10.86.3.16GetSubMesh	524
10.86.3.17GetSubMesh	524

10.86.3.18	GetSubMeshCount	524
10.86.3.19	GetTexCoordCount	524
10.86.3.20	GetVertexCount	525
10.86.3.21	HasSkeleton	525
10.86.3.22	RecalculateNormals	525
10.86.3.23	Scale	525
10.86.3.24	SetName	525
10.86.3.25	SetPath	525
10.86.3.26	SetScale	525
10.86.3.27	SetSkeleton	526
10.86.3.28	Translate	526
10.87	gazebo::common::MeshCSG Class Reference	526
10.87.1	Detailed Description	526
10.87.2	Member Enumeration Documentation	526
10.87.2.1	BooleanOperation	526
10.87.3	Constructor & Destructor Documentation	527
10.87.3.1	MeshCSG	527
10.87.3.2	~MeshCSG	527
10.87.4	Member Function Documentation	527
10.87.4.1	CreateBoolean	527
10.88	gazebo::common::MeshLoader Class Reference	527
10.88.1	Detailed Description	528
10.88.2	Constructor & Destructor Documentation	528
10.88.2.1	MeshLoader	528
10.88.2.2	~MeshLoader	528
10.88.3	Member Function Documentation	528
10.88.3.1	Load	528
10.89	gazebo::common::MeshManager Class Reference	529
10.89.1	Detailed Description	530
10.89.2	Member Function Documentation	530
10.89.2.1	AddMesh	530
10.89.2.2	CreateBox	530
10.89.2.3	CreateCamera	531
10.89.2.4	CreateCone	531
10.89.2.5	CreateCylinder	531
10.89.2.6	CreatePlane	531
10.89.2.7	CreatePlane	532

10.89.2.8 CreateSphere	532
10.89.2.9 CreateTube	532
10.89.2.10 GenSphericalTexCoord	532
10.89.2.11 GetMesh	532
10.89.2.12 GetMeshAABB	533
10.89.2.13 HasMesh	533
10.89.2.14 IsValidFilename	533
10.89.2.15 Load	533
10.90 gazebo::physics::MeshShape Class Reference	534
10.90.1 Detailed Description	535
10.90.2 Constructor & Destructor Documentation	535
10.90.2.1 MeshShape	535
10.90.2.2 ~MeshShape	535
10.90.3 Member Function Documentation	535
10.90.3.1 FillMsg	535
10.90.3.2 GetMeshURI	536
10.90.3.3 GetSize	536
10.90.3.4 Init	536
10.90.3.5 ProcessMsg	536
10.90.3.6 SetMesh	536
10.90.3.7 SetScale	536
10.90.3.8 Update	537
10.90.4 Member Data Documentation	537
10.90.4.1 mesh	537
10.90.4.2 submesh	537
10.91 gazebo::physics::Model Class Reference	537
10.91.1 Detailed Description	541
10.91.2 Constructor & Destructor Documentation	541
10.91.2.1 Model	541
10.91.2.2 ~Model	541
10.91.3 Member Function Documentation	541
10.91.3.1 AttachStaticModel	541
10.91.3.2 DetachStaticModel	542
10.91.3.3 FillMsg	542
10.91.3.4 Fini	542
10.91.3.5 GetAutoDisable	542
10.91.3.6 GetBoundingBox	542

10.91.3.7 GetGripper	543
10.91.3.8 GetGripperCount	543
10.91.3.9 GetJoint	543
10.91.3.10 GetJointController	543
10.91.3.11 GetJointCount	543
10.91.3.12 GetJoints	544
10.91.3.13 GetLink	544
10.91.3.14 GetLinks	544
10.91.3.15 GetPluginCount	544
10.91.3.16 GetRelativeAngularAccel	544
10.91.3.17 GetRelativeAngularVel	545
10.91.3.18 GetRelativeLinearAccel	545
10.91.3.19 GetRelativeLinearVel	545
10.91.3.20 GetSDF	545
10.91.3.21 GetSensorCount	545
10.91.3.22 GetWorldAngularAccel	546
10.91.3.23 GetWorldAngularVel	546
10.91.3.24 GetWorldLinearAccel	546
10.91.3.25 GetWorldLinearVel	546
10.91.3.26 Init	546
10.91.3.27 Load	546
10.91.3.28 LoadJoints	547
10.91.3.29 LoadPlugins	547
10.91.3.30 OnPoseChange	547
10.91.3.31 ProcessMsg	547
10.91.3.32 RemoveChild	547
10.91.3.33 Reset	547
10.91.3.34 SetAngularAccel	547
10.91.3.35 SetAngularVel	548
10.91.3.36 SetAutoDisable	548
10.91.3.37 SetCollideMode	548
10.91.3.38 SetEnabled	548
10.91.3.39 SetGravityMode	548
10.91.3.40 SetJointAnimation	549
10.91.3.41 SetJointPosition	549
10.91.3.42 SetJointPositions	549
10.91.3.43 SetLaserRetro	549

10.91.3.44	SetLinearAccel	549
10.91.3.45	SetLinearVel	550
10.91.3.46	SetLinkWorldPose	550
10.91.3.47	SetLinkWorldPose	550
10.91.3.48	SetScale	550
10.91.3.49	SetState	550
10.91.3.50	StopAnimation	551
10.91.3.51	Update	551
10.91.3.52	UpdateParameters	551
10.91.4	Member Data Documentation	551
10.91.4.1	attachedModels	551
10.91.4.2	attachedModelsOffset	551
10.91.4.3	jointPub	551
10.92	gazebo::common::ModelDatabase Class Reference	551
10.92.1	Detailed Description	553
10.93	gazebo::ModelPlugin Class Reference	553
10.93.1	Detailed Description	553
10.93.2	Constructor & Destructor Documentation	554
10.93.2.1	ModelPlugin	554
10.93.2.2	~ModelPlugin	554
10.93.3	Member Function Documentation	554
10.93.3.1	Init	554
10.93.3.2	Load	554
10.93.3.3	Reset	554
10.94	gazebo::physics::ModelState Class Reference	554
10.94.1	Detailed Description	556
10.94.2	Constructor & Destructor Documentation	556
10.94.2.1	ModelState	556
10.94.2.2	ModelState	557
10.94.2.3	ModelState	557
10.94.2.4	ModelState	557
10.94.2.5	~ModelState	557
10.94.3	Member Function Documentation	557
10.94.3.1	FillSDF	557
10.94.3.2	GetJointState	557
10.94.3.3	GetJointState	558
10.94.3.4	GetJointStateCount	558

10.94.3.5	GetJointStates	558
10.94.3.6	GetJointStates	559
10.94.3.7	GetLinkState	559
10.94.3.8	GetLinkStateCount	559
10.94.3.9	GetLinkStates	559
10.94.3.10	GetLinkStates	560
10.94.3.11	GetPose	560
10.94.3.12	HasJointState	560
10.94.3.13	HasLinkState	560
10.94.3.14	IsZero	560
10.94.3.15	Load	561
10.94.3.16	Load	561
10.94.3.17	operator+	561
10.94.3.18	operator-	561
10.94.3.19	operator=	562
10.94.3.20	SetRealTime	562
10.94.3.21	SetSimTime	562
10.94.3.22	SetWallTime	562
10.94.4	Friends And Related Function Documentation	562
10.94.4.1	operator<<	563
10.95	gazebo::common::MouseEvent Class Reference	563
10.95.1	Detailed Description	564
10.95.2	Member Enumeration Documentation	564
10.95.2.1	Buttons	564
10.95.2.2	EventType	564
10.95.3	Constructor & Destructor Documentation	565
10.95.3.1	MouseEvent	565
10.95.4	Member Data Documentation	565
10.95.4.1	alt	565
10.95.4.2	button	565
10.95.4.3	buttons	565
10.95.4.4	control	565
10.95.4.5	dragging	565
10.95.4.6	moveScale	565
10.95.4.7	pos	565
10.95.4.8	pressPos	565
10.95.4.9	prevPos	565

10.95.4.10scroll	566
10.95.4.11shift	566
10.95.4.12type	566
10.96gazebo::rendering::MovableText Class Reference	566
10.96.1 Detailed Description	568
10.96.2 Member Enumeration Documentation	568
10.96.2.1 HorizAlign	568
10.96.2.2 VertAlign	568
10.96.3 Constructor & Destructor Documentation	568
10.96.3.1 MovableText	568
10.96.3.2 ~MovableText	568
10.96.4 Member Function Documentation	569
10.96.4.1 _setupGeometry	569
10.96.4.2 _updateColors	569
10.96.4.3 GetAABB	569
10.96.4.4 GetBaseline	569
10.96.4.5 getBoundingRadius	569
10.96.4.6 GetCharHeight	569
10.96.4.7 GetColor	569
10.96.4.8 GetFont	569
10.96.4.9 getLights	569
10.96.4.10getMaterial	569
10.96.4.11getRenderOperation	570
10.96.4.12GetShowOnTop	570
10.96.4.13GetSpaceWidth	570
10.96.4.14getSquaredViewDepth	570
10.96.4.15GetText	570
10.96.4.16getWorldTransforms	570
10.96.4.17Load	570
10.96.4.18SetBaseline	570
10.96.4.19SetCharHeight	571
10.96.4.20SetColor	571
10.96.4.21SetFontName	571
10.96.4.22SetShowOnTop	571
10.96.4.23SetSpaceWidth	571
10.96.4.24SetText	571
10.96.4.25SetTextAlignment	572

10.96.4.26Update	572
10.96.4.27visitRenderables	572
10.97gazebo::msgs::MsgFactory Class Reference	572
10.97.1 Detailed Description	572
10.97.2 Member Function Documentation	573
10.97.2.1 GetMsgTypes	573
10.97.2.2 NewMsg	573
10.97.2.3 RegisterMsg	573
10.98gazebo::sensors::MultiCameraSensor Class Reference	573
10.98.1 Detailed Description	575
10.98.2 Constructor & Destructor Documentation	575
10.98.2.1 MultiCameraSensor	575
10.98.2.2 ~MultiCameraSensor	575
10.98.3 Member Function Documentation	575
10.98.3.1 Fini	575
10.98.3.2 GetCamera	575
10.98.3.3 GetCameraCount	576
10.98.3.4 GetImageData	576
10.98.3.5 GetImageHeight	576
10.98.3.6 GetImageWidth	576
10.98.3.7 GetTopic	577
10.98.3.8 Init	577
10.98.3.9 IsActive	577
10.98.3.10Load	577
10.98.3.11SaveFrame	577
10.98.3.12UpdateImpl	578
10.99gazebo::physics::MultiRayShape Class Reference	578
10.99.1 Detailed Description	581
10.99.2 Constructor & Destructor Documentation	581
10.99.2.1 MultiRayShape	581
10.99.2.2 ~MultiRayShape	581
10.99.3 Member Function Documentation	581
10.99.3.1 AddRay	581
10.99.3.2 ConnectNewLaserScans	582
10.99.3.3 DisconnectNewLaserScans	582
10.99.3.4 FillMsg	582
10.99.3.5 GetFiducial	582

10.99.3.6	GetMaxAngle	583
10.99.3.7	GetMaxRange	583
10.99.3.8	GetMinAngle	583
10.99.3.9	GetMinRange	583
10.99.3.10	GetRange	583
10.99.3.11	GetResRange	584
10.99.3.12	GetRetro	584
10.99.3.13	GetSampleCount	584
10.99.3.14	GetScanResolution	584
10.99.3.15	GetVerticalMaxAngle	584
10.99.3.16	GetVerticalMinAngle	584
10.99.3.17	GetVerticalSampleCount	585
10.99.3.18	GetVerticalScanResolution	585
10.99.3.19	Init	585
10.99.3.20	ProcessMsg	585
10.99.3.21	SetScale	585
10.99.3.22	Update	585
10.99.3.23	UpdateRays	586
10.99.4	Member Data Documentation	586
10.99.4.1	horzElem	586
10.99.4.2	newLaserScans	586
10.99.4.3	offset	586
10.99.4.4	rangeElem	586
10.99.4.5	rayElem	586
10.99.4.6	rays	586
10.99.4.7	scanElem	586
10.99.4.8	vertElem	586
10.100	gazebo::transport::Node Class Reference	587
10.100.1	Detailed Description	588
10.100.2	Constructor & Destructor Documentation	588
10.100.2.1	Node	588
10.100.2.2	~Node	588
10.100.3	Member Function Documentation	589
10.100.3.1	Advertise	589
10.100.3.2	DecodeTopicName	589
10.100.3.3	EncodeTopicName	589
10.100.3.4	Fini	589

10.100.3.5	GetId	590
10.100.3.6	GetMsgType	590
10.100.3.7	GetTopicNamespace	590
10.100.3.8	HandleData	590
10.100.3.9	HandleMessage	590
10.100.3.10	HasLatchedSubscriber	591
10.100.3.11	hit	591
10.100.3.12	InsertLatchedMsg	591
10.100.3.13	InsertLatchedMsg	591
10.100.3.14	ProcessIncoming	591
10.100.3.15	ProcessPublishers	592
10.100.3.16	Publish	592
10.100.3.17	RemoveCallback	592
10.100.3.18	Subscribe	592
10.100.3.19	Subscribe	592
10.100.3.20	Subscribe	593
10.100.3.21	Subscribe	593
10.101	gazebo::common::NodeAnimation Class Reference	593
10.101.1	Detailed Description	594
10.101.2	Constructor & Destructor Documentation	595
10.101.2.1	NodeAnimation	595
10.101.2.2	~NodeAnimation	595
10.101.3	Member Function Documentation	595
10.101.3.1	AddKeyFrame	595
10.101.3.2	AddKeyFrame	595
10.101.3.3	GetFrameAt	595
10.101.3.4	GetFrameCount	595
10.101.3.5	GetKeyFrame	596
10.101.3.6	GetKeyFrame	596
10.101.3.7	GetLength	596
10.101.3.8	GetName	596
10.101.3.9	GetTimeAtX	596
10.101.3.10	Scale	597
10.101.3.11	SetName	597
10.101.4	Member Data Documentation	597
10.101.4.1	keyFrames	597
10.101.4.2	length	597

10.101.4.3name	597
10.102 gazebo::common::NodeAssignment Struct Reference	597
10.102.1Detailed Description	598
10.102.2Member Data Documentation	598
10.102.2.1nodeIndex	598
10.102.2.2vertexIndex	598
10.102.2.3weight	598
10.103 gazebo::common::NodeTransform Class Reference	598
10.103.1Detailed Description	599
10.103.2Member Enumeration Documentation	600
10.103.2.1TransformType	600
10.103.3Constructor & Destructor Documentation	600
10.103.3.1NodeTransform	600
10.103.3.2NodeTransform	600
10.103.3.3~NodeTransform	600
10.103.4Member Function Documentation	600
10.103.4.1Get	600
10.103.4.2GetSID	601
10.103.4.3GetType	601
10.103.4.4operator()	601
10.103.4.5operator*	601
10.103.4.6operator*	601
10.103.4.7PrintSource	602
10.103.4.8RecalculateMatrix	602
10.103.4.9Set	602
10.103.4.10SetComponent	602
10.103.4.11SetSID	602
10.103.4.12SetSourceValues	602
10.103.4.13SetSourceValues	602
10.103.4.14SetSourceValues	602
10.103.4.15SetType	603
10.103.5Member Data Documentation	603
10.103.5.1sid	603
10.103.5.2source	603
10.103.5.3transform	603
10.103.5.4type	603
10.104 gazebo::sensors::Noise Class Reference	603

10.104.1	Detailed Description	604
10.104.2	Member Enumeration Documentation	604
10.104.2.1	NoiseType	604
10.104.3	Constructor & Destructor Documentation	604
10.104.3.1	Noise	604
10.104.3.2	~Noise	604
10.104.4	Member Function Documentation	604
10.104.4.1	Apply	604
10.104.4.2	GetBias	605
10.104.4.3	GetMean	605
10.104.4.4	GetNoiseType	605
10.104.4.5	GetStdDev	605
10.104.4.6	Load	605
10.105	gazebo::common::NumericAnimation Class Reference	606
10.105.1	Detailed Description	606
10.105.2	Constructor & Destructor Documentation	606
10.105.2.1	NumericAnimation	606
10.105.2.2	~NumericAnimation	607
10.105.3	Member Function Documentation	607
10.105.3.1	CreateKeyFrame	607
10.105.3.2	GetInterpolatedKeyFrame	607
10.106	gazebo::common::NumericKeyFrame Class Reference	607
10.106.1	Detailed Description	608
10.106.2	Constructor & Destructor Documentation	608
10.106.2.1	NumericKeyFrame	608
10.106.2.2	~NumericKeyFrame	609
10.106.3	Member Function Documentation	609
10.106.3.1	GetValue	609
10.106.3.2	SetValue	609
10.106.4	Member Data Documentation	609
10.106.4.1	value	609
10.107	gazebo::util::OpenAL Class Reference	609
10.107.1	Detailed Description	610
10.107.2	Member Function Documentation	610
10.107.2.1	CreateSink	610
10.107.2.2	CreateSource	611
10.107.2.3	Finis	611

10.107.2.4	Load	611
10.108	gazebo::util::OpenALSink Class Reference	611
10.108.1	Detailed Description	611
10.108.2	Constructor & Destructor Documentation	612
10.108.2.1	OpenALSink	612
10.108.2.2	~OpenALSink	612
10.108.3	Member Function Documentation	612
10.108.3.1	SetPose	612
10.108.3.2	SetVelocity	612
10.109	gazebo::util::OpenALSource Class Reference	612
10.109.1	Detailed Description	613
10.109.2	Constructor & Destructor Documentation	613
10.109.2.1	OpenALSource	613
10.109.2.2	~OpenALSource	614
10.109.3	Member Function Documentation	614
10.109.3.1	FillBufferFromFile	614
10.109.3.2	FillBufferFromPCM	614
10.109.3.3	GetCollisionNames	614
10.109.3.4	GetOnContact	614
10.109.3.5	HasCollisionName	615
10.109.3.6	IsPlaying	615
10.109.3.7	Load	615
10.109.3.8	Pause	615
10.109.3.9	Play	615
10.109.3.10	Rewind	615
10.109.3.11	SetGain	615
10.109.3.12	SetLoop	616
10.109.3.13	SetPitch	616
10.109.3.14	SetPose	616
10.109.3.15	SetVelocity	616
10.109.3.16	Stop	617
10.110	gazebo::rendering::OrbitViewController Class Reference	617
10.110.1	Detailed Description	618
10.110.2	Constructor & Destructor Documentation	618
10.110.2.1	OrbitViewController	618
10.110.2.2	~OrbitViewController	618
10.110.3	Member Function Documentation	618

10.110.3.1	GetFocalPoint	618
10.110.3.2	GetTypeString	619
10.110.3.3	HandleKeyPressEvent	619
10.110.3.4	HandleKeyReleaseEvent	619
10.110.3.5	HandleMouseEvent	619
10.110.3.6	Init	619
10.110.3.7	Init	619
10.110.3.8	SetDistance	620
10.110.3.9	SetFocalPoint	620
10.110.3.10	Update	620
10.110	gazebo::common::ParamT< T > Class Template Reference	620
10.110	gazebo::physics::PhysicsEngine Class Reference	620
10.112.1	Detailed Description	624
10.112.2	Constructor & Destructor Documentation	624
10.112.2.1	PhysicsEngine	624
10.112.2.2	~PhysicsEngine	624
10.112.3	Member Function Documentation	624
10.112.3.1	CreateCollision	624
10.112.3.2	CreateCollision	624
10.112.3.3	CreateJoint	625
10.112.3.4	CreateLink	625
10.112.3.5	CreateModel	625
10.112.3.6	CreateShape	625
10.112.3.7	DebugPrint	626
10.112.3.8	Finis	626
10.112.3.9	GetAutoDisableFlag	626
10.112.3.10	GetContactManager	626
10.112.3.11	GetContactMaxCorrectingVel	626
10.112.3.12	GetContactSurfaceLayer	626
10.112.3.13	GetGravity	627
10.112.3.14	GetMaxContacts	627
10.112.3.15	GetMaxStepSize	627
10.112.3.16	GetParam	627
10.112.3.17	GetPhysicsUpdateMutex	627
10.112.3.18	GetRealTimeUpdateRate	627
10.112.3.19	GetSORPGSIters	628
10.112.3.20	GetSORPGSPreconIters	628

10.112.3.20	GetSORPGSW	628
10.112.3.20	GetTargetRealTimeFactor	628
10.112.3.23	GetType	628
10.112.3.24	GetUpdatePeriod	629
10.112.3.25	GetWorldCFM	629
10.112.3.26	GetWorldERP	629
10.112.3.27	Init	629
10.112.3.28	InitForThread	629
10.112.3.29	Load	629
10.112.3.30	OnPhysicsMsg	630
10.112.3.30	OnRequest	630
10.112.3.32	Reset	630
10.112.3.33	SetAutoDisableFlag	630
10.112.3.33	SetContactMaxCorrectingVel	630
10.112.3.35	SetContactSurfaceLayer	630
10.112.3.36	SetGravity	631
10.112.3.37	SetMaxContacts	631
10.112.3.38	SetMaxStepSize	631
10.112.3.39	SetParam	631
10.112.3.40	SetRealTimeUpdateRate	631
10.112.3.41	SetSeed	632
10.112.3.42	SetSORPGSIters	632
10.112.3.43	SetSORPGSPreconIters	632
10.112.3.43	SetSORPGSW	632
10.112.3.45	SetTargetRealTimeFactor	632
10.112.3.46	SetWorldCFM	633
10.112.3.47	SetWorldERP	633
10.112.3.48	UpdateCollision	633
10.112.3.49	UpdatePhysics	633
10.112.4	Member Data Documentation	633
10.112.4.1	contactManager	633
10.112.4.2	maxStepSize	633
10.112.4.3	node	633
10.112.4.4	physicsSub	634
10.112.4.5	physicsUpdateMutex	634
10.112.4.6	realTimeUpdateRate	634
10.112.4.7	requestSub	634

10.112.4.8	responsePub	634
10.112.4.9	sdf	634
10.112.4.10	targetRealTimeFactor	634
10.112.4.11	World	634
10.113	gazebo::physics::PhysicsFactory Class Reference	634
10.113.1	Detailed Description	635
10.113.2	Member Function Documentation	635
10.113.2.1	IsRegistered	635
10.113.2.2	NewPhysicsEngine	635
10.113.2.3	RegisterAll	635
10.113.2.4	RegisterPhysicsEngine	635
10.114	gazebo::common::PID Class Reference	636
10.114.1	Detailed Description	637
10.114.2	Constructor & Destructor Documentation	637
10.114.2.1	PID	637
10.114.2.2	~PID	637
10.114.3	Member Function Documentation	637
10.114.3.1	GetCmd	637
10.114.3.2	GetErrors	637
10.114.3.3	init	638
10.114.3.4	operator=	638
10.114.3.5	Reset	638
10.114.3.6	SetCmd	638
10.114.3.7	SetCmdMax	638
10.114.3.8	SetCmdMin	639
10.114.3.9	SetDGain	639
10.114.3.10	SetIGain	639
10.114.3.11	SetIMax	639
10.114.3.12	SetIMin	639
10.114.3.13	SetPGain	639
10.114.3.14	update	640
10.115	gazebo::math::Plane Class Reference	640
10.115.1	Detailed Description	641
10.115.2	Constructor & Destructor Documentation	641
10.115.2.1	Plane	641
10.115.2.2	Plane	641
10.115.2.3	Plane	641

10.115.2.4	~Plane	641
10.115.3	Member Function Documentation	641
10.115.3.1	Distance	641
10.115.3.2	operator=	642
10.115.3.3	Set	642
10.115.4	Member Data Documentation	642
10.115.4.1	id	642
10.115.4.2	normal	642
10.115.4.3	size	642
10.116	Gazebo::physics::PlaneShape Class Reference	642
10.116.1	Detailed Description	644
10.116.2	Constructor & Destructor Documentation	644
10.116.2.1	PlaneShape	644
10.116.2.2	~PlaneShape	644
10.116.3	Member Function Documentation	644
10.116.3.1	CreatePlane	644
10.116.3.2	FillMsg	644
10.116.3.3	GetNormal	645
10.116.3.4	GetSize	645
10.116.3.5	Init	645
10.116.3.6	ProcessMsg	645
10.116.3.7	SetAltitude	645
10.116.3.8	SetNormal	645
10.116.3.9	SetScale	646
10.116.3.10	SetSize	646
10.117	Gazebo::PluginT< T > Class Template Reference	646
10.117.1	Detailed Description	647
10.117.2	Member Typedef Documentation	647
10.117.2.1	TPtr	647
10.117.3	Constructor & Destructor Documentation	647
10.117.3.1	PluginT	647
10.117.3.2	~PluginT	647
10.117.4	Member Function Documentation	647
10.117.4.1	Create	647
10.117.4.2	GetFilename	648
10.117.4.3	GetHandle	648
10.117.4.4	GetType	648

10.117.5	Member Data Documentation	648
10.117.5.1	filename	648
10.117.5.2	handle	648
10.117.5.3	type	648
10.118	gazebo::math::Pose Class Reference	648
10.118.1	Detailed Description	650
10.118.2	Constructor & Destructor Documentation	650
10.118.2.1	Pose	650
10.118.2.2	Pose	651
10.118.2.3	Pose	651
10.118.2.4	Pose	651
10.118.2.5	~Pose	651
10.118.3	Member Function Documentation	651
10.118.3.1	CoordPoseSolve	651
10.118.3.2	CoordPositionAdd	651
10.118.3.3	CoordPositionAdd	652
10.118.3.4	CoordPositionSub	652
10.118.3.5	CoordRotationAdd	652
10.118.3.6	CoordRotationSub	653
10.118.3.7	Correct	653
10.118.3.8	GetInverse	653
10.118.3.9	IsFinite	653
10.118.3.10	operator!=	653
10.118.3.11	operator*	653
10.118.3.12	operator+	654
10.118.3.13	operator+=	654
10.118.3.14	operator-	654
10.118.3.15	operator-	654
10.118.3.16	operator-=	655
10.118.3.17	operator=	655
10.118.3.18	operator==	655
10.118.3.19	Reset	655
10.118.3.20	RotatePositionAboutOrigin	655
10.118.3.21	Round	656
10.118.3.22	Set	656
10.118.3.23	Set	656
10.118.3.24	Set	656

10.118.4	Friends And Related Function Documentation	656
10.118.4.1	operator<<	656
10.118.4.2	operator>>	657
10.118.5	Member Data Documentation	657
10.118.5.1	pos	657
10.118.5.2	rot	657
10.118.5.3	Zero	657
10.119	gazebo::common::PoseAnimation Class Reference	657
10.119.1	Detailed Description	658
10.119.2	Constructor & Destructor Documentation	658
10.119.2.1	PoseAnimation	658
10.119.2.2	~PoseAnimation	659
10.119.3	Member Function Documentation	659
10.119.3.1	BuildInterpolationSplines	659
10.119.3.2	CreateKeyFrame	659
10.119.3.3	GetInterpolatedKeyFrame	659
10.119.3.4	GetInterpolatedKeyFrame	659
10.120	gazebo::common::PoseKeyFrame Class Reference	660
10.120.1	Detailed Description	660
10.120.2	Constructor & Destructor Documentation	661
10.120.2.1	PoseKeyFrame	661
10.120.2.2	~PoseKeyFrame	661
10.120.3	Member Function Documentation	661
10.120.3.1	GetRotation	661
10.120.3.2	GetTranslation	661
10.120.3.3	SetRotation	661
10.120.3.4	SetTranslation	661
10.120.4	Member Data Documentation	662
10.120.4.1	rotate	662
10.120.4.2	translate	662
10.121	gazebo::rendering::Projector Class Reference	662
10.121.1	Detailed Description	662
10.121.2	Constructor & Destructor Documentation	663
10.121.2.1	Projector	663
10.121.2.2	~Projector	663
10.121.3	Member Function Documentation	663
10.121.3.1	GetParent	663

10.121.3.2	Load	663
10.121.3.3	Load	663
10.121.3.4	Load	663
10.121.3.5	setEnabled	664
10.121.3.6	setTexture	664
10.121.3.7	Toggle	664
10.122	gazebo::transport::Publication Class Reference	664
10.122.1	Detailed Description	665
10.122.2	Constructor & Destructor Documentation	665
10.122.2.1	Publication	665
10.122.2.2	~Publication	666
10.122.3	Member Function Documentation	666
10.122.3.1	AddPublisher	666
10.122.3.2	AddSubscription	666
10.122.3.3	AddSubscription	666
10.122.3.4	AddTransport	666
10.122.3.5	GetCallbackCount	666
10.122.3.6	GetLocallyAdvertised	667
10.122.3.7	GetMsgType	667
10.122.3.8	GetNodeCount	667
10.122.3.9	GetRemoteSubscriptionCount	667
10.122.3.10	GetTransportCount	667
10.122.3.11	HasTransport	667
10.122.3.12	LocalPublish	668
10.122.3.13	Publish	668
10.122.3.14	RemoveSubscription	668
10.122.3.15	RemoveSubscription	668
10.122.3.16	RemoveTransport	668
10.122.3.17	SetLocallyAdvertised	669
10.123	gazebo::transport::PublicationTransport Class Reference	669
10.123.1	Detailed Description	669
10.123.2	Constructor & Destructor Documentation	669
10.123.2.1	PublicationTransport	669
10.123.2.2	~PublicationTransport	670
10.123.3	Member Function Documentation	670
10.123.3.1	AddCallback	670
10.123.3.2	Fin	670

10.123.3.3	GetConnection	670
10.123.3.4	GetMsgType	670
10.123.3.5	GetTopic	670
10.123.3.6	Init	671
10.124	Gazebo::transport::Publisher Class Reference	671
10.124.1	Detailed Description	672
10.124.2	Constructor & Destructor Documentation	672
10.124.2.1	Publisher	672
10.124.2.2	~Publisher	672
10.124.3	Member Function Documentation	672
10.124.3.1	GetMsgType	672
10.124.3.2	GetOutgoingCount	672
10.124.3.3	GetPrevMsg	672
10.124.3.4	GetPrevMsgPtr	673
10.124.3.5	GetTopic	673
10.124.3.6	HasConnections	673
10.124.3.7	Publish	673
10.124.3.8	Publish	673
10.124.3.9	SendMessage	674
10.124.3.10	SetNode	674
10.124.3.11	SetPublication	674
10.124.3.12	WaitForConnection	674
10.124.3.13	WaitForConnection	674
10.125	QuadNode Class Reference	674
10.126	Gazebo::math::Quaternion Class Reference	675
10.126.1	Detailed Description	677
10.126.2	Constructor & Destructor Documentation	678
10.126.2.1	Quaternion	678
10.126.2.2	Quaternion	678
10.126.2.3	Quaternion	678
10.126.2.4	Quaternion	678
10.126.2.5	Quaternion	678
10.126.2.6	Quaternion	678
10.126.2.7	~Quaternion	679
10.126.3	Member Function Documentation	679
10.126.3.1	Correct	679
10.126.3.2	Dot	679

10.126.3.3EulerToQuaternion	679
10.126.3.4EulerToQuaternion	679
10.126.3.5GetAsAxis	680
10.126.3.6GetAsEuler	680
10.126.3.7GetAsMatrix3	680
10.126.3.8GetAsMatrix4	680
10.126.3.9GetExp	680
10.126.3.10GetInverse	680
10.126.3.11GetLog	681
10.126.3.12GetPitch	681
10.126.3.13GetRoll	681
10.126.3.14GetXAxis	681
10.126.3.15GetYaw	681
10.126.3.16GetYAxis	681
10.126.3.17GetZAxis	682
10.126.3.18Invert	682
10.126.3.19IsFinite	682
10.126.3.20Normalize	682
10.126.3.21operator!=	682
10.126.3.22operator*	682
10.126.3.23operator*	683
10.126.3.24operator*	683
10.126.3.25operator*=	683
10.126.3.26operator+	683
10.126.3.27operator+=	684
10.126.3.28operator-	684
10.126.3.29operator-	684
10.126.3.30operator-=	684
10.126.3.31operator=	684
10.126.3.32operator==	685
10.126.3.33RotateVector	685
10.126.3.34RotateVectorReverse	685
10.126.3.35Round	685
10.126.3.36Scale	686
10.126.3.37Set	686
10.126.3.38SetFromAxis	686
10.126.3.39SetFromAxis	686

10.126.3.4	GetFromEuler	686
10.126.3.4	SetFromEuler	687
10.126.3.4	GetTolIdentity	687
10.126.3.4	Slerp	687
10.126.3.4	Squad	687
10.126.4	Friends And Related Function Documentation	687
10.126.4.1	operator<<	687
10.126.4.2	operator>>	688
10.126.5	Member Data Documentation	688
10.126.5.1	w	688
10.126.5.2	x	688
10.126.5.3	y	688
10.126.5.4	z	688
10.127	gazebo::math::Rand Class Reference	688
10.127.1	Detailed Description	689
10.127.2	Member Function Documentation	689
10.127.2.1	GetDbfNormal	689
10.127.2.2	GetDbfUniform	689
10.127.2.3	GetIntNormal	689
10.127.2.4	GetIntUniform	690
10.127.2.5	GetSeed	690
10.127.2.6	SetSeed	690
10.128	gazebo::transport::RawCallbackHelper Class Reference	690
10.128.1	Detailed Description	691
10.128.2	Constructor & Destructor Documentation	691
10.128.2.1	RawCallbackHelper	691
10.128.3	Member Function Documentation	692
10.128.3.1	GetMsgType	692
10.128.3.2	HandleData	692
10.128.3.3	HandleMessage	692
10.128.3.4	IsLocal	692
10.129	gazebo::sensors::RaySensor Class Reference	693
10.129.1	Detailed Description	695
10.129.2	Constructor & Destructor Documentation	695
10.129.2.1	RaySensor	695
10.129.2.2	~RaySensor	695
10.129.3	Member Function Documentation	695

10.129.3.1	Finis	695
10.129.3.2	GetAngleMax	695
10.129.3.3	GetAngleMin	695
10.129.3.4	GetAngleResolution	695
10.129.3.5	GetFiducial	696
10.129.3.6	GetLaserShape	696
10.129.3.7	GetRange	696
10.129.3.8	GetRangeCount	696
10.129.3.9	GetRangeMax	697
10.129.3.10	GetRangeMin	697
10.129.3.11	GetRangeResolution	697
10.129.3.12	GetRanges	697
10.129.3.13	GetRayCount	697
10.129.3.14	GetRetro	697
10.129.3.15	GetTopic	698
10.129.3.16	GetVerticalAngleMax	698
10.129.3.17	GetVerticalAngleMin	698
10.129.3.18	GetVerticalRangeCount	698
10.129.3.19	GetVerticalRayCount	699
10.129.3.20	hit	699
10.129.3.21	IsActive	699
10.129.3.22	load	699
10.129.3.23	updateImpl	699
10.130	gazebo::physics::RayShape Class Reference	700
10.130.1	Detailed Description	702
10.130.2	Constructor & Destructor Documentation	702
10.130.2.1	RayShape	702
10.130.2.2	RayShape	702
10.130.2.3	~RayShape	702
10.130.3	Member Function Documentation	702
10.130.3.1	FillMsg	702
10.130.3.2	GetFiducial	702
10.130.3.3	GetGlobalPoints	703
10.130.3.4	GetIntersection	703
10.130.3.5	GetLength	703
10.130.3.6	GetRelativePoints	703
10.130.3.7	GetRetro	703

10.130.3.8	init	704
10.130.3.9	ProcessMsg	704
10.130.3.10	SetFiducial	704
10.130.3.11	SetLength	704
10.130.3.12	SetPoints	704
10.130.3.13	SetRetro	704
10.130.3.14	SetScale	705
10.130.3.15	Update	705
10.130.4	Member Data Documentation	705
10.130.4.1	contactFiducial	705
10.130.4.2	contactLen	705
10.130.4.3	contactRetro	705
10.130.4.4	globalEndPos	705
10.130.4.5	globalStartPos	705
10.130.4.6	relativeEndPos	705
10.130.4.7	relativeStartPos	705
10.131	Gazebo::rendering::RenderEngine Class Reference	706
10.131.1	Detailed Description	707
10.131.2	Member Enumeration Documentation	707
10.131.2.1	RenderPathType	707
10.131.3	Member Function Documentation	708
10.131.3.1	AddResourcePath	708
10.131.3.2	CreateScene	708
10.131.3.3	Finis	708
10.131.3.4	GetRenderPathType	708
10.131.3.5	GetScene	708
10.131.3.6	GetScene	708
10.131.3.7	GetSceneCount	709
10.131.3.8	GetWindowManager	709
10.131.3.9	init	709
10.131.3.10	Load	709
10.131.3.11	RemoveScene	709
10.131.4	Member Data Documentation	709
10.131.4.1	dummyContext	709
10.131.4.2	dummyDisplay	710
10.131.4.3	dummyWindowId	710
10.131.4.4	root	710

10.132	gazebo::sensors::RFIDSensor Class Reference	710
10.132.1	Detailed Description	711
10.132.2	Constructor & Destructor Documentation	711
10.132.2.1	RFIDSensor	711
10.132.2.2	~RFIDSensor	711
10.132.3	Member Function Documentation	711
10.132.3.1	AddTag	711
10.132.3.2	Finis	711
10.132.3.3	Init	711
10.132.3.4	Load	712
10.132.3.5	Load	712
10.132.3.6	UpdateImpl	712
10.133	gazebo::sensors::RFIDTag Class Reference	712
10.133.1	Detailed Description	713
10.133.2	Constructor & Destructor Documentation	714
10.133.2.1	RFIDTag	714
10.133.2.2	~RFIDTag	714
10.133.3	Member Function Documentation	714
10.133.3.1	Finis	714
10.133.3.2	GetTagPose	714
10.133.3.3	Init	714
10.133.3.4	Load	714
10.133.3.5	Load	714
10.133.3.6	UpdateImpl	714
10.134	gazebo::rendering::RFIDTagVisual Class Reference	715
10.134.1	Detailed Description	716
10.134.2	Constructor & Destructor Documentation	716
10.134.2.1	RFIDTagVisual	716
10.134.2.2	~RFIDTagVisual	716
10.135	gazebo::rendering::RFIDVisual Class Reference	716
10.135.1	Detailed Description	717
10.135.2	Constructor & Destructor Documentation	717
10.135.2.1	RFIDVisual	717
10.135.2.2	~RFIDVisual	718
10.136	Road Class Reference	718
10.136.1	Detailed Description	718
10.137	gazebo::physics::Road Class Reference	718

10.137.1	Detailed Description	719
10.137.2	Constructor & Destructor Documentation	719
10.137.2.1	Road	719
10.137.2.2	~Road	719
10.137.3	Member Function Documentation	719
10.137.3.1	Init	719
10.137.3.2	Load	719
10.138	Gazebo::rendering::Road2d Class Reference	720
10.138.1	Constructor & Destructor Documentation	720
10.138.1.1	Road2d	720
10.138.1.2	~Road2d	720
10.138.2	Member Function Documentation	720
10.138.2.1	Load	720
10.139	Gazebo::math::RotationSpline Class Reference	720
10.139.1	Detailed Description	721
10.139.2	Constructor & Destructor Documentation	721
10.139.2.1	RotationSpline	721
10.139.2.2	~RotationSpline	721
10.139.3	Member Function Documentation	722
10.139.3.1	AddPoint	722
10.139.3.2	Clear	722
10.139.3.3	GetNumPoints	722
10.139.3.4	GetPoint	722
10.139.3.5	Interpolate	722
10.139.3.6	Interpolate	723
10.139.3.7	RecalcTangents	723
10.139.3.8	SetAutoCalculate	723
10.139.3.9	UpdatePoint	723
10.139.4	Member Data Documentation	724
10.139.4.1	autoCalc	724
10.139.4.2	points	724
10.139.4.3	tangents	724
10.140	Gazebo::rendering::RTShaderSystem Class Reference	724
10.140.1	Detailed Description	725
10.140.2	Member Enumeration Documentation	726
10.140.2.1	LightingModel	726
10.140.3	Member Function Documentation	726

10.140.3.1	AddScene	726
10.140.3.2	ApplyShadows	726
10.140.3.3	AttachEntity	726
10.140.3.4	AttachViewport	726
10.140.3.5	Clear	727
10.140.3.6	DetachEntity	727
10.140.3.7	DetachViewport	727
10.140.3.8	Fin	727
10.140.3.9	GenerateShaders	727
10.140.3.10	GetPSSMShadowCameraSetup	727
10.140.3.11	hit	727
10.140.3.12	RemoveScene	728
10.140.3.13	RemoveShadows	728
10.140.3.14	SetPerPixelLighting	728
10.140.3.15	UpdateShaders	728
10.141	gazebo::rendering::Scene Class Reference	728
10.141.1	Detailed Description	732
10.141.2	Member Enumeration Documentation	732
10.141.2.1	SkyXMode	732
10.141.3	Constructor & Destructor Documentation	733
10.141.3.1	Scene	733
10.141.3.2	~Scene	733
10.141.4	Member Function Documentation	733
10.141.4.1	AddVisual	733
10.141.4.2	Clear	733
10.141.4.3	CloneVisual	733
10.141.4.4	CreateCamera	733
10.141.4.5	CreateDepthCamera	734
10.141.4.6	CreateGpuLaser	734
10.141.4.7	CreateGrid	734
10.141.4.8	CreateUserCamera	734
10.141.4.9	DrawLine	735
10.141.4.10	GetAmbientColor	735
10.141.4.11	GetBackgroundColor	735
10.141.4.12	GetCamera	735
10.141.4.13	GetCamera	735
10.141.4.14	GetCameraCount	736

10.141.4.16	GetFirstContact	736
10.141.4.16	GetGrid	736
10.141.4.16	GetGridCount	736
10.141.4.16	GetHeightBelowPoint	737
10.141.4.16	GetHeightmap	737
10.141.4.20	GetId	737
10.141.4.20	GetIdString	737
10.141.4.20	GetInitialized	737
10.141.4.23	GetLight	737
10.141.4.23	GetLight	738
10.141.4.25	GetLightCount	738
10.141.4.26	GetManager	738
10.141.4.27	GetModelVisualAt	738
10.141.4.28	GetName	739
10.141.4.29	GetSelectedVisual	739
10.141.4.30	GetShadowsEnabled	739
10.141.4.30	GetShowClouds	739
10.141.4.30	GetSimTime	739
10.141.4.33	GetUserCamera	739
10.141.4.33	GetUserCameraCount	740
10.141.4.35	GetVisual	740
10.141.4.36	GetVisual	740
10.141.4.37	GetVisualAt	740
10.141.4.38	GetVisualAt	741
10.141.4.39	GetVisualBelow	741
10.141.4.40	GetVisualCount	741
10.141.4.40	GetVisualsBelowPoint	741
10.141.4.40	GetWorldVisual	742
10.141.4.40	Hit	742
10.141.4.44	Load	742
10.141.4.45	Load	742
10.141.4.46	PreRender	742
10.141.4.47	PrintSceneGraph	742
10.141.4.48	RemoveCamera	742
10.141.4.49	RemoveVisual	742
10.141.4.50	SelectVisual	743
10.141.4.53	SetAmbientColor	743

10.141.4.52	SetBackgroundColor	743
10.141.4.53	SetFog	743
10.141.4.53	SetGrid	743
10.141.4.55	SetShadowsEnabled	744
10.141.4.56	SetSkyXMode	744
10.141.4.57	SetTransparent	744
10.141.4.58	SetVisible	744
10.141.4.59	SetWireframe	744
10.141.4.60	ShowClouds	744
10.141.4.63	ShowCollisions	745
10.141.4.62	ShowCOMs	745
10.141.4.63	ShowContacts	745
10.141.4.63	ShowJoints	745
10.141.4.65	SnapVisualToNearestBelow	745
10.141.4.66	StripSceneName	745
10.141.5	Member Data Documentation	746
10.141.5.1	skyx	746
10.142	gazebo::physics::ScrewJoint< T > Class Template Reference	746
10.142.1	Detailed Description	747
10.142.2	Constructor & Destructor Documentation	747
10.142.2.1	ScrewJoint	747
10.142.2.2	~ScrewJoint	747
10.142.3	Member Function Documentation	747
10.142.3.1	GetAnchor	747
10.142.3.2	GetAngleCount	748
10.142.3.3	GetThreadPitch	748
10.142.3.4	Load	748
10.142.3.5	SetAnchor	748
10.142.3.6	SetThreadPitch	749
10.142.4	Member Data Documentation	749
10.142.4.1	fakeAnchor	749
10.142.4.2	threadPitch	749
10.143	gazebo::rendering::SelectionObj Class Reference	749
10.143.1	Detailed Description	751
10.144	gazebo::sensors::Sensor Class Reference	751
10.144.1	Detailed Description	754
10.144.2	Constructor & Destructor Documentation	754

10.144.2.1Sensor	754
10.144.2.2~Sensor	755
10.144.3Member Function Documentation	755
10.144.3.1ConnectUpdated	755
10.144.3.2DisconnectUpdated	755
10.144.3.3FillMsg	755
10.144.3.4Fini	755
10.144.3.5GetCategory	756
10.144.3.6GetId	756
10.144.3.7GetLastMeasurementTime	756
10.144.3.8GetLastUpdateTime	756
10.144.3.9GetName	756
10.144.3.10GetParentId	757
10.144.3.11GetParentName	757
10.144.3.12GetPose	757
10.144.3.13GetScopedName	757
10.144.3.14GetTopic	757
10.144.3.15GetType	758
10.144.3.16GetUpdateRate	758
10.144.3.17GetVisualize	758
10.144.3.18GetWorldName	758
10.144.3.19Init	758
10.144.3.20Active	758
10.144.3.21load	759
10.144.3.22load	759
10.144.3.23ResetLastUpdateTime	759
10.144.3.24SetActive	759
10.144.3.25SetParent	760
10.144.3.26SetParent	760
10.144.3.27SetUpdateRate	760
10.144.3.28update	760
10.144.3.29updateImpl	760
10.144.4Member Data Documentation	761
10.144.4.1active	761
10.144.4.2connections	761
10.144.4.3lastMeasurementTime	761
10.144.4.4lastUpdateTime	761

10.144.4.5	mutexLastUpdateTime	761
10.144.4.6	node	761
10.144.4.7	parentId	761
10.144.4.8	parentName	761
10.144.4.9	plugins	761
10.144.4.10	pose	761
10.144.4.11	poseSub	762
10.144.4.12	scene	762
10.144.4.13	stf	762
10.144.4.14	updatePeriod	762
10.144.4.15	world	762
10.145	SensorFactor Class Reference	762
10.145.1	Detailed Description	762
10.146	gazebo::sensors::SensorFactory Class Reference	762
10.146.1	Member Function Documentation	763
10.146.1.1	GetSensorTypes	763
10.146.1.2	NewSensor	763
10.146.1.3	RegisterAll	763
10.146.1.4	RegisterSensor	764
10.147	gazebo::sensors::SensorManager Class Reference	764
10.147.1	Detailed Description	765
10.147.2	Member Function Documentation	765
10.147.2.1	CreateSensor	765
10.147.2.2	CreateSensor	765
10.147.2.3	Fini	766
10.147.2.4	GetSensor	766
10.147.2.5	GetSensors	766
10.147.2.6	GetSensorTypes	766
10.147.2.7	Init	766
10.147.2.8	RemoveSensor	766
10.147.2.9	RemoveSensors	766
10.147.2.10	ResetLastUpdateTimes	767
10.147.2.11	RunThreads	767
10.147.2.12	SensorsInitialized	767
10.147.2.13	Stop	767
10.147.2.14	Update	767
10.148	gazebo::SensorPlugin Class Reference	767

10.148.1	Detailed Description	768
10.148.2	Constructor & Destructor Documentation	768
10.148.2.1	SensorPlugin	768
10.148.2.2	~SensorPlugin	768
10.148.3	Member Function Documentation	769
10.148.3.1	Init	769
10.148.3.2	Load	769
10.148.3.3	Reset	769
10.149	Gazebo::Server Class Reference	769
10.149.1	Constructor & Destructor Documentation	770
10.149.1.1	Server	770
10.149.1.2	~Server	770
10.149.2	Member Function Documentation	770
10.149.2.1	Fini	770
10.149.2.2	GetInitialized	770
10.149.2.3	Init	770
10.149.2.4	LoadFile	770
10.149.2.5	LoadString	770
10.149.2.6	ParseArgs	770
10.149.2.7	PreLoad	770
10.149.2.8	PrintUsage	770
10.149.2.9	Run	770
10.149.2.10	SetParams	770
10.149.2.11	Stop	770
10.149.3	Member Data Documentation	770
10.149.3.1	systemPluginsArgc	771
10.149.3.2	systemPluginsArgv	771
10.150	Gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg Class Reference	771
10.150.1	Detailed Description	772
10.150.2	Member Function Documentation	772
10.150.2.1	defaultVpParams	772
10.150.2.2	generateFragmentProgram	772
10.150.2.3	generateVertexProgram	772
10.150.2.4	generateVertexProgramSource	772
10.150.2.5	generateVpDynamicShadows	772
10.150.2.6	generateVpDynamicShadowsParams	772
10.150.2.7	generateVpFooter	772

10.150.2.8	generateVpHeader	772
10.150	gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL Class Reference	773
10.151.1	Detailed Description	774
10.151.2	Member Function Documentation	774
10.151.2.1	defaultVpParams	774
10.151.2.2	generateFpDynamicShadows	774
10.151.2.3	generateFpDynamicShadowsHelpers	774
10.151.2.4	generateFpDynamicShadowsParams	774
10.151.2.5	generateFpFooter	774
10.151.2.6	generateFpHeader	774
10.151.2.7	generateFpLayer	774
10.151.2.8	generateFragmentProgram	775
10.151.2.9	generateFragmentProgramSource	775
10.151.2.10	generateVertexProgram	775
10.151.2.11	generateVertexProgramSource	775
10.151.2.12	generateVpDynamicShadows	775
10.151.2.13	generateVpDynamicShadowsParams	775
10.151.2.14	generateVpFooter	775
10.151.2.15	generateVpHeader	775
10.151.2.16	updateParams	775
10.151.2.17	updateVpParams	775
10.152	gazebo::physics::Shape Class Reference	775
10.152.1	Detailed Description	776
10.152.2	Constructor & Destructor Documentation	777
10.152.2.1	Shape	777
10.152.2.2	~Shape	777
10.152.3	Member Function Documentation	777
10.152.3.1	FillMsg	777
10.152.3.2	GetScale	777
10.152.3.3	Init	777
10.152.3.4	ProcessMsg	778
10.152.3.5	SetScale	778
10.152.4	Member Data Documentation	778
10.152.4.1	collisionParent	778
10.152.4.2	scale	778
10.153	gazebo::physics::SimbodyBallJoint Class Reference	778
10.153.1	Detailed Description	780

10.153.2	Constructor & Destructor Documentation	780
10.153.2.1	SimbodyBallJoint	780
10.153.2.2	~SimbodyBallJoint	780
10.153.3	Member Function Documentation	781
10.153.3.1	GetAnchor	781
10.153.3.2	GetAngleImpl	781
10.153.3.3	GetAxis	781
10.153.3.4	GetGlobalAxis	781
10.153.3.5	GetMaxForce	781
10.153.3.6	GetVelocity	782
10.153.3.7	nit	782
10.153.3.8	Load	782
10.153.3.9	SetDamping	782
10.153.3.10	SetForceImpl	783
10.153.3.11	SetHighStop	783
10.153.3.12	SetLowStop	783
10.153.3.13	SetMaxForce	783
10.153.3.14	SetVelocity	784
10.154	gazebo::physics::SimbodyBoxShape Class Reference	784
10.154.1	Detailed Description	785
10.154.2	Constructor & Destructor Documentation	785
10.154.2.1	SimbodyBoxShape	785
10.154.2.2	~SimbodyBoxShape	785
10.154.3	Member Function Documentation	785
10.154.3.1	SetSize	785
10.155	gazebo::physics::SimbodyCollision Class Reference	785
10.155.1	Detailed Description	787
10.155.2	Constructor & Destructor Documentation	787
10.155.2.1	SimbodyCollision	787
10.155.2.2	~SimbodyCollision	787
10.155.3	Member Function Documentation	787
10.155.3.1	GetBoundingBox	787
10.155.3.2	GetCollisionShape	787
10.155.3.3	Load	787
10.155.3.4	OnPoseChange	788
10.155.3.5	SetCategoryBits	788
10.155.3.6	SetCollideBits	788

10.155.3.7	SetCollisionShape	788
10.156	gazebo::physics::SimbodyCylinderShape Class Reference	788
10.156.1	Detailed Description	789
10.156.2	Constructor & Destructor Documentation	790
10.156.2.1	SimbodyCylinderShape	790
10.156.2.2	~SimbodyCylinderShape	790
10.156.3	Member Function Documentation	790
10.156.3.1	SetSize	790
10.157	gazebo::physics::SimbodyHeightmapShape Class Reference	790
10.157.1	Detailed Description	791
10.157.2	Constructor & Destructor Documentation	792
10.157.2.1	SimbodyHeightmapShape	792
10.157.2.2	~SimbodyHeightmapShape	792
10.157.3	Member Function Documentation	792
10.157.3.1	Init	792
10.158	gazebo::physics::SimbodyHinge2Joint Class Reference	792
10.158.1	Detailed Description	794
10.158.2	Constructor & Destructor Documentation	794
10.158.2.1	SimbodyHinge2Joint	794
10.158.2.2	~SimbodyHinge2Joint	794
10.158.3	Member Function Documentation	795
10.158.3.1	GetAnchor	795
10.158.3.2	GetAngleImpl	795
10.158.3.3	GetAxis	795
10.158.3.4	GetGlobalAxis	795
10.158.3.5	GetHighStop	795
10.158.3.6	GetLowStop	796
10.158.3.7	GetMaxForce	796
10.158.3.8	GetVelocity	796
10.158.3.9	Init	797
10.158.3.10	Load	797
10.158.3.11	SetAxis	797
10.158.3.12	SetDamping	797
10.158.3.13	SetForceImpl	797
10.158.3.14	SetHighStop	798
10.158.3.15	SetLowStop	798
10.158.3.16	SetMaxForce	798

10.158.3.1	SetVelocity	798
10.159	Gazebo::physics::SimbodyHingeJoint Class Reference	799
10.159.1	Detailed Description	800
10.159.2	Constructor & Destructor Documentation	800
10.159.2.1	SimbodyHingeJoint	800
10.159.2.2	~SimbodyHingeJoint	801
10.159.3	Member Function Documentation	801
10.159.3.1	GetAngleImpl	801
10.159.3.2	GetGlobalAxis	801
10.159.3.3	GetHighStop	801
10.159.3.4	GetLowStop	802
10.159.3.5	GetMaxForce	802
10.159.3.6	GetVelocity	802
10.159.3.7	Load	802
10.159.3.8	RestoreSimbodyState	803
10.159.3.9	SaveSimbodyState	803
10.159.3.10	SetAxis	803
10.159.3.11	SetDamping	803
10.159.3.12	SetForceImpl	803
10.159.3.13	SetHighStop	804
10.159.3.14	SetLowStop	804
10.159.3.15	SetMaxForce	804
10.159.3.16	SetVelocity	804
10.160	Gazebo::physics::SimbodyJoint Class Reference	805
10.160.1	Detailed Description	807
10.160.2	Constructor & Destructor Documentation	807
10.160.2.1	SimbodyJoint	807
10.160.2.2	~SimbodyJoint	807
10.160.3	Member Function Documentation	807
10.160.3.1	AreConnected	807
10.160.3.2	CacheForceTorque	807
10.160.3.3	Detach	807
10.160.3.4	GetAnchor	807
10.160.3.5	GetAttribute	808
10.160.3.6	GetForce	808
10.160.3.7	GetForceTorque	808
10.160.3.8	GetJointLink	809

10.160.3.9	GetLinkForce	809
10.160.3.10	GetLinkTorque	809
10.160.3.11	load	810
10.160.3.12	reset	810
10.160.3.13	RestoreSimbodyState	810
10.160.3.13	saveSimbodyState	810
10.160.3.15	SetAnchor	810
10.160.3.16	SetAttribute	811
10.160.3.16	SetAttribute	811
10.160.3.18	SetAxis	811
10.160.3.19	SetDamping	811
10.160.3.20	SetForce	811
10.160.3.23	SetForceImpl	812
10.160.4	Member Data Documentation	812
10.160.4.1	constraint	812
10.160.4.2	damper	812
10.160.4.3	defxAB	812
10.160.4.4	isReversed	812
10.160.4.5	limitForce	813
10.160.4.6	mobod	813
10.160.4.7	mustBreakLoopHere	813
10.160.4.8	physicsInitialized	813
10.160.4.9	simbodyPhysics	813
10.160.4.10	world	813
10.160.4.11	CB	813
10.160.4.12	PA	813
10.161	gazebo::physics::SimbodyLink Class Reference	813
10.161.1	Detailed Description	816
10.161.2	Constructor & Destructor Documentation	816
10.161.2.1	SimbodyLink	816
10.161.2.2	~SimbodyLink	816
10.161.3	Member Function Documentation	816
10.161.3.1	AddForce	816
10.161.3.2	AddForceAtRelativePosition	817
10.161.3.3	AddForceAtWorldPosition	817
10.161.3.4	AddRelativeForce	817
10.161.3.5	AddRelativeTorque	817

10.161.3.6	AddTorque	817
10.161.3.7	Fin	818
10.161.3.8	GetEffectiveMassProps	818
10.161.3.9	GetEnabled	818
10.161.3.10	GetGravityMode	818
10.161.3.10	GetMassProperties	818
10.161.3.10	GetWorldAngularVel	818
10.161.3.10	GetWorldCoGLinearVel	818
10.161.3.10	GetWorldForce	819
10.161.3.10	GetWorldLinearVel	819
10.161.3.10	GetWorldLinearVel	819
10.161.3.10	GetWorldTorque	819
10.161.3.10	hit	820
10.161.3.10	load	820
10.161.3.20	OnPoseChange	820
10.161.3.21	RestoreSimbodyState	820
10.161.3.22	SaveSimbodyState	820
10.161.3.23	SetAngularDamping	820
10.161.3.23	SetAngularVel	820
10.161.3.25	SetAutoDisable	820
10.161.3.26	SetDirtyPose	821
10.161.3.27	SetEnabled	821
10.161.3.28	SetForce	821
10.161.3.29	SetGravityMode	821
10.161.3.30	SetLinearDamping	821
10.161.3.30	SetLinearVel	821
10.161.3.30	SetLinkStatic	822
10.161.3.30	SetSelfCollide	822
10.161.3.30	SetTorque	822
10.161.4	Member Data Documentation	822
10.161.4.1	masterMobod	822
10.161.4.2	mustBeBaseLink	822
10.161.4.3	physicsInitialized	822
10.161.4.4	slaveMobods	822
10.161.4.5	slaveWelds	823
10.162	Gazebo::physics::SimbodyMeshShape Class Reference	823
10.162.1	Detailed Description	824

10.162.2	Constructor & Destructor Documentation	824
10.162.2.1	SimbodyMeshShape	824
10.162.2.2	~SimbodyMeshShape	824
10.162.3	Member Function Documentation	824
10.162.3.1	Init	824
10.162.3.2	Load	824
10.163	Gazebo::physics::SimbodyModel Class Reference	825
10.163.1	Detailed Description	826
10.163.2	Constructor & Destructor Documentation	826
10.163.2.1	SimbodyModel	826
10.163.2.2	~SimbodyModel	826
10.163.3	Member Function Documentation	826
10.163.3.1	Init	826
10.163.3.2	Load	826
10.164	Gazebo::physics::SimbodyMultiRayShape Class Reference	826
10.164.1	Detailed Description	828
10.164.2	Constructor & Destructor Documentation	828
10.164.2.1	SimbodyMultiRayShape	828
10.164.2.2	~SimbodyMultiRayShape	828
10.164.3	Member Function Documentation	828
10.164.3.1	AddRay	828
10.164.3.2	UpdateRays	828
10.165	Gazebo::physics::SimbodyPhysics Class Reference	828
10.165.1	Detailed Description	831
10.165.2	Constructor & Destructor Documentation	831
10.165.2.1	SimbodyPhysics	831
10.165.2.2	~SimbodyPhysics	831
10.165.3	Member Function Documentation	831
10.165.3.1	CreateCollision	831
10.165.3.2	CreateJoint	831
10.165.3.3	CreateLink	832
10.165.3.4	CreateModel	832
10.165.3.5	CreateShape	832
10.165.3.6	DebugPrint	832
10.165.3.7	Finis	832
10.165.3.8	GetDynamicsWorld	832
10.165.3.9	GetPose	832

10.165.3.10	GetType	833
10.165.3.10	GetTypeString	833
10.165.3.10	GetTypeString	833
10.165.3.15	Wait	833
10.165.3.14	WaitForThread	833
10.165.3.15	WaitModel	834
10.165.3.16	Load	834
10.165.3.10	OnPhysicsMsg	834
10.165.3.10	OnRequest	834
10.165.3.19	Pose2Transform	834
10.165.3.20	QuadToQuad	835
10.165.3.20	QuadToQuad	835
10.165.3.22	Reset	835
10.165.3.29	SetGravity	835
10.165.3.29	SetSeed	835
10.165.3.27	Transform2Pose	836
10.165.3.26	UpdateCollision	836
10.165.3.27	UpdatePhysics	836
10.165.3.26	Vec3ToVector3	836
10.165.3.26	Vector3ToVec3	836
10.165.4	Member Data Documentation	837
10.165.4.1	contact	837
10.165.4.2	discreteForces	837
10.165.4.3	forces	837
10.165.4.4	gravity	837
10.165.4.5	integ	837
10.165.4.6	matter	837
10.165.4.7	simbodyPhysicsInitialized	837
10.165.4.8	simbodyPhysicsStepped	837
10.165.4.9	system	837
10.165.4.10	tacker	837
10.166	Gazebo::physics::SimbodyPlaneShape Class Reference	837
10.166.1	Detailed Description	838
10.166.2	Constructor & Destructor Documentation	839
10.166.2.1	SimbodyPlaneShape	839
10.166.2.2	~SimbodyPlaneShape	839
10.166.3	Member Function Documentation	839

10.166.3.1	CreatePlane	839
10.166.3.2	SetAltitude	839
10.167	gazebo::physics::SimbodyRayShape Class Reference	839
10.167.1	Detailed Description	841
10.167.2	Constructor & Destructor Documentation	841
10.167.2.1	SimbodyRayShape	841
10.167.2.2	SimbodyRayShape	841
10.167.2.3	~SimbodyRayShape	841
10.167.3	Member Function Documentation	841
10.167.3.1	GetIntersection	841
10.167.3.2	SetPoints	841
10.167.3.3	Update	842
10.168	gazebo::physics::SimbodyScrewJoint Class Reference	842
10.168.1	Detailed Description	844
10.168.2	Constructor & Destructor Documentation	844
10.168.2.1	SimbodyScrewJoint	844
10.168.2.2	~SimbodyScrewJoint	845
10.168.3	Member Function Documentation	845
10.168.3.1	GetAngleImpl	845
10.168.3.2	GetGlobalAxis	845
10.168.3.3	GetHighStop	845
10.168.3.4	GetLowStop	846
10.168.3.5	GetMaxForce	846
10.168.3.6	GetThreadPitch	846
10.168.3.7	GetVelocity	846
10.168.3.8	Init	847
10.168.3.9	Load	847
10.168.3.10	SetAxis	847
10.168.3.11	SetDamping	847
10.168.3.12	SetForceImpl	848
10.168.3.13	SetHighStop	848
10.168.3.14	SetLowStop	848
10.168.3.15	SetMaxForce	848
10.168.3.16	SetThreadPitch	849
10.168.3.17	SetVelocity	849
10.169	gazebo::physics::SimbodySliderJoint Class Reference	849
10.169.1	Detailed Description	851

10.169.2	Constructor & Destructor Documentation	851
10.169.2.1	SimbodySliderJoint	851
10.169.2.2	~SimbodySliderJoint	851
10.169.3	Member Function Documentation	852
10.169.3.1	GetAngleImpl	852
10.169.3.2	GetGlobalAxis	852
10.169.3.3	GetHighStop	852
10.169.3.4	GetLowStop	852
10.169.3.5	GetMaxForce	853
10.169.3.6	GetVelocity	853
10.169.3.7	Load	853
10.169.3.8	SetAxis	854
10.169.3.9	SetDamping	854
10.169.3.10	SetForceImpl	854
10.169.3.11	SetHighStop	854
10.169.3.12	SetLowStop	855
10.169.3.13	SetMaxForce	855
10.169.3.14	SetVelocity	855
10.170	gazebo::physics::SimbodySphereShape Class Reference	855
10.170.1	Detailed Description	856
10.170.2	Constructor & Destructor Documentation	856
10.170.2.1	SimbodySphereShape	857
10.170.2.2	~SimbodySphereShape	857
10.170.3	Member Function Documentation	857
10.170.3.1	SetRadius	857
10.171	gazebo::physics::SimbodyUniversalJoint Class Reference	857
10.171.1	Detailed Description	859
10.171.2	Constructor & Destructor Documentation	859
10.171.2.1	SimbodyUniversalJoint	859
10.171.2.2	~SimbodyUniversalJoint	860
10.171.3	Member Function Documentation	860
10.171.3.1	GetAnchor	860
10.171.3.2	GetAngleImpl	860
10.171.3.3	GetAxis	860
10.171.3.4	GetGlobalAxis	860
10.171.3.5	GetHighStop	861
10.171.3.6	GetLowStop	861

10.171.3.7	GetMaxForce	861
10.171.3.8	GetVelocity	861
10.171.3.9	Init	862
10.171.3.10	Load	862
10.171.3.11	SetAxis	862
10.171.3.12	SetDamping	862
10.171.3.13	SetForceImpl	863
10.171.3.14	SetHighStop	863
10.171.3.15	SetLowStop	863
10.171.3.16	SetMaxForce	863
10.171.3.17	SetVelocity	864
10.172	SingletonT< T > Class Template Reference	864
10.172.1	Detailed Description	866
10.172.2	Constructor & Destructor Documentation	866
10.172.2.1	SingletonT	866
10.172.2.2	~SingletonT	866
10.172.3	Member Function Documentation	866
10.172.3.1	Instance	866
10.173	Gazebo::common::Skeleton Class Reference	866
10.173.1	Detailed Description	868
10.173.2	Constructor & Destructor Documentation	868
10.173.2.1	Skeleton	868
10.173.2.2	~Skeleton	868
10.173.2.3	~Skeleton	868
10.173.3	Member Function Documentation	868
10.173.3.1	AddAnimation	868
10.173.3.2	AddVertNodeWeight	869
10.173.3.3	BuildNodeMap	869
10.173.3.4	GetAnimation	869
10.173.3.5	GetBindShapeTransform	869
10.173.3.6	GetNodeByHandle	869
10.173.3.7	GetNodeById	869
10.173.3.8	GetNodeByName	870
10.173.3.9	GetNodes	870
10.173.3.10	GetNumAnimations	870
10.173.3.11	GetNumJoints	870
10.173.3.12	GetNumNodes	870

10.173.3.1	GetNumVertNodeWeights	871
10.173.3.1	GetRootNode	871
10.173.3.1	GetVertNodeWeight	871
10.173.3.1	PrintTransforms	871
10.173.3.1	Scale	871
10.173.3.1	SetBindShapeTransform	871
10.173.3.1	SetNumVertAttached	872
10.173.3.2	SetRootNode	872
10.173.4	Member Data Documentation	872
10.173.4.1	animis	872
10.173.4.2	bindShapeTransform	872
10.173.4.3	nodes	872
10.173.4.4	rawNW	872
10.173.4.5	root	872
10.174	Gazebo::common::SkeletonAnimation Class Reference	873
10.174.1	Detailed Description	874
10.174.2	Constructor & Destructor Documentation	874
10.174.2.1	SkeletonAnimation	874
10.174.2.2	~SkeletonAnimation	874
10.174.3	Member Function Documentation	874
10.174.3.1	AddKeyFrame	874
10.174.3.2	AddKeyFrame	874
10.174.3.3	GetLength	874
10.174.3.4	GetName	875
10.174.3.5	GetNodeCount	875
10.174.3.6	GetNodePoseAt	875
10.174.3.7	GetPoseAt	875
10.174.3.8	GetPoseAtX	876
10.174.3.9	HasNode	876
10.174.3.10	Scale	876
10.174.3.11	SetName	876
10.174.4	Member Data Documentation	876
10.174.4.1	animations	876
10.174.4.2	length	877
10.174.4.3	name	877
10.175	Gazebo::common::SkeletonNode Class Reference	877
10.175.1	Detailed Description	879

10.175.2	Member Enumeration Documentation	879
10.175.2.1	SkeletonNodeType	879
10.175.3	Constructor & Destructor Documentation	879
10.175.3.1	SkeletonNode	879
10.175.3.2	SkeletonNode	880
10.175.3.3	~SkeletonNode	880
10.175.4	Member Function Documentation	880
10.175.4.1	AddChild	880
10.175.4.2	AddRawTransform	880
10.175.4.3	GetChild	880
10.175.4.4	GetChildById	880
10.175.4.5	GetChildByName	881
10.175.4.6	GetChildCount	881
10.175.4.7	GetHandle	881
10.175.4.8	GetId	881
10.175.4.9	GetInverseBindTransform	881
10.175.4.10	GetModelTransform	882
10.175.4.11	GetName	882
10.175.4.12	GetNumRawTrans	882
10.175.4.13	GetParent	882
10.175.4.14	GetRawTransform	882
10.175.4.15	GetRawTransforms	882
10.175.4.16	GetTransform	883
10.175.4.17	GetTransforms	883
10.175.4.18	Joint	883
10.175.4.19	RootNode	883
10.175.4.20	Reset	883
10.175.4.21	SetHandle	883
10.175.4.22	SetId	884
10.175.4.23	SetInitialTransform	884
10.175.4.24	SetInverseBindTransform	884
10.175.4.25	SetModelTransform	884
10.175.4.26	SetName	884
10.175.4.27	SetParent	884
10.175.4.28	SetTransform	885
10.175.4.29	SetType	885
10.175.4.30	UpdateChildrenTransforms	885

10.175.5	Member Data Documentation	885
10.175.5.1	children	885
10.175.5.2	handle	885
10.175.5.3	d	885
10.175.5.4	initialTransform	885
10.175.5.5	invBindTransform	885
10.175.5.6	modelTransform	885
10.175.5.7	name	886
10.175.5.8	parent	886
10.175.5.9	rawTransforms	886
10.175.5.10	ransform	886
10.175.5.11	type	886
10.176	Gazebo::physics::SliderJoint< T > Class Template Reference	886
10.176.1	Detailed Description	887
10.176.2	Constructor & Destructor Documentation	887
10.176.2.1	SliderJoint	887
10.176.2.2	~SliderJoint	887
10.176.3	Member Function Documentation	887
10.176.3.1	GetAnchor	887
10.176.3.2	GetAngleCount	888
10.176.3.3	Load	888
10.176.3.4	SetAnchor	888
10.176.4	Member Data Documentation	888
10.176.4.1	fakeAnchor	888
10.177	Gazebo::rendering::GzTerrainMatGen::SM2Profile Class Reference	888
10.177.1	Detailed Description	889
10.177.2	Constructor & Destructor Documentation	890
10.177.2.1	SM2Profile	890
10.177.2.2	~SM2Profile	890
10.177.3	Member Function Documentation	890
10.177.3.1	addTechnique	890
10.177.3.2	generate	890
10.177.3.3	generateForCompositeMap	890
10.177.3.4	updateParams	890
10.177.3.5	updateParamsForCompositeMap	890
10.178	Gazebo::sensors::SonarSensor Class Reference	890
10.178.1	Detailed Description	892

10.178.2	Constructor & Destructor Documentation	892
10.178.2.1	SonarSensor	892
10.178.2.2	~SonarSensor	892
10.178.3	Member Function Documentation	892
10.178.3.1	ConnectUpdate	892
10.178.3.2	DisconnectUpdate	892
10.178.3.3	Finis	893
10.178.3.4	GetRadius	893
10.178.3.5	GetRange	893
10.178.3.6	GetRangeMax	893
10.178.3.7	GetRangeMin	893
10.178.3.8	GetTopic	894
10.178.3.9	Init	894
10.178.3.10	IsActive	894
10.178.3.11	Load	894
10.178.3.12	UpdateImpl	894
10.178.4	Member Data Documentation	895
10.178.4.1	update	895
10.179	gazebo::rendering::SonarVisual Class Reference	895
10.179.1	Detailed Description	896
10.179.2	Constructor & Destructor Documentation	896
10.179.2.1	SonarVisual	896
10.179.2.2	~SonarVisual	896
10.179.3	Member Function Documentation	896
10.179.3.1	Load	896
10.180	joint_TEST::SpawnJointOptions Class Reference	896
10.180.1	Detailed Description	897
10.180.2	Constructor & Destructor Documentation	897
10.180.2.1	SpawnJointOptions	897
10.180.2.2	~SpawnJointOptions	897
10.180.3	Member Data Documentation	897
10.180.3.1	axis	897
10.180.3.2	childLinkPose	897
10.180.3.3	jointPose	898
10.180.3.4	modelPose	898
10.180.3.5	parentLinkPose	898
10.180.3.6	type	898

10.180.3.7wait	898
10.180.3.8worldChild	898
10.180.3.9worldParent	898
10.180.4gazebo::physics::SphereShape Class Reference	898
10.181.1Detailed Description	900
10.181.2Constructor & Destructor Documentation	900
10.181.2.1SphereShape	900
10.181.2.2~SphereShape	900
10.181.3Member Function Documentation	900
10.181.3.1FillMsg	900
10.181.3.2GetRadius	900
10.181.3.3Init	900
10.181.3.4ProcessMsg	900
10.181.3.5SetRadius	901
10.181.3.6SetScale	901
10.180.5gazebo::common::SphericalCoordinates Class Reference	901
10.182.1Detailed Description	902
10.182.2Member Enumeration Documentation	902
10.182.2.1SurfaceType	902
10.182.3Constructor & Destructor Documentation	903
10.182.3.1SphericalCoordinates	903
10.182.3.2SphericalCoordinates	903
10.182.3.3SphericalCoordinates	903
10.182.3.4~SphericalCoordinates	903
10.182.4Member Function Documentation	903
10.182.4.1Convert	903
10.182.4.2GetElevationReference	903
10.182.4.3GetHeadingOffset	904
10.182.4.4GetLatitudeReference	904
10.182.4.5GetLongitudeReference	904
10.182.4.6GetSurfaceType	904
10.182.4.7GlobalFromLocal	904
10.182.4.8SetElevationReference	905
10.182.4.9SetHeadingOffset	905
10.182.4.10SetLatitudeReference	905
10.182.4.11SetLongitudeReference	905
10.182.4.12SetSurfaceType	905

10.182.4.1	SphericalFromLocal	905
10.183	gazebo::math::Spline Class Reference	906
10.183.1	Detailed Description	907
10.183.2	Constructor & Destructor Documentation	907
10.183.2.1	Spline	907
10.183.2.2	~Spline	907
10.183.3	Member Function Documentation	907
10.183.3.1	AddPoint	907
10.183.3.2	Clear	907
10.183.3.3	GetPoint	907
10.183.3.4	GetPointCount	908
10.183.3.5	GetTangent	908
10.183.3.6	GetTension	908
10.183.3.7	Interpolate	908
10.183.3.8	Interpolate	908
10.183.3.9	RecalcTangents	909
10.183.3.10	SetAutoCalculate	909
10.183.3.11	SetTension	909
10.183.3.12	UpdatePoint	909
10.183.4	Member Data Documentation	910
10.183.4.1	autoCalc	910
10.183.4.2	coeffs	910
10.183.4.3	points	910
10.183.4.4	tangents	910
10.183.4.5	tension	910
10.184	gazebo::physics::State Class Reference	910
10.184.1	Detailed Description	912
10.184.2	Constructor & Destructor Documentation	912
10.184.2.1	State	912
10.184.2.2	State	912
10.184.2.3	~State	912
10.184.3	Member Function Documentation	912
10.184.3.1	GetName	912
10.184.3.2	GetRealTime	913
10.184.3.3	GetSimTime	913
10.184.3.4	GetWallTime	913
10.184.3.5	Load	913

10.184.3.6operator-	913
10.184.3.7operator=	914
10.184.3.8SetName	914
10.184.3.9SetRealTime	914
10.184.3.10SetSimTime	914
10.184.3.11SetWallTime	914
10.184.4Member Data Documentation	915
10.184.4.1name	915
10.184.4.2realTime	915
10.184.4.3simTime	915
10.184.4.4wallTime	915
10.185gazebo::common::STLLoader Class Reference	915
10.185.1Detailed Description	916
10.185.2Constructor & Destructor Documentation	916
10.185.2.1STLLoader	916
10.185.2.2~STLLoader	916
10.185.3Member Function Documentation	916
10.185.3.1Load	916
10.186gazebo::common::SubMesh Class Reference	916
10.186.1Detailed Description	919
10.186.2Member Enumeration Documentation	919
10.186.2.1PrimitiveType	919
10.186.3Constructor & Destructor Documentation	919
10.186.3.1SubMesh	919
10.186.3.2SubMesh	919
10.186.3.3~SubMesh	919
10.186.4Member Function Documentation	919
10.186.4.1AddIndex	919
10.186.4.2AddNodeAssignment	920
10.186.4.3AddNormal	920
10.186.4.4AddNormal	920
10.186.4.5AddTexCoord	920
10.186.4.6AddVertex	920
10.186.4.7AddVertex	920
10.186.4.8Center	921
10.186.4.9CopyNormals	921
10.186.4.10CopyVertices	921

10.186.4.1	FillArrays	921
10.186.4.1	GenSphericalTexCoord	921
10.186.4.1	GetIndex	922
10.186.4.1	GetIndexCount	922
10.186.4.1	GetMaterialIndex	922
10.186.4.1	GetMax	922
10.186.4.1	GetMaxIndex	922
10.186.4.1	GetMin	922
10.186.4.1	GetName	922
10.186.4.2	GetNodeAssignment	923
10.186.4.2	GetNodeAssignmentsCount	923
10.186.4.2	GetNormal	923
10.186.4.2	GetNormalCount	923
10.186.4.2	GetPrimitiveType	923
10.186.4.2	GetTexCoord	923
10.186.4.2	GetTexCoordCount	924
10.186.4.2	GetVertex	924
10.186.4.2	GetVertexCount	924
10.186.4.2	GetVertexIndex	924
10.186.4.3	HasVertex	924
10.186.4.3	RecalculateNormals	924
10.186.4.3	Scale	924
10.186.4.3	SetIndexCount	925
10.186.4.3	SetMaterialIndex	925
10.186.4.3	SetName	925
10.186.4.3	SetNormal	925
10.186.4.3	SetNormalCount	925
10.186.4.3	SetPrimitiveType	925
10.186.4.3	SetScale	926
10.186.4.4	GetSubMeshCenter	926
10.186.4.4	SetTexCoord	926
10.186.4.4	SetTexCoordCount	926
10.186.4.4	SetVertex	926
10.186.4.4	SetVertexCount	927
10.186.4.4	Translate	927
10.187	Gazebo::transport::SubscribeOptions Class Reference	927
10.187	Detailed Description	927

10.187.2	Constructor & Destructor Documentation	928
10.187.2.1	SubscribeOptions	928
10.187.3	Member Function Documentation	928
10.187.3.1	GetLatching	928
10.187.3.2	GetMsgType	928
10.187.3.3	GetNode	928
10.187.3.4	GetTopic	928
10.187.3.5	init	928
10.187.3.6	init	929
10.188	gazebo::transport::Subscriber Class Reference	929
10.188.1	Detailed Description	929
10.188.2	Constructor & Destructor Documentation	929
10.188.2.1	Subscriber	929
10.188.2.2	~Subscriber	930
10.188.3	Member Function Documentation	930
10.188.3.1	GetCallbackId	930
10.188.3.2	GetTopic	930
10.188.3.3	SetCallbackId	930
10.188.3.4	Unsubscribe	930
10.189	gazebo::transport::SubscriptionTransport Class Reference	930
10.189.1	Detailed Description	931
10.189.2	Constructor & Destructor Documentation	931
10.189.2.1	SubscriptionTransport	931
10.189.2.2	~SubscriptionTransport	932
10.189.3	Member Function Documentation	932
10.189.3.1	GetConnection	932
10.189.3.2	HandleData	932
10.189.3.3	HandleMessage	932
10.189.3.4	init	932
10.189.3.5	isLocal	933
10.190	gazebo::physics::SurfaceParams Class Reference	933
10.190.1	Detailed Description	934
10.190.2	Constructor & Destructor Documentation	934
10.190.2.1	SurfaceParams	934
10.190.2.2	~SurfaceParams	934
10.190.3	Member Function Documentation	934
10.190.3.1	FillMsg	934

10.190.3.2	Load	935
10.190.3.3	ProcessMsg	935
10.190.4	Member Data Documentation	935
10.190.4.1	bounce	935
10.190.4.2	bounceThreshold	935
10.190.4.3	cfm	935
10.190.4.4	collideWithoutContact	935
10.190.4.5	collideWithoutContactBitmask	935
10.190.4.6	erp	935
10.190.4.7	dir1	936
10.190.4.8	kd	936
10.190.4.9	kp	936
10.190.4.10	axVel	936
10.190.4.11	rhinDepth	936
10.190.4.12	u1	937
10.190.4.13	u2	937
10.190.4.14	slip1	937
10.190.4.15	slip2	937
10.191	Gazebo::common::SystemPaths Class Reference	937
10.191.1	Detailed Description	939
10.191.2	Member Function Documentation	939
10.191.2.1	AddGazeboPaths	939
10.191.2.2	AddModelPaths	939
10.191.2.3	AddOgrePaths	940
10.191.2.4	AddPluginPaths	940
10.191.2.5	AddSearchPathSuffix	940
10.191.2.6	ClearGazeboPaths	940
10.191.2.7	ClearModelPaths	940
10.191.2.8	ClearOgrePaths	940
10.191.2.9	ClearPluginPaths	940
10.191.2.10	FindFile	940
10.191.2.11	FindFileURI	941
10.191.2.12	GetGazeboPaths	941
10.191.2.13	GetLogPath	941
10.191.2.14	GetModelPaths	941
10.191.2.15	GetOgrePaths	941
10.191.2.16	GetPluginPaths	942

10.191.2.1	GetWorldPathExtension	942
10.191.3	Member Data Documentation	942
10.191.3.1	gazeboPathsFromEnv	942
10.191.3.2	modelPathsFromEnv	942
10.191.3.3	ogrePathsFromEnv	942
10.191.3.4	pluginPathsFromEnv	942
10.192	gazebo::SystemPlugin Class Reference	942
10.192.1	Detailed Description	943
10.192.2	Constructor & Destructor Documentation	943
10.192.2.1	SystemPlugin	943
10.192.2.2	~SystemPlugin	943
10.192.3	Member Function Documentation	944
10.192.3.1	Init	944
10.192.3.2	Load	944
10.192.3.3	Reset	944
10.193	gazebo::common::Time Class Reference	944
10.193.1	Detailed Description	948
10.193.2	Constructor & Destructor Documentation	948
10.193.2.1	Time	948
10.193.2.2	Time	948
10.193.2.3	Time	948
10.193.2.4	Time	948
10.193.2.5	Time	949
10.193.2.6	Time	949
10.193.2.7	~Time	949
10.193.3	Member Function Documentation	949
10.193.3.1	Double	949
10.193.3.2	Float	949
10.193.3.3	GetWallTime	949
10.193.3.4	GetWallTimeAsISOString	950
10.193.3.5	MicToNano	950
10.193.3.6	MilToNano	950
10.193.3.7	MSleep	950
10.193.3.8	NSleep	950
10.193.3.9	operator!=	951
10.193.3.10	operator!=	951
10.193.3.11	operator!=	951

10.193.3.10	operator!=	951
10.193.3.10	operator*	952
10.193.3.10	operator*	952
10.193.3.15	operator*	952
10.193.3.16	operator*==	952
10.193.3.17	operator*==	953
10.193.3.16	operator*==	953
10.193.3.16	operator+	953
10.193.3.20	operator+	953
10.193.3.21	operator+	954
10.193.3.22	operator+=	954
10.193.3.23	operator+=	954
10.193.3.24	operator+=	954
10.193.3.25	operator-	955
10.193.3.26	operator-	955
10.193.3.27	operator-	955
10.193.3.28	operator-=	955
10.193.3.29	operator-=	956
10.193.3.30	operator-=	956
10.193.3.31	operator/	956
10.193.3.32	operator/	956
10.193.3.33	operator/	957
10.193.3.34	operator/=	957
10.193.3.35	operator/=	957
10.193.3.36	operator/=	957
10.193.3.37	operator<	958
10.193.3.38	operator<	958
10.193.3.39	operator<	958
10.193.3.40	operator<	958
10.193.3.41	operator<=	959
10.193.3.42	operator<=	959
10.193.3.43	operator<=	959
10.193.3.44	operator<=	959
10.193.3.45	operator=	960
10.193.3.46	operator=	960
10.193.3.47	operator=	960
10.193.3.48	operator==	960

10.193.3.49	operator==	961
10.193.3.50	operator==	961
10.193.3.51	operator==	961
10.193.3.52	operator>	961
10.193.3.53	operator>	962
10.193.3.54	operator>	962
10.193.3.55	operator>	962
10.193.3.56	operator>=	962
10.193.3.57	operator>=	963
10.193.3.58	operator>=	963
10.193.3.59	operator>=	963
10.193.3.60	SecToNano	963
10.193.3.61	Set	964
10.193.3.62	Set	964
10.193.3.63	SetToWallTime	964
10.193.3.64	sleep	964
10.193.4	Friends And Related Function Documentation	964
10.193.4.1	operator<<	964
10.193.4.2	operator>>	965
10.193.5	Member Data Documentation	965
10.193.5.1	insec	965
10.193.5.2	sec	965
10.193.5.3	Zero	965
10.194	gazebo::common::Timer Class Reference	965
10.194.1	Detailed Description	966
10.194.2	Constructor & Destructor Documentation	966
10.194.2.1	Timer	966
10.194.2.2	~Timer	967
10.194.3	Member Function Documentation	967
10.194.3.1	GetElapsed	967
10.194.3.2	GetRunning	967
10.194.3.3	Start	967
10.194.3.4	Stop	967
10.194.4	Friends And Related Function Documentation	967
10.194.4.1	operator<<	967
10.195	gazebo::transport::TopicManager Class Reference	967
10.195.1	Detailed Description	969

10.195.2	Member Typedef Documentation	969
10.195.2.1	SubNodeMap	969
10.195.3	Member Function Documentation	969
10.195.3.1	AddNode	969
10.195.3.2	AddNodeToProcess	970
10.195.3.3	Advertise	970
10.195.3.4	ClearBuffers	970
10.195.3.5	ConnectPubToSub	970
10.195.3.6	ConnectSubscribers	970
10.195.3.7	ConnectSubToPub	971
10.195.3.8	DisconnectPubFromSub	971
10.195.3.9	DisconnectSubFromPub	971
10.195.3.10	FindPublication	971
10.195.3.11	Finis	971
10.195.3.12	GetTopicNamespaces	972
10.195.3.13	Hit	972
10.195.3.14	IsAdvertised	972
10.195.3.15	PauseIncoming	972
10.195.3.16	ProcessNodes	972
10.195.3.17	Publish	972
10.195.3.18	RegisterTopicNamespace	973
10.195.3.19	RemoveNode	973
10.195.3.20	Subscribe	973
10.195.3.21	Unadvertise	973
10.195.3.22	Unsubscribe	973
10.195.3.23	UpdatePublications	974
10.196	Gazebo::physics::TrajectoryInfo Struct Reference	974
10.196.1	Member Data Documentation	974
10.196.1.1	duration	974
10.196.1.2	endTime	974
10.196.1.3	d	974
10.196.1.4	startTime	974
10.196.1.5	translated	974
10.196.1.6	type	975
10.197	Gazebo::rendering::TransmitterVisual Class Reference	975
10.197.1	Detailed Description	976
10.197.2	Constructor & Destructor Documentation	976

10.197.2.1	TransmitterVisual	976
10.197.2.2	~TransmitterVisual	976
10.197.3	Member Function Documentation	976
10.197.3.1	Load	976
10.197.3.2	Update	976
10.198	gazebo::physics::UniversalJoint< T > Class Template Reference	976
10.198.1	Detailed Description	977
10.198.2	Constructor & Destructor Documentation	977
10.198.2.1	UniversalJoint	977
10.198.2.2	~UniversalJoint	978
10.198.3	Member Function Documentation	978
10.198.3.1	GetAngleCount	978
10.198.3.2	Load	978
10.199	gazebo::common::UpdateInfo Class Reference	978
10.199.1	Detailed Description	978
10.199.2	Member Data Documentation	978
10.199.2.1	realTime	978
10.199.2.2	simTime	979
10.199.2.3	worldName	979
10.200	gazebo::rendering::UserCamera Class Reference	979
10.200.1	Detailed Description	981
10.200.2	Constructor & Destructor Documentation	981
10.200.2.1	UserCamera	981
10.200.2.2	~UserCamera	981
10.200.3	Member Function Documentation	981
10.200.3.1	AnimationComplete	981
10.200.3.2	AttachToVisualImpl	982
10.200.3.3	EnableViewController	982
10.200.3.4	Finis	982
10.200.3.5	GetAvgFPS	982
10.200.3.6	GetGUIOverlay	982
10.200.3.7	GetImageHeight	983
10.200.3.8	GetImageWidth	983
10.200.3.9	GetTriangleCount	983
10.200.3.10	GetViewControllerTypeString	983
10.200.3.11	GetVisual	983
10.200.3.12	GetVisual	984

10.200.3.13	HandleKeyPressEvent	984
10.200.3.14	HandleKeyReleaseEvent	984
10.200.3.15	HandleMouseEvent	984
10.200.3.16	Hit	984
10.200.3.17	Load	984
10.200.3.18	Load	985
10.200.3.19	MoveToPosition	985
10.200.3.20	MoveToVisual	985
10.200.3.21	MoveToVisual	985
10.200.3.22	PostRender	985
10.200.3.23	Resize	986
10.200.3.24	SetFocalPoint	986
10.200.3.25	SetRenderTarget	986
10.200.3.26	SetViewController	986
10.200.3.27	SetViewController	986
10.200.3.28	SetViewportDimensions	986
10.200.3.29	SetWorldPose	987
10.200.3.30	TrackVisualImpl	987
10.200.3.31	Update	987
10.201	gazebo::math::Vector2d Class Reference	987
10.201.1	Detailed Description	989
10.201.2	Constructor & Destructor Documentation	989
10.201.2.1	Vector2d	989
10.201.2.2	Vector2d	989
10.201.2.3	Vector2d	989
10.201.2.4	~Vector2d	990
10.201.3	Member Function Documentation	990
10.201.3.1	Cross	990
10.201.3.2	Distance	990
10.201.3.3	IsFinite	990
10.201.3.4	Normalize	990
10.201.3.5	operator!=	990
10.201.3.6	operator*	991
10.201.3.7	operator*	991
10.201.3.8	operator*=	991
10.201.3.9	operator*=	991
10.201.3.10	operator+	992

10.201.3.1	operator+=	992
10.201.3.1	operator-	992
10.201.3.1	operator--	992
10.201.3.1	operator/	993
10.201.3.1	operator/	993
10.201.3.1	operator/=	993
10.201.3.1	operator/=	993
10.201.3.1	operator=	994
10.201.3.1	operator=	994
10.201.3.2	operator==	994
10.201.3.2	operator[]	994
10.201.3.2	set	995
10.201.4	Friends And Related Function Documentation	995
10.201.4.1	operator<<	995
10.201.4.2	operator>>	995
10.201.5	Member Data Documentation	995
10.201.5.1	x	995
10.201.5.2	y	996
10.202	Gazebo::math::Vector2i Class Reference	996
10.202.1	Detailed Description	997
10.202.2	Constructor & Destructor Documentation	997
10.202.2.1	Vector2i	997
10.202.2.2	Vector2i	998
10.202.2.3	Vector2i	998
10.202.2.4	~Vector2i	998
10.202.3	Member Function Documentation	998
10.202.3.1	Cross	998
10.202.3.2	Distance	998
10.202.3.3	IsFinite	999
10.202.3.4	Normalize	999
10.202.3.5	operator!=	999
10.202.3.6	operator*	999
10.202.3.7	operator*	999
10.202.3.8	operator*==	1000
10.202.3.9	operator*==	1000
10.202.3.10	operator+	1000
10.202.3.11	operator+=	1000

10.202.3.10	operator-	1001
10.202.3.10	operator==	1001
10.202.3.10	operator/	1001
10.202.3.10	operator/	1002
10.202.3.10	operator/=	1002
10.202.3.10	operator/=	1002
10.202.3.10	operator=	1003
10.202.3.10	operator=	1003
10.202.3.20	operator==	1003
10.202.3.21	operator[]	1003
10.202.3.22	set	1003
10.202.4	Friends And Related Function Documentation	1004
10.202.4.1	operator<<	1004
10.202.4.2	operator>>	1004
10.202.5	Member Data Documentation	1004
10.202.5.1x		1004
10.202.5.2y		1004
10.203	Gazebo::math::Vector3 Class Reference	1004
10.203.1	Detailed Description	1007
10.203.2	Constructor & Destructor Documentation	1007
10.203.2.1	Vector3	1007
10.203.2.2	Vector3	1008
10.203.2.3	Vector3	1008
10.203.2.4	~Vector3	1008
10.203.3	Member Function Documentation	1008
10.203.3.1	Correct	1008
10.203.3.2	Cross	1008
10.203.3.3	Distance	1008
10.203.3.4	Distance	1009
10.203.3.5	Dot	1009
10.203.3.6	Equal	1009
10.203.3.7	GetAbs	1009
10.203.3.8	GetDistToLine	1010
10.203.3.9	GetLength	1010
10.203.3.10	GetMax	1010
10.203.3.10	GetMin	1010
10.203.3.10	GetNormal	1010

10.203.3.10	GetPerpendicular	1011
10.203.3.10	GetRounded	1011
10.203.3.10	GetSquaredLength	1011
10.203.3.10	GetSum	1011
10.203.3.11	Finite	1011
10.203.3.11	Normalize	1011
10.203.3.16	operator!=	1011
10.203.3.20	operator*	1012
10.203.3.21	operator*	1012
10.203.3.22	operator*==	1012
10.203.3.23	operator*==	1013
10.203.3.24	operator+	1013
10.203.3.25	operator+=	1013
10.203.3.26	operator-	1013
10.203.3.27	operator-	1013
10.203.3.28	operator-=	1014
10.203.3.29	operator/	1014
10.203.3.30	operator/	1014
10.203.3.31	operator/=	1014
10.203.3.32	operator/=	1015
10.203.3.33	operator=	1015
10.203.3.34	operator=	1015
10.203.3.35	operator==	1015
10.203.3.36	operator[]	1016
10.203.3.37	Round	1016
10.203.3.38	Round	1016
10.203.3.39	Set	1016
10.203.3.40	SetToMax	1016
10.203.3.41	SetToMin	1016
10.203.4	Friends And Related Function Documentation	1017
10.203.4.1	operator*	1017
10.203.4.2	operator<<	1017
10.203.4.3	operator>>	1017
10.203.5	Member Data Documentation	1017
10.203.5.1	One	1017
10.203.5.2	UnitX	1017
10.203.5.3	UnitY	1018

10.203.5.4	UnitZ	1018
10.203.5.5	x	1018
10.203.5.6	y	1018
10.203.5.7	z	1018
10.203.5.8	Zero	1018
10.204	Gazebo::math::Vector4 Class Reference	1018
10.204.1	Detailed Description	1020
10.204.2	Constructor & Destructor Documentation	1020
10.204.2.1	Vector4	1020
10.204.2.2	Vector4	1020
10.204.2.3	Vector4	1021
10.204.2.4	~Vector4	1021
10.204.3	Member Function Documentation	1021
10.204.3.1	Distance	1021
10.204.3.2	GetLength	1021
10.204.3.3	GetSquaredLength	1021
10.204.3.4	IsFinite	1021
10.204.3.5	Normalize	1021
10.204.3.6	operator!=	1022
10.204.3.7	operator*	1022
10.204.3.8	operator*	1022
10.204.3.9	operator*	1022
10.204.3.10	operator*=	1023
10.204.3.11	operator*=	1023
10.204.3.12	operator+	1023
10.204.3.13	operator+=	1023
10.204.3.14	operator-	1024
10.204.3.15	operator-=	1024
10.204.3.16	operator/	1024
10.204.3.17	operator/	1025
10.204.3.18	operator/=	1025
10.204.3.19	operator/=	1025
10.204.3.20	operator=	1025
10.204.3.21	operator=	1026
10.204.3.22	operator==	1026
10.204.3.23	operator[]	1026
10.204.3.24	set	1026

10.204.4	Friends And Related Function Documentation	1026
10.204.4.1	operator<<	1027
10.204.4.2	operator>>	1027
10.204.5	Member Data Documentation	1027
10.204.5.1	w	1027
10.204.5.2	x	1027
10.204.5.3	y	1027
10.204.5.4	z	1027
10.205	gazebo::common::Video Class Reference	1028
10.205.1	Detailed Description	1028
10.205.2	Constructor & Destructor Documentation	1028
10.205.2.1	Video	1028
10.205.2.2	~Video	1028
10.205.3	Member Function Documentation	1028
10.205.3.1	GetHeight	1028
10.205.3.2	GetNextFrame	1029
10.205.3.3	GetWidth	1029
10.205.3.4	Load	1029
10.206	gazebo::rendering::VideoVisual Class Reference	1029
10.206.1	Detailed Description	1030
10.206.2	Constructor & Destructor Documentation	1030
10.206.2.1	VideoVisual	1030
10.206.2.2	~VideoVisual	1031
10.207	gazebo::rendering::ViewController Class Reference	1031
10.207.1	Detailed Description	1032
10.207.2	Constructor & Destructor Documentation	1032
10.207.2.1	ViewController	1032
10.207.2.2	~ViewController	1032
10.207.3	Member Function Documentation	1032
10.207.3.1	GetTypeString	1032
10.207.3.2	HandleKeyPressEvent	1032
10.207.3.3	HandleKeyReleaseEvent	1033
10.207.3.4	HandleMouseEvent	1033
10.207.3.5	nit	1033
10.207.3.6	nit	1033
10.207.3.7	SetEnabled	1033
10.207.3.8	Update	1034

10.207.4	Member Data Documentation	1034
10.207.4.1	camera	1034
10.207.4.2	enabled	1034
10.207.4.3	typeString	1034
10.208	gazebo::rendering::Visual Class Reference	1034
10.208.1	Detailed Description	1040
10.208.2	Constructor & Destructor Documentation	1040
10.208.2.1	Visual	1040
10.208.2.2	Visual	1040
10.208.2.3	~Visual	1040
10.208.3	Member Function Documentation	1040
10.208.3.1	AttachAxes	1040
10.208.3.2	AttachLineVertex	1040
10.208.3.3	AttachMesh	1041
10.208.3.4	AttachObject	1041
10.208.3.5	AttachVisual	1041
10.208.3.6	ClearParent	1041
10.208.3.7	Clone	1041
10.208.3.8	CreateDynamicLine	1042
10.208.3.9	DeleteDynamicLine	1042
10.208.3.10	DetachObjects	1042
10.208.3.11	DetachVisual	1042
10.208.3.12	DetachVisual	1042
10.208.3.13	DisableTrackVisual	1042
10.208.3.14	EnableTrackVisual	1042
10.208.3.15	Ini	1043
10.208.3.16	GetAttachedObjectCount	1043
10.208.3.17	GetBoundingBox	1043
10.208.3.18	GetChild	1043
10.208.3.19	GetChildCount	1043
10.208.3.20	GetId	1043
10.208.3.21	GetMaterialName	1044
10.208.3.22	GetMeshName	1044
10.208.3.23	GetName	1044
10.208.3.24	GetNormalMap	1044
10.208.3.25	GetParent	1044
10.208.3.26	GetPose	1044

10.208.3.27	GetPosition	1045
10.208.3.28	GetRootVisual	1045
10.208.3.29	GetRotation	1045
10.208.3.30	GetScale	1045
10.208.3.31	GetScene	1045
10.208.3.32	GetSceneNode	1045
10.208.3.33	GetShaderType	1046
10.208.3.34	GetSubMeshName	1046
10.208.3.35	GetTransparency	1046
10.208.3.36	GetVisibilityFlags	1046
10.208.3.37	GetVisible	1046
10.208.3.38	GetWorldPose	1047
10.208.3.39	HasAttachedObject	1047
10.208.3.40	hit	1047
10.208.3.41	InsertMesh	1047
10.208.3.42	InsertMesh	1047
10.208.3.43	Plane	1047
10.208.3.44	Static	1048
10.208.3.45	Load	1048
10.208.3.46	Load	1048
10.208.3.47	LoadFromMsg	1048
10.208.3.48	LoadPlugin	1048
10.208.3.49	MakeStatic	1049
10.208.3.50	MoveToPosition	1049
10.208.3.51	MoveToPositions	1049
10.208.3.52	RemovePlugin	1049
10.208.3.53	SetAmbient	1049
10.208.3.54	SetCastShadows	1049
10.208.3.55	SetDiffuse	1050
10.208.3.56	SetEmissive	1050
10.208.3.57	SetHighlighted	1050
10.208.3.58	SetId	1050
10.208.3.59	SetMaterial	1050
10.208.3.60	SetName	1050
10.208.3.61	SetNormalMap	1051
10.208.3.62	SetPose	1051
10.208.3.63	SetPosition	1051

10.208.3.63	SetRibbonTrail	1051
10.208.3.65	SetRotation	1051
10.208.3.66	SetScale	1051
10.208.3.67	SetScene	1052
10.208.3.68	SetShaderType	1052
10.208.3.69	SetSkeletonPose	1052
10.208.3.70	SetSpecular	1052
10.208.3.73	SetTransparency	1052
10.208.3.72	SetVisibilityFlags	1052
10.208.3.75	SetVisible	1053
10.208.3.74	SetWireframe	1053
10.208.3.75	SetWorldPose	1053
10.208.3.76	SetWorldPosition	1053
10.208.3.77	SetWorldRotation	1053
10.208.3.78	ShowBoundingBox	1054
10.208.3.79	ShowCollision	1054
10.208.3.80	ShowCOM	1054
10.208.3.83	ShowJoints	1054
10.208.3.82	ShowSkeleton	1054
10.208.3.85	ToggleVisible	1054
10.208.3.84	Update	1054
10.208.3.85	UpdateFromMsg	1055
10.208.4	Member Data Documentation	1055
10.208.4.1	parent	1055
10.208.4.2	scene	1055
10.208.4.3	sceneNode	1055
10.209	gazebo::VisualPlugin Class Reference	1055
10.209.1	Detailed Description	1056
10.209.2	Constructor & Destructor Documentation	1056
10.209.2.1	VisualPlugin	1056
10.209.3	Member Function Documentation	1056
10.209.3.1	Init	1056
10.209.3.2	Load	1056
10.209.3.3	Reset	1056
10.210	gazebo::rendering::WindowManager Class Reference	1057
10.210.1	Detailed Description	1057
10.210.2	Constructor & Destructor Documentation	1057

10.210.2.1	WindowManager	1057
10.210.2.2	~WindowManager	1057
10.210.3	Member Function Documentation	1058
10.210.3.1	CreateWindow	1058
10.210.3.2	Fini	1058
10.210.3.3	GetAvgFPS	1058
10.210.3.4	GetTriangleCount	1058
10.210.3.5	GetWindow	1058
10.210.3.6	Moved	1059
10.210.3.7	Resize	1059
10.210.3.8	SetCamera	1059
10.210	gazebo::rendering::WireBox Class Reference	1059
10.211.1	Detailed Description	1060
10.211.2	Constructor & Destructor Documentation	1060
10.211.2.1	WireBox	1060
10.211.2.2	~WireBox	1060
10.211.3	Member Function Documentation	1060
10.211.3.1	Init	1060
10.211.3.2	SetVisible	1060
10.210	gazebo::sensors::WirelessReceiver Class Reference	1060
10.212.1	Detailed Description	1062
10.212.2	Constructor & Destructor Documentation	1062
10.212.2.1	WirelessReceiver	1062
10.212.2.2	~WirelessReceiver	1062
10.212.3	Member Function Documentation	1062
10.212.3.1	Fini	1062
10.212.3.2	GetMaxFreqFiltered	1062
10.212.3.3	GetMinFreqFiltered	1062
10.212.3.4	GetSensitivity	1062
10.212.3.5	Init	1063
10.212.3.6	Load	1063
10.210	gazebo::sensors::WirelessTransceiver Class Reference	1063
10.213.1	Detailed Description	1064
10.213.2	Constructor & Destructor Documentation	1064
10.213.2.1	WirelessTransceiver	1064
10.213.2.2	~WirelessTransceiver	1064
10.213.3	Member Function Documentation	1065

10.213.3.1	Finis	1065
10.213.3.2	GetGain	1065
10.213.3.3	GetPower	1065
10.213.3.4	GetTopic	1065
10.213.3.5	Init	1065
10.213.3.6	Load	1065
10.213.4	Member Data Documentation	1066
10.213.4.1	gain	1066
10.213.4.2	parentEntity	1066
10.213.4.3	power	1066
10.213.4.4	pub	1066
10.213.4.5	referencePose	1066
10.214	gazebo::sensors::WirelessTransmitter Class Reference	1066
10.214.1	Detailed Description	1068
10.214.2	Constructor & Destructor Documentation	1068
10.214.2.1	WirelessTransmitter	1068
10.214.2.2	~WirelessTransmitter	1068
10.214.3	Member Function Documentation	1068
10.214.3.1	GetESSID	1068
10.214.3.2	GetFreq	1068
10.214.3.3	GetSignalStrength	1069
10.214.3.4	Init	1069
10.214.3.5	Load	1069
10.214.3.6	UpdateImpl	1069
10.214.4	Member Data Documentation	1069
10.214.4.1	freq	1069
10.214.4.2	ModelStdDesv	1069
10.214.4.3	NEEmpty	1070
10.214.4.4	NObstacle	1070
10.215	gazebo::physics::World Class Reference	1070
10.215.1	Detailed Description	1073
10.215.2	Constructor & Destructor Documentation	1073
10.215.2.1	World	1073
10.215.2.2	~World	1073
10.215.3	Member Function Documentation	1073
10.215.3.1	Clear	1073
10.215.3.2	DisableAllModels	1073

10.215.3.3EnableAllModels	1073
10.215.3.4EnablePhysicsEngine	1073
10.215.3.5Fini	1074
10.215.3.6GetByName	1074
10.215.3.7GetEnablePhysicsEngine	1074
10.215.3.8GetEntity	1074
10.215.3.9GetEntityBelowPoint	1074
10.215.3.10GetModel	1075
10.215.3.10GetModel	1075
10.215.3.10GetModelBelowPoint	1075
10.215.3.10GetModelCount	1075
10.215.3.10GetModels	1076
10.215.3.10GetName	1076
10.215.3.10GetPauseTime	1076
10.215.3.10GetPhysicsEngine	1076
10.215.3.10GetRealTime	1076
10.215.3.10GetRunning	1077
10.215.3.20GetSelectedEntity	1077
10.215.3.20GetSetWorldPoseMutex	1077
10.215.3.20GetSimTime	1077
10.215.3.20GetSphericalCoordinates	1077
10.215.3.20GetStartTime	1077
10.215.3.25it	1078
10.215.3.26InsertModelFile	1078
10.215.3.27InsertModelSDF	1078
10.215.3.26InsertModelString	1078
10.215.3.29Loaded	1078
10.215.3.30Paused	1078
10.215.3.31load	1079
10.215.3.32LoadPlugin	1079
10.215.3.33PrintEntityTree	1079
10.215.3.34PublishModelPose	1079
10.215.3.35RemovePlugin	1079
10.215.3.36Reset	1079
10.215.3.37ResetEntities	1080
10.215.3.38ResetTime	1080
10.215.3.39Run	1080

10.215.3.4	Save	1080
10.215.3.4	SetPaused	1080
10.215.3.4	SetSimTime	1080
10.215.3.4	SetState	1081
10.215.3.4	StepWorld	1081
10.215.3.4	Stop	1081
10.215.3.4	StripWorldName	1081
10.215.3.4	UpdateStateSDF	1081
10.215.4	Member Data Documentation	1081
10.215.4.1	dirtyPoses	1081
10.216	Gazebo::WorldPlugin Class Reference	1082
10.216.1	Detailed Description	1082
10.216.2	Constructor & Destructor Documentation	1082
10.216.2.1	WorldPlugin	1082
10.216.2.2	~WorldPlugin	1083
10.216.3	Member Function Documentation	1083
10.216.3.1	Init	1083
10.216.3.2	Load	1083
10.216.3.3	Reset	1083
10.217	Gazebo::physics::WorldState Class Reference	1083
10.217.1	Detailed Description	1085
10.217.2	Constructor & Destructor Documentation	1085
10.217.2.1	WorldState	1085
10.217.2.2	WorldState	1085
10.217.2.3	WorldState	1085
10.217.2.4	~WorldState	1085
10.217.3	Member Function Documentation	1085
10.217.3.1	FillSDF	1085
10.217.3.2	GetModelState	1086
10.217.3.3	GetModelStateCount	1086
10.217.3.4	GetModelStates	1086
10.217.3.5	GetModelStates	1086
10.217.3.6	HasModelState	1086
10.217.3.7	IsZero	1087
10.217.3.8	Load	1087
10.217.3.9	Load	1087
10.217.3.10	operator+	1087

10.217.3.1operator-	1088
10.217.3.1operator=	1088
10.217.3.1SetRealTime	1088
10.217.3.1SetSimTime	1088
10.217.3.1SetWallTime	1088
10.217.3.1SetWorld	1089
10.217.4Friends And Related Function Documentation	1089
10.217.4.1operator<<	1089
10.218gazebo::rendering::WrenchVisual Class Reference	1089
10.218.1Detailed Description	1090
10.218.2Constructor & Destructor Documentation	1090
10.218.2.1WrenchVisual	1090
10.218.2.2~WrenchVisual	1091
10.218.3Member Function Documentation	1091
10.218.3.1Load	1091
10.218.3.2SetEnabled	1091
11 File Documentation	1093
11.1 Actor.hh File Reference	1093
11.2 Angle.hh File Reference	1094
11.2.1 Macro Definition Documentation	1096
11.2.1.1 GZ_DTOR	1096
11.2.1.2 GZ_NORMALIZE	1096
11.2.1.3 GZ_RTOD	1096
11.3 Animation.hh File Reference	1097
11.4 ArrowVisual.hh File Reference	1098
11.5 Assert.hh File Reference	1099
11.5.1 Macro Definition Documentation	1100
11.5.1.1 GZ_ASSERT	1100
11.6 AudioDecoder.hh File Reference	1101
11.7 AxisVisual.hh File Reference	1102
11.8 BallJoint.hh File Reference	1102
11.9 Base.hh File Reference	1103
11.10Base64.hh File Reference	1104
11.10.1 Function Documentation	1106
11.10.1.1 Base64Decode	1106
11.10.1.2 Base64Encode	1106

11.11Box.hh File Reference	1106
11.12BoxShape.hh File Reference	1107
11.13BVHLoader.hh File Reference	1108
11.13.1 Macro Definition Documentation	1110
11.13.1.1 X_POSITION	1110
11.13.1.2 X_ROTATION	1110
11.13.1.3 Y_POSITION	1110
11.13.1.4 Y_ROTATION	1110
11.13.1.5 Z_POSITION	1110
11.13.1.6 Z_ROTATION	1110
11.14CallbackHelper.hh File Reference	1110
11.15Camera.hh File Reference	1112
11.16CameraSensor.hh File Reference	1113
11.17CameraVisual.hh File Reference	1113
11.18cegui.h File Reference	1114
11.19ColladaLoader.hh File Reference	1115
11.20Collision.hh File Reference	1116
11.21CollisionState.hh File Reference	1117
11.22Color.hh File Reference	1117
11.23Commonface.hh File Reference	1118
11.24CommonTypes.hh File Reference	1120
11.24.1 Macro Definition Documentation	1122
11.24.1.1 GAZEBO_DEPRECATED	1122
11.24.1.2 GAZEBO_FORCEINLINE	1122
11.24.1.3 NULL	1122
11.25COMVisual.hh File Reference	1122
11.26Connection.hh File Reference	1123
11.26.1 Macro Definition Documentation	1125
11.26.1.1 HEADER_LENGTH	1125
11.27ConnectionManager.hh File Reference	1125
11.28Console.hh File Reference	1127
11.29Contact.hh File Reference	1128
11.29.1 Macro Definition Documentation	1129
11.29.1.1 MAX_COLLIDE_RETURNS	1129
11.29.1.2 MAX_CONTACT_JOINTS	1129
11.30ContactManager.hh File Reference	1130
11.31ContactSensor.hh File Reference	1131

11.32ContactVisual.hh File Reference	1131
11.33Conversions.hh File Reference	1132
11.34CylinderShape.hh File Reference	1133
11.35DepthCamera.hh File Reference	1134
11.36DepthCameraSensor.hh File Reference	1135
11.37Diagnostics.hh File Reference	1136
11.38DynamicLines.hh File Reference	1137
11.39DynamicRenderable.hh File Reference	1137
11.40Entity.hh File Reference	1138
11.41Event.hh File Reference	1139
11.42Events.hh File Reference	1140
11.43Exception.hh File Reference	1141
11.44ForceTorqueSensor.hh File Reference	1143
11.45FPSViewController.hh File Reference	1143
11.46gazebo.hh File Reference	1144
11.47gazebo_core.hh File Reference	1145
11.48GazeboGenerator.hh File Reference	1146
11.49GpsSensor.hh File Reference	1147
11.50GpuLaser.hh File Reference	1148
11.51GpuRaySensor.hh File Reference	1148
11.52Grid.hh File Reference	1149
11.53Gripper.hh File Reference	1150
11.54GUIOverlay.hh File Reference	1151
11.55Heightmap.hh File Reference	1151
11.56HeightmapShape.hh File Reference	1152
11.57Helpers.hh File Reference	1153
11.57.1 Macro Definition Documentation	1155
11.57.1.1 GZ_DBL_MAX	1155
11.57.1.2 GZ_DBL_MIN	1155
11.57.1.3 GZ_FLT_MAX	1156
11.57.1.4 GZ_FLT_MIN	1156
11.57.1.5 GZ_UINT32_MAX	1156
11.57.1.6 GZ_UINT32_MIN	1156
11.58Hinge2Joint.hh File Reference	1156
11.59HingeJoint.hh File Reference	1157
11.60Image.hh File Reference	1159
11.61ImuSensor.hh File Reference	1160

11.62Inertial.hh File Reference	1160
11.63IOManager.hh File Reference	1161
11.64Joint.hh File Reference	1163
11.64.1 Macro Definition Documentation	1164
11.64.1.1 MAX_JOINT_AXIS	1164
11.65Joint_TEST.hh File Reference	1164
11.65.1 Typedef Documentation	1165
11.65.1.1 std_string2	1165
11.66JointController.hh File Reference	1165
11.67JointState.hh File Reference	1166
11.68JointVisual.hh File Reference	1167
11.69JointWrench.hh File Reference	1167
11.70KeyEvent.hh File Reference	1169
11.71KeyFrame.hh File Reference	1169
11.72LaserVisual.hh File Reference	1171
11.73Light.hh File Reference	1171
11.74Link.hh File Reference	1172
11.75LinkState.hh File Reference	1173
11.76LogPlay.hh File Reference	1175
11.77LogRecord.hh File Reference	1175
11.77.1 Macro Definition Documentation	1176
11.77.1.1 GZ_LOG_VERSION	1176
11.78mainpage.html File Reference	1176
11.79MapShape.hh File Reference	1176
11.80Master.hh File Reference	1178
11.81Material.hh File Reference	1179
11.82Material.hh File Reference	1180
11.83MathTypes.hh File Reference	1180
11.83.1 Detailed Description	1180
11.84Matrix3.hh File Reference	1181
11.85Matrix4.hh File Reference	1181
11.86Mesh.hh File Reference	1182
11.87MeshCSG.hh File Reference	1184
11.87.1 Typedef Documentation	1185
11.87.1.1 GPtrArray	1185
11.87.1.2 GtsSurface	1185
11.88MeshLoader.hh File Reference	1186

11.89	MeshManager.hh File Reference	1187
11.90	MeshShape.hh File Reference	1188
11.91	Model.hh File Reference	1190
11.92	ModelDatabase.hh File Reference	1191
11.92.1	Macro Definition Documentation	1192
11.92.1.1	GZ_MODEL_DB_MANIFEST_FILENAME	1192
11.92.1.2	GZ_MODEL_MANIFEST_FILENAME	1192
11.93	ModelState.hh File Reference	1192
11.94	MouseEvent.hh File Reference	1193
11.95	MovableText.hh File Reference	1195
11.96	MsgFactory.hh File Reference	1195
11.97	msgs.hh File Reference	1196
11.98	MultiCameraSensor.hh File Reference	1199
11.99	MultiRayShape.hh File Reference	1199
11.100	Node.hh File Reference	1200
11.101	Noise.hh File Reference	1202
11.102	ogre_gazebo.h File Reference	1202
11.103	OpenAL.hh File Reference	1204
11.104	OrbitViewController.hh File Reference	1204
11.105	PhysicsEngine.hh File Reference	1205
11.106	PhysicsFactory.hh File Reference	1206
11.107	PhysicsIface.hh File Reference	1207
11.108	PhysicsTypes.hh File Reference	1209
11.108.1	Detailed Description	1212
11.108.2	Macro Definition Documentation	1212
11.108.2.1	GZ_ALL_COLLIDE	1212
11.108.2.2	GZ_FIXED_COLLIDE	1212
11.108.2.3	GZ_GHOST_COLLIDE	1212
11.108.2.4	GZ_NONE_COLLIDE	1212
11.108.2.5	GZ_SENSOR_COLLIDE	1212
11.109	ID.hh File Reference	1212
11.110	Plane.hh File Reference	1213
11.111	PlaneShape.hh File Reference	1214
11.112	Plugin.hh File Reference	1215
11.112.1	Macro Definition Documentation	1217
11.112.1.1	GZ_REGISTER_MODEL_PLUGIN	1217
11.112.1.2	GZ_REGISTER_SENSOR_PLUGIN	1218

11.112.1.3GZ_REGISTER_SYSTEM_PLUGIN	1218
11.112.1.4GZ_REGISTER_VISUAL_PLUGIN	1218
11.112.1.5GZ_REGISTER_WORLD_PLUGIN	1219
11.111Pose.hh File Reference	1219
11.111Projector.hh File Reference	1220
11.111Publication.hh File Reference	1220
11.111PublicationTransport.hh File Reference	1223
11.111Publisher.hh File Reference	1225
11.111Quaternion.hh File Reference	1227
11.111Rand.hh File Reference	1228
11.120RaySensor.hh File Reference	1229
11.120RayShape.hh File Reference	1230
11.120RenderEngine.hh File Reference	1231
11.120RenderEvents.hh File Reference	1232
11.120RenderingIface.hh File Reference	1233
11.120RenderTypes.hh File Reference	1234
11.125. Macro Definition Documentation	1236
11.125.1.1GZ_VISIBILITY_ALL	1236
11.125.1.2GZ_VISIBILITY_GUI	1236
11.125.1.3GZ_VISIBILITY_SELECTABLE	1236
11.125.1.4GZ_VISIBILITY_SELECTION	1236
11.120RFIDSensor.hh File Reference	1236
11.120RFIDTag.hh File Reference	1237
11.120RFIDTagVisual.hh File Reference	1237
11.120RFIDVisual.hh File Reference	1238
11.130Road.hh File Reference	1239
11.130Road2d.hh File Reference	1240
11.130RotationSpline.hh File Reference	1240
11.130RTShaderSystem.hh File Reference	1242
11.130Scene.hh File Reference	1242
11.130ScrewJoint.hh File Reference	1244
11.130SelectionObj.hh File Reference	1245
11.130Sensor.hh File Reference	1245
11.130SensorFactory.hh File Reference	1247
11.130SensorManager.hh File Reference	1248
11.140SensorsIface.hh File Reference	1248
11.140SensorTypes.hh File Reference	1250

11.141. Detailed Description	1252
11.142 Server.hh File Reference	1252
11.143 Shape.hh File Reference	1253
11.144 Simbody_inc.h File Reference	1254
11.145 SimbodyBallJoint.hh File Reference	1254
11.146 SimbodyBoxShape.hh File Reference	1255
11.147 SimbodyCollision.hh File Reference	1255
11.148 SimbodyCylinderShape.hh File Reference	1256
11.149 SimbodyHeightmapShape.hh File Reference	1257
11.150 SimbodyHinge2Joint.hh File Reference	1257
11.151 SimbodyHingeJoint.hh File Reference	1258
11.152 SimbodyJoint.hh File Reference	1259
11.153 SimbodyLink.hh File Reference	1259
11.154 SimbodyMeshShape.hh File Reference	1260
11.155 SimbodyModel.hh File Reference	1261
11.156 SimbodyMultiRayShape.hh File Reference	1261
11.157 SimbodyPhysics.hh File Reference	1262
11.158 SimbodyPlaneShape.hh File Reference	1263
11.159 SimbodyRayShape.hh File Reference	1263
11.160 SimbodyScrewJoint.hh File Reference	1264
11.161 SimbodySliderJoint.hh File Reference	1265
11.162 SimbodySphereShape.hh File Reference	1265
11.163 SimbodyTypes.hh File Reference	1266
11.163. Detailed Description	1267
11.164 SimbodyUniversalJoint.hh File Reference	1267
11.165 SingletonT.hh File Reference	1268
11.166 Skeleton.hh File Reference	1268
11.167 SkeletonAnimation.hh File Reference	1270
11.168 SliderJoint.hh File Reference	1271
11.169 SonarSensor.hh File Reference	1272
11.170 SonarVisual.hh File Reference	1273
11.171 SphereShape.hh File Reference	1274
11.172 SphericalCoordinates.hh File Reference	1275
11.173 Spline.hh File Reference	1276
11.174 State.hh File Reference	1278
11.175 STLLoader.hh File Reference	1279
11.175. Macro Definition Documentation	1280

11.175.1.1COR3_MAX	1280
11.175.1.2FACE_MAX	1280
11.175.1.3LINE_MAX_LEN	1280
11.175.1.4ORDER_MAX	1280
11.176SubscribeOptions.hh File Reference	1280
11.177Subscriber.hh File Reference	1282
11.178SubscriptionTransport.hh File Reference	1284
11.179SurfaceParams.hh File Reference	1286
11.180SystemPaths.hh File Reference	1287
11.180.1Macro Definition Documentation	1288
11.180.1.1GetCurrentDir	1288
11.180.1.2LINUX	1288
11.181Time.hh File Reference	1288
11.182Timer.hh File Reference	1289
11.183TopicManager.hh File Reference	1290
11.184TransmitterVisual.hh File Reference	1292
11.185TransportIface.hh File Reference	1293
11.186TransportTypes.hh File Reference	1295
11.186.1Detailed Description	1296
11.187UniversalJoint.hh File Reference	1296
11.188UpdateInfo.hh File Reference	1297
11.189UserCamera.hh File Reference	1298
11.190UtilTypes.hh File Reference	1299
11.191Vector2d.hh File Reference	1300
11.192Vector2i.hh File Reference	1301
11.193Vector3.hh File Reference	1302
11.194Vector4.hh File Reference	1303
11.195Video.hh File Reference	1305
11.196VideoVisual.hh File Reference	1306
11.197ViewController.hh File Reference	1307
11.198Visual.hh File Reference	1308
11.199WindowManager.hh File Reference	1309
11.200WireBox.hh File Reference	1310
11.201WirelessReceiver.hh File Reference	1311
11.202WirelessTransceiver.hh File Reference	1311
11.203WirelessTransmitter.hh File Reference	1312
11.204World.hh File Reference	1313

11.205	WorldState.hh File Reference	1314
11.206	WrenchVisual.hh File Reference	1316

Index	1316
--------------	-------------

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. 419) (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::sensors::CameraSensor::GetTopic` (p. 208) () const

to be implemented

Class `gazebo::SystemPlugin` (p. 942)

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	31
Events	43
Math	46
Messages	52
Classes for physics and dynamics	63
Simbody Physics	71
Rendering	73
Sensors	80
Transport	85
Utility	91

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

boost	93
gazebo	
Forward declarations for the common classes	93
gazebo::common	
Common namespace	95
gazebo::event	
Event (p. 305) namespace	99
gazebo::math	
Math namespace	99
gazebo::msgs	
Messages namespace	102
gazebo::physics	
Namespace for physics	104
gazebo::rendering	
Rendering namespace	111
gazebo::sensors	
Sensors namespace	115
gazebo::transport	119
gazebo::util	121
google	122
google::protobuf	122
google::protobuf::compiler	122
google::protobuf::compiler::cpp	123
Ogre	123
ogre	123
SimTK	123
SkyX	123

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

gazebo::math::Angle	131
gazebo::common::Animation	139
gazebo::common::NumericAnimation	606
gazebo::common::PoseAnimation	657
gazebo::common::AudioDecoder	147
gazebo::math::Box	164
gazebo::common::BVHLoader	172
gazebo::transport::CallbackHelper	173
gazebo::transport::CallbackHelperT< M >	177
gazebo::transport::RawCallbackHelper	690
gazebo::transport::SubscriptionTransport	930
CodeGenerator	
google::protobuf::compiler::cpp::GazeboGenerator	340
gazebo::common::Color	226
gazebo::event::Connection	239
gazebo::physics::Contact	253
gazebo::physics::ContactManager	256
gazebo::physics::ContactPublisher	259
gazebo::rendering::Conversions	266
enable_shared_from_this	
gazebo::physics::Base	153
gazebo::physics::Entity	293
gazebo::physics::Collision	213
gazebo::physics::SimbodyCollision	785
gazebo::physics::Link	455
gazebo::physics::SimbodyLink	813
gazebo::physics::Model	537
gazebo::physics::Actor	125
gazebo::physics::SimbodyModel	825
gazebo::physics::Joint	411
gazebo::physics::SimbodyJoint	805
gazebo::physics::BallJoint< SimbodyJoint >	151
gazebo::physics::SimbodyBallJoint	778

gazebo::physics::Hinge2Joint< SimbodyJoint >	386
gazebo::physics::SimbodyHinge2Joint	792
gazebo::physics::HingeJoint< SimbodyJoint >	387
gazebo::physics::SimbodyHingeJoint	799
gazebo::physics::ScrewJoint< SimbodyJoint >	746
gazebo::physics::SimbodyScrewJoint	842
gazebo::physics::SliderJoint< SimbodyJoint >	886
gazebo::physics::SimbodySliderJoint	849
gazebo::physics::UniversalJoint< SimbodyJoint >	976
gazebo::physics::SimbodyUniversalJoint	857
gazebo::physics::Road	718
gazebo::physics::Shape	775
gazebo::physics::BoxShape	169
gazebo::physics::SimbodyBoxShape	784
gazebo::physics::CylinderShape	268
gazebo::physics::SimbodyCylinderShape	788
gazebo::physics::HeightmapShape	380
gazebo::physics::SimbodyHeightmapShape	790
gazebo::physics::MapShape	492
gazebo::physics::MeshShape	534
gazebo::physics::SimbodyMeshShape	823
gazebo::physics::MultiRayShape	578
gazebo::physics::SimbodyMultiRayShape	826
gazebo::physics::PlaneShape	642
gazebo::physics::SimbodyPlaneShape	837
gazebo::physics::RayShape	700
gazebo::physics::SimbodyRayShape	839
gazebo::physics::SphereShape	898
gazebo::physics::SimbodySphereShape	855
gazebo::physics::World	1070
gazebo::rendering::Camera	179
gazebo::rendering::DepthCamera	272
gazebo::rendering::GpuLaser	344
gazebo::rendering::UserCamera	979
gazebo::rendering::Scene	728
gazebo::rendering::Visual	1034
gazebo::rendering::ArrowVisual	143
gazebo::rendering::AxisVisual	149
gazebo::rendering::CameraVisual	210
gazebo::rendering::COMVisual	237
gazebo::rendering::ContactVisual	265
gazebo::rendering::JointVisual	441
gazebo::rendering::LaserVisual	447
gazebo::rendering::RFIDTagVisual	715
gazebo::rendering::RFIDVisual	716
gazebo::rendering::SelectionObj	749
gazebo::rendering::SonarVisual	895
gazebo::rendering::TransmitterVisual	975
gazebo::rendering::VideoVisual	1029
gazebo::rendering::WrenchVisual	1089
gazebo::sensors::Sensor	751
gazebo::sensors::CameraSensor	206

gazebo::sensors::ContactSensor	260
gazebo::sensors::DepthCameraSensor	276
gazebo::sensors::ForceTorqueSensor	333
gazebo::sensors::GpsSensor	341
gazebo::sensors::GpuRaySensor	353
gazebo::sensors::ImuSensor	395
gazebo::sensors::MultiCameraSensor	573
gazebo::sensors::RaySensor	693
gazebo::sensors::RFIDSensor	710
gazebo::sensors::RFIDTag	712
gazebo::sensors::SonarSensor	890
gazebo::sensors::WirelessTransceiver	1063
gazebo::sensors::WirelessReceiver	1060
gazebo::sensors::WirelessTransmitter	1066
gazebo::transport::Connection	240
gazebo::transport::Node	587
gazebo::event::Event	305
gazebo::event::EventT< void() >	322
gazebo::event::EventT< void(bool) >	322
gazebo::event::EventT< void(const common::UpdateInfo &) >	322
gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &) >	322
gazebo::event::EventT< void(const float *_frame, unsigned int _width, unsigned int _height, unsigned int _depth, const std::string &_format) >	322
gazebo::event::EventT< void(const std::string &) >	322
gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &) >	322
gazebo::event::EventT< void(msgs::SonarStamped) >	322
gazebo::event::EventT< void(msgs::WrenchStamped) >	322
gazebo::event::EventT< void(std::string) >	322
gazebo::event::EventT< void(std::string, std::string) >	322
gazebo::event::EventT< T >	322
gazebo::rendering::Events	308
gazebo::event::Events	310
gazebo::common::Exception	331
gazebo::common::InternalError	408
gazebo::common::AssertionInternalError	145
gazebo::rendering::Grid	365
gazebo::physics::Gripper	368
gazebo::rendering::GUIOverlay	370
gazebo::rendering::Heightmap	375
gazebo::common::Image	389
gazebo::physics::Inertial	398
gazebo::transport::IOManager	409
gazebo::physics::JointController	434
gazebo::physics::JointWrench	442
gazebo::common::KeyEvent	444
gazebo::common::KeyFrame	445
gazebo::common::NumericKeyFrame	607
gazebo::common::PoseKeyFrame	660
gazebo::rendering::Light	448
Logplay	486
gazebo::Master	496
gazebo::common::Material	498

gazebo::math::Matrix3	507
gazebo::math::Matrix4	512
gazebo::common::Mesh	519
gazebo::common::MeshCSG	526
gazebo::common::MeshLoader	527
gazebo::common::ColladaLoader	211
gazebo::common::STLLoader	915
gazebo::common::MouseEvent	563
MovableObject	
gazebo::rendering::MovableText	566
gazebo::msgs::MsgFactory	572
gazebo::common::NodeAnimation	593
gazebo::common::NodeAssignment	597
gazebo::common::NodeTransform	598
gazebo::sensors::Noise	603
gazebo::util::OpenALSink	611
gazebo::util::OpenALSource	612
PageProvider	
gazebo::rendering::DummyPageProvider	284
gazebo::common::ParamT< T >	620
gazebo::physics::PhysicsEngine	620
gazebo::physics::SimbodyPhysics	828
gazebo::physics::PhysicsFactory	634
gazebo::common::PID	636
gazebo::math::Plane	640
gazebo::PluginT< T >	646
gazebo::PluginT< ModelPlugin >	646
gazebo::ModelPlugin	553
gazebo::PluginT< SensorPlugin >	646
gazebo::SensorPlugin	767
gazebo::PluginT< SystemPlugin >	646
gazebo::SystemPlugin	942
gazebo::PluginT< VisualPlugin >	646
gazebo::VisualPlugin	1055
gazebo::PluginT< WorldPlugin >	646
gazebo::WorldPlugin	1082
gazebo::math::Pose	648
gazebo::rendering::Projector	662
gazebo::transport::Publication	664
gazebo::transport::PublicationTransport	669
gazebo::transport::Publisher	671
QuadNode	674
gazebo::math::Quaternion	675
gazebo::math::Rand	688
Renderable	
gazebo::rendering::MovableText	566
RenderObjectListener	
gazebo::rendering::GpuLaser	344
Road	718
gazebo::rendering::Road2d	720
gazebo::math::RotationSpline	720
SensorFactor	762

gazebo::sensors::SensorFactory	762
gazebo::Server	769
ServerFixture	
Joint_TEST	430
ShaderHelperCg	
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg	771
ShaderHelperGLSL	
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL	773
SimpleRenderable	
gazebo::rendering::DynamicRenderable	289
gazebo::rendering::DynamicLines	286
SingletonT< T >	864
gazebo::common::Console	252
gazebo::common::MeshManager	529
gazebo::common::ModelDatabase	551
gazebo::common::SystemPaths	937
gazebo::rendering::RenderEngine	706
gazebo::rendering::RTShaderSystem	724
gazebo::sensors::SensorManager	764
gazebo::transport::ConnectionManager	247
gazebo::transport::TopicManager	967
gazebo::util::DiagnosticManager	279
gazebo::util::LogPlay	483
gazebo::util::LogRecord	486
gazebo::util::OpenAL	609
SingletonT< ConnectionManager >	864
SingletonT< Console >	864
SingletonT< DiagnosticManager >	864
SingletonT< LogPlay >	864
SingletonT< LogRecord >	864
SingletonT< MeshManager >	864
SingletonT< ModelDatabase >	864
SingletonT< OpenAL >	864
SingletonT< RenderEngine >	864
SingletonT< RTShaderSystem >	864
SingletonT< SensorManager >	864
SingletonT< SystemPaths >	864
SingletonT< TopicManager >	864
gazebo::common::Skeleton	866
gazebo::common::SkeletonAnimation	873
gazebo::common::SkeletonNode	877
SM2Profile	
gazebo::rendering::GzTerrainMatGen::SM2Profile	888
Joint_TEST::SpawnJointOptions	896
gazebo::common::SphericalCoordinates	901
gazebo::math::Spline	906
gazebo::physics::State	910
gazebo::physics::CollisionState	222
gazebo::physics::JointState	436
gazebo::physics::LinkState	476
gazebo::physics::ModelState	554
gazebo::physics::WorldState	1083
gazebo::common::SubMesh	916
gazebo::transport::SubscribeOptions	927

gazebo::transport::Subscriber	929
gazebo::physics::SurfaceParams	933
TerrainMaterialGeneratorA	
gazebo::rendering::GzTerrainMatGen	374
gazebo::common::Time	944
gazebo::common::Timer	965
gazebo::util::DiagnosticTimer	283
gazebo::physics::TrajectoryInfo	974
gazebo::common::UpdateInfo	978
gazebo::math::Vector2d	987
gazebo::math::Vector2i	996
gazebo::math::Vector3	1004
gazebo::math::Vector4	1018
gazebo::common::Video	1028
gazebo::rendering::ViewController	1031
gazebo::rendering::FPSViewController	337
gazebo::rendering::OrbitViewController	617
gazebo::rendering::WindowManager	1057
gazebo::rendering::WireBox	1059
WithParamInterface	
Joint_TEST	430
T	
gazebo::physics::BallJoint< T >	151
gazebo::physics::Hinge2Joint< T >	386
gazebo::physics::HingeJoint< T >	387
gazebo::physics::ScrewJoint< T >	746
gazebo::physics::SliderJoint< T >	886
gazebo::physics::UniversalJoint< T >	976

Chapter 6

Class Index

6.1 Class List

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

gazebo::physics::Actor	
Actor (p. 125) class enables GPU based mesh model / skeleton scriptable animation	125
gazebo::math::Angle	
An angle and related functions	131
gazebo::common::Animation	
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes	139
gazebo::rendering::ArrowVisual	
Basic arrow visualization	143
gazebo::common::AssertionInternalError	
Class for generating Exceptions which come from gazebo assertions	145
gazebo::common::AudioDecoder	
An audio decoder based on FFMPEG	147
gazebo::rendering::AxisVisual	
Basic axis visualization	149
gazebo::physics::BallJoint< T >	
Base (p. 153) class for a ball joint	151
gazebo::physics::Base	
Base (p. 153) class for most physics classes	153
gazebo::math::Box	
Mathematical representation of a box and related functions	164
gazebo::physics::BoxShape	
Box geometry primitive	169
gazebo::common::BVHLoader	
Handles loading BVH animation files	172
gazebo::transport::CallbackHelper	
A helper class to handle callbacks when messages arrive	173
gazebo::transport::CallbackHelperT< M >	
Callback helper Template	177
gazebo::rendering::Camera	
Basic camera sensor	179
gazebo::sensors::CameraSensor	
Basic camera sensor	206

gazebo::rendering::CameraVisual	
Basic camera visualization	210
gazebo::common::ColladaLoader	
Class used to load Collada mesh files	211
gazebo::physics::Collision	
Base (p. 153) class for all collision entities	213
gazebo::physics::CollisionState	
Store state information of a physics::Collision (p. 213) object	222
gazebo::common::Color	
Defines a color	226
gazebo::rendering::COMVisual	
Basic Center of Mass visualization	237
gazebo::event::Connection	
A class that encapsulates a connection	239
gazebo::transport::Connection	
Single TCP/IP connection manager	240
gazebo::transport::ConnectionManager	
Manager of connections	247
gazebo::common::Console	
Message, error, warning functionality	252
gazebo::physics::Contact	
A contact between two collisions	253
gazebo::physics::ContactManager	
Aggregates all the contact information generated by the collision detection engine	256
gazebo::physics::ContactPublisher	
A custom contact publisher created for each contact filter in the Contact (p. 253) Manager	259
gazebo::sensors::ContactSensor	
Contact sensor	260
gazebo::rendering::ContactVisual	
Contact visualization	265
gazebo::rendering::Conversions	
Conversions (p. 266) Conversions.hh (p. 1132) rendering/Conversions.hh (p. 1132)	266
gazebo::physics::CylinderShape	
Cylinder collision	268
gazebo::rendering::DepthCamera	
Depth camera used to render depth data into an image buffer	272
gazebo::sensors::DepthCameraSensor	276
gazebo::util::DiagnosticManager	
A diagnostic manager class	279
gazebo::util::DiagnosticTimer	
A timer designed for diagnostics	283
gazebo::rendering::DummyPageProvider	
Pretends to provide procedural page content to avoid page loading	284
gazebo::rendering::DynamicLines	
Class for drawing lines that can change	286
gazebo::rendering::DynamicRenderable	
Abstract base class providing mechanisms for dynamically growing hardware buffers	289
gazebo::physics::Entity	
Base (p. 153) class for all physics objects in Gazebo	293
gazebo::event::Event	
Base class for all events	305
gazebo::rendering::Events	
Base class for rendering events	308

gazebo::event::Events	
An Event (p. 305) class to get notifications for simulator events	310
gazebo::event::EventT< T >	
A class for event processing	322
gazebo::common::Exception	
Class for generating exceptions	331
gazebo::sensors::ForceTorqueSensor	
Sensor (p. 751) for measure force and torque on a joint	333
gazebo::rendering::FPSViewController	
First Person Shooter style view controller	337
google::protobuf::compiler::cpp::GazeboGenerator	
Google protobuf message generator for gazebo::msgs (p. 102)	340
gazebo::sensors::GpsSensor	
GpsSensor (p. 341) to provide position measurement	341
gazebo::rendering::GpuLaser	
GPU based laser distance sensor	344
gazebo::sensors::GpuRaySensor	353
gazebo::rendering::Grid	
Displays a grid of cells, drawn with lines	365
gazebo::physics::Gripper	
A gripper abstraction	368
gazebo::rendering::GUIOverlay	
A class that creates a CEGUI overlay on a render window	370
gazebo::rendering::GzTerrainMatGen	374
gazebo::rendering::Heightmap	
Rendering a terrain using heightmap information	375
gazebo::physics::HeightmapShape	
HeightmapShape (p. 380) collision shape builds a heightmap from an image	380
gazebo::physics::Hinge2Joint< T >	
A two axis hinge joint	386
gazebo::physics::HingeJoint< T >	
A single axis hinge joint	387
gazebo::common::Image	
Encapsulates an image	389
gazebo::sensors::ImuSensor	
An IMU sensor	395
gazebo::physics::Inertial	
A class for inertial information about a link	398
gazebo::common::InternalError	
Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs	408
gazebo::transport::IOManager	
Manages boost::asio IO	409
gazebo::physics::Joint	
Base (p. 153) class for all joints	411
Joint_TEST	430
gazebo::physics::JointController	
A class for manipulating physics::Joint (p. 411)	434
gazebo::physics::JointState	
Keeps track of state of a physics::Joint (p. 411)	436
gazebo::rendering::JointVisual	
Visualization for joints	441
gazebo::physics::JointWrench	
Wrench information from a joint	442

gazebo::common::KeyEvent	Generic description of a keyboard event	444
gazebo::common::KeyFrame	A key frame in an animation	445
gazebo::rendering::LaserVisual	Visualization for laser data	447
gazebo::rendering::Light	A light source	448
gazebo::physics::Link	Link (p. 455) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body	455
gazebo::physics::LinkState	Store state information of a physics::Link (p. 455) object	476
gazebo::util::LogPlay		483
Logplay	Open and playback log files that were recorded using LogRecord	486
gazebo::util::LogRecord	Addtogroup gazebo_util	486
gazebo::physics::MapShape	Creates box extrusions based on an image	492
gazebo::Master	A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication	496
gazebo::common::Material	Encapsulates description of a material	498
gazebo::math::Matrix3	A 3x3 matrix class	507
gazebo::math::Matrix4	A 3x3 matrix class	512
gazebo::common::Mesh	A 3D mesh	519
gazebo::common::MeshCSG	Creates CSG meshes	526
gazebo::common::MeshLoader	Base class for loading meshes	527
gazebo::common::MeshManager	Maintains and manages all meshes	529
gazebo::physics::MeshShape	Triangle mesh collision shape	534
gazebo::physics::Model	A model is a collection of links, joints, and plugins	537
gazebo::common::ModelDatabase	Connects to model database, and has utility functions to find models	551
gazebo::ModelPlugin	A plugin with access to physics::Model (p. 537)	553
gazebo::physics::ModelState	Store state information of a physics::Model (p. 537) object	554
gazebo::common::MouseEvent	Generic description of a mouse event	563
gazebo::rendering::MovableText	Movable text	566
gazebo::msgs::MsgFactory	A factory that generates protobuf message based on a string type	572

gazebo::sensors::MultiCameraSensor	
Multiple camera sensor	573
gazebo::physics::MultiRayShape	
Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner	578
gazebo::transport::Node	
A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics	587
gazebo::common::NodeAnimation	
Node animation	593
gazebo::common::NodeAssignment	
Vertex to node weighted assignment for skeleton animation visualization	597
gazebo::common::NodeTransform	
NodeTransform (p. 598) Skeleton.hh (p. 1268) common/common.hh	598
gazebo::sensors::Noise	
Noise (p. 603) models for sensor output signals	603
gazebo::common::NumericAnimation	
A numeric animation	606
gazebo::common::NumericKeyFrame	
A keyframe for a NumericAnimation (p. 606)	607
gazebo::util::OpenAL	
3D audio setup and playback	609
gazebo::util::OpenALSink	
OpenAL (p. 609) Listener	611
gazebo::util::OpenALSource	
OpenAL (p. 609) Source	612
gazebo::rendering::OrbitViewController	
Orbit view controller	617
gazebo::common::ParamT< T >	620
gazebo::physics::PhysicsEngine	
Base (p. 153) class for a physics engine	620
gazebo::physics::PhysicsFactory	
The physics factory instantiates different physics engines	634
gazebo::common::PID	
Generic PID (p. 636) controller class	636
gazebo::math::Plane	
A plane and related functions	640
gazebo::physics::PlaneShape	
Collision (p. 213) for an infinite plane	642
gazebo::PluginT< T >	
A class which all plugins must inherit from	646
gazebo::math::Pose	
Encapsulates a position and rotation in three space	648
gazebo::common::PoseAnimation	
A pose animation	657
gazebo::common::PoseKeyFrame	
A keyframe for a PoseAnimation (p. 657)	660
gazebo::rendering::Projector	
Projects a material onto surface, light a light projector	662
gazebo::transport::Publication	
A publication for a topic	664
gazebo::transport::PublicationTransport	
Transport/transport.hh	669
gazebo::transport::Publisher	
A publisher of messages on a topic	671

QuadNode	674
gazebo::math::Quaternion	
A quaternion class	675
gazebo::math::Rand	
Random number generator class	688
gazebo::transport::RawCallbackHelper	
Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher	690
gazebo::sensors::RaySensor	
Sensor (p. 751) with one or more rays	693
gazebo::physics::RayShape	
Base (p. 153) class for Ray collision geometry	700
gazebo::rendering::RenderEngine	
Adaptor to Ogre3d	706
gazebo::sensors::RFIDSensor	
Sensor (p. 751) class for RFID type of sensor	710
gazebo::sensors::RFIDTag	
RFIDTag (p. 712) to interact with RFIDTagSensors	712
gazebo::rendering::RFIDTagVisual	
Visualization for RFID tags sensor	715
gazebo::rendering::RFIDVisual	
Visualization for RFID sensor	716
Road	
Used to render a strip of road	718
gazebo::physics::Road	
For building a Road (p. 718) from SDF	718
gazebo::rendering::Road2d	720
gazebo::math::RotationSpline	
Spline (p. 906) for rotations	720
gazebo::rendering::RTShaderSystem	
Implements Ogre (p. 123)'s Run-Time Shader system	724
gazebo::rendering::Scene	
Representation of an entire scene graph	728
gazebo::physics::ScrewJoint< T >	
A screw joint, which has both prismatic and rotational DOFs	746
gazebo::rendering::SelectionObj	
Interactive selection object for models and links	749
gazebo::sensors::Sensor	
Base class for sensors	751
SensorFactor	
The sensor factory; the class is just for namespacing purposes	762
gazebo::sensors::SensorFactory	762
gazebo::sensors::SensorManager	
Class to manage and update all sensors	764
gazebo::SensorPlugin	
A plugin with access to physics::Sensor	767
gazebo::Server	769
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg	
Keeping the CG shader for reference	771
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL	
Utility class to help with generating shaders for GLSL	773
gazebo::physics::Shape	
Base (p. 153) class for all shapes	775

gazebo::physics::SimbodyBallJoint	
SimbodyBallJoint (p. 778) class models a ball joint in Simbody	778
gazebo::physics::SimbodyBoxShape	
Simbody box collision	784
gazebo::physics::SimbodyCollision	
Simbody collisions	785
gazebo::physics::SimbodyCylinderShape	
Cylinder collision	788
gazebo::physics::SimbodyHeightmapShape	
Height map collision	790
gazebo::physics::SimbodyHinge2Joint	
A two axis hinge joint	792
gazebo::physics::SimbodyHingeJoint	
A single axis hinge joint	799
gazebo::physics::SimbodyJoint	
Base (p. 153) class for all joints	805
gazebo::physics::SimbodyLink	
Simbody Link (p. 455) class	813
gazebo::physics::SimbodyMeshShape	
Triangle mesh collision	823
gazebo::physics::SimbodyModel	
A model is a collection of links, joints, and plugins	825
gazebo::physics::SimbodyMultiRayShape	
Simbody specific version of MultiRayShape (p. 578)	826
gazebo::physics::SimbodyPhysics	
Simbody physics engine	828
gazebo::physics::SimbodyPlaneShape	
Simbody collision for an infinite plane	837
gazebo::physics::SimbodyRayShape	
Ray shape for simbody	839
gazebo::physics::SimbodyScrewJoint	
A screw joint	842
gazebo::physics::SimbodySliderJoint	
A slider joint	849
gazebo::physics::SimbodySphereShape	
Simbody sphere collision	855
gazebo::physics::SimbodyUniversalJoint	
A simbody universal joint class	857
SingletonT< T >	
Singleton template class	864
gazebo::common::Skeleton	
A skeleton	866
gazebo::common::SkeletonAnimation	
Skeleton (p. 866) animation	873
gazebo::common::SkeletonNode	
A skeleton node	877
gazebo::physics::SliderJoint< T >	
A slider joint	886
gazebo::rendering::GzTerrainMatGen::SM2Profile	
Shader model 2 profile target	888
gazebo::sensors::SonarSensor	
Sensor (p. 751) with sonar cone	890
gazebo::rendering::SonarVisual	
Visualization for sonar data	895

Joint_TEST::SpawnJointOptions	
Class to hold parameters for spawning joints	896
gazebo::physics::SphereShape	
Sphere collision shape	898
gazebo::common::SphericalCoordinates	
Convert spherical coordinates for planetary surfaces	901
gazebo::math::Spline	
Splines	906
gazebo::physics::State	
State (p.910) of an entity	910
gazebo::common::STLLoader	
Class used to load STL mesh files	915
gazebo::common::SubMesh	
A child mesh	916
gazebo::transport::SubscribeOptions	
Options for a subscription	927
gazebo::transport::Subscriber	
A subscriber to a topic	929
gazebo::transport::SubscriptionTransport	
Transport/transport.hh	930
gazebo::physics::SurfaceParams	
SurfaceParams (p.933) defines various Surface contact parameters	933
gazebo::common::SystemPaths	
Functions to handle getting system paths, keeps track of:	937
gazebo::SystemPlugin	
A plugin loaded within the gzserver on startup	942
gazebo::common::Time	
A Time (p.944) class, can be used to hold wall- or sim-time	944
gazebo::common::Timer	
A timer class, used to time things in real world walltime	965
gazebo::transport::TopicManager	
Manages topics and their subscriptions	967
gazebo::physics::TrajectoryInfo	
.	974
gazebo::rendering::TransmitterVisual	
Visualization for the wireless propagation data	975
gazebo::physics::UniversalJoint< T >	
A universal joint	976
gazebo::common::UpdateInfo	
Information for use in an update event	978
gazebo::rendering::UserCamera	
A camera used for user visualization of a scene	979
gazebo::math::Vector2d	
Generic double x, y vector	987
gazebo::math::Vector2i	
Generic integer x, y vector	996
gazebo::math::Vector3	
Generic vector containing 3 elements	1004
gazebo::math::Vector4	
Double Generic x, y, z, w vector	1018
gazebo::common::Video	
Handle video encoding and decoding using libavcodec	1028
gazebo::rendering::VideoVisual	
A visual element that displays a video as a texture	1029

gazebo::rendering::ViewController	
Base class for view controllers	1031
gazebo::rendering::Visual	
A renderable object	1034
gazebo::VisualPlugin	
A plugin loaded within the gzserver on startup	1055
gazebo::rendering::WindowManager	
Class to manage render windows	1057
gazebo::rendering::WireBox	
Draws a wireframe box	1059
gazebo::sensors::WirelessReceiver	
Sensor (p. 751) class for receiving wireless signals	1060
gazebo::sensors::WirelessTransceiver	
Sensor (p. 751) class for receiving wireless signals	1063
gazebo::sensors::WirelessTransmitter	
Transmitter to send wireless signals	1066
gazebo::physics::World	
The world provides access to all other object within a simulated environment	1070
gazebo::WorldPlugin	
A plugin with access to physics::World (p. 1070)	1082
gazebo::physics::WorldState	
Store state information of a physics::World (p. 1070) object	1083
gazebo::rendering::WrenchVisual	
Visualization for sonar data	1089

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Actor.hh	1093
Angle.hh	1094
Animation.hh	1097
ArrowVisual.hh	1098
Assert.hh	1099
AudioDecoder.hh	1101
AxisVisual.hh	1102
BallJoint.hh	1102
Base.hh	1103
Base64.hh	1104
Box.hh	1106
BoxShape.hh	1107
BVHLoader.hh	1108
CallbackHelper.hh	1110
Camera.hh	1112
CameraSensor.hh	1113
CameraVisual.hh	1113
cegui.h	1114
ColladaLoader.hh	1115
Collision.hh	1116
CollisionState.hh	1117
Color.hh	1117
Commonface.hh	1118
CommonTypes.hh	1120
COMVisual.hh	1122
Connection.hh	1123
ConnectionManager.hh	1125
Console.hh	1127
Contact.hh	1128
ContactManager.hh	1130
ContactSensor.hh	1131
ContactVisual.hh	1131
Conversions.hh	1132
CylinderShape.hh	1133

DepthCamera.hh	1134
DepthCameraSensor.hh	1135
Diagnostics.hh	1136
DynamicLines.hh	1137
DynamicRenderable.hh	1137
Entity.hh	1138
Event.hh	1139
Events.hh	1140
Exception.hh	1141
ForceTorqueSensor.hh	1143
FPSViewController.hh	1143
gazebo.hh	1144
gazebo_core.hh	1145
GazeboGenerator.hh	1146
GpsSensor.hh	1147
GpuLaser.hh	1148
GpuRaySensor.hh	1148
Grid.hh	1149
Gripper.hh	1150
GUIOverlay.hh	1151
Heightmap.hh	1151
HeightmapShape.hh	1152
Helpers.hh	1153
Hinge2Joint.hh	1156
HingeJoint.hh	1157
Image.hh	1159
ImuSensor.hh	1160
Inertial.hh	1160
IOManager.hh	1161
Joint.hh	1163
Joint_TEST.hh	1164
JointController.hh	1165
JointState.hh	1166
JointVisual.hh	1167
JointWrench.hh	1167
KeyEvent.hh	1169
KeyFrame.hh	1169
LaserVisual.hh	1171
Light.hh	1171
Link.hh	1172
LinkState.hh	1173
LogPlay.hh	1175
LogRecord.hh	1175
mainpage.html	1176
MapShape.hh	1176
Master.hh	1178
common/Material.hh	1179
rendering/Material.hh	1180
MathTypes.hh	
Forward declarations for the math classes	1180
Matrix3.hh	1181
Matrix4.hh	1181
Mesh.hh	1182
MeshCSG.hh	1184

MeshLoader.hh	1186
MeshManager.hh	1187
MeshShape.hh	1188
Model.hh	1190
ModelDatabase.hh	1191
ModelState.hh	1192
MouseEvent.hh	1193
MovableText.hh	1195
MsgFactory.hh	1195
msgs.hh	1196
MultiCameraSensor.hh	1199
MultiRayShape.hh	1199
Node.hh	1200
Noise.hh	1202
ogre_gazebo.h	1202
OpenAL.hh	1204
OrbitViewController.hh	1204
PhysicsEngine.hh	1205
PhysicsFactory.hh	1206
PhysicsIface.hh	1207
PhysicsTypes.hh	
Default namespace for gazebo	1209
PID.hh	1212
Plane.hh	1213
PlaneShape.hh	1214
Plugin.hh	1215
Pose.hh	1219
Projector.hh	1220
Publication.hh	1220
PublicationTransport.hh	1223
Publisher.hh	1225
Quaternion.hh	1227
Rand.hh	1228
RaySensor.hh	1229
RayShape.hh	1230
RenderEngine.hh	1231
RenderEvents.hh	1232
RenderingIface.hh	1233
RenderTypes.hh	1234
RFIDSensor.hh	1236
RFIDTag.hh	1237
RFIDTagVisual.hh	1237
RFIDVisual.hh	1238
Road.hh	1239
Road2d.hh	1240
RotationSpline.hh	1240
RTShaderSystem.hh	1242
Scene.hh	1242
ScrewJoint.hh	1244
SelectionObj.hh	1245
Sensor.hh	1245
SensorFactory.hh	1247
SensorManager.hh	1248
SensorsIface.hh	1248

SensorTypes.hh	
Forward declarations and typedefs for sensors	1250
Server.hh	1252
Shape.hh	1253
simbody_inc.h	1254
SimbodyBallJoint.hh	1254
SimbodyBoxShape.hh	1255
SimbodyCollision.hh	1255
SimbodyCylinderShape.hh	1256
SimbodyHeightmapShape.hh	1257
SimbodyHinge2Joint.hh	1257
SimbodyHingeJoint.hh	1258
SimbodyJoint.hh	1259
SimbodyLink.hh	1259
SimbodyMeshShape.hh	1260
SimbodyModel.hh	1261
SimbodyMultiRayShape.hh	1261
SimbodyPhysics.hh	1262
SimbodyPlaneShape.hh	1263
SimbodyRayShape.hh	1263
SimbodyScrewJoint.hh	1264
SimbodySliderJoint.hh	1265
SimbodySphereShape.hh	1265
SimbodyTypes.hh	
Simbody wrapper forward declarations and typedefs	1266
SimbodyUniversalJoint.hh	1267
SingletonT.hh	1268
Skeleton.hh	1268
SkeletonAnimation.hh	1270
SliderJoint.hh	1271
SonarSensor.hh	1272
SonarVisual.hh	1273
SphereShape.hh	1274
SphericalCoordinates.hh	1275
Spline.hh	1276
State.hh	1278
STLLoader.hh	1279
SubscribeOptions.hh	1280
Subscriber.hh	1282
SubscriptionTransport.hh	1284
SurfaceParams.hh	1286
SystemPaths.hh	1287
Time.hh	1288
Timer.hh	1289
TopicManager.hh	1290
TransmitterVisual.hh	1292
Transportface.hh	1293
TransportTypes.hh	
Forward declarations for transport	1295
UniversalJoint.hh	1296
UpdateInfo.hh	1297
UserCamera.hh	1298
UtilTypes.hh	1299
Vector2d.hh	1300

Vector2i.hh	1301
Vector3.hh	1302
Vector4.hh	1303
Video.hh	1305
VideoVisual.hh	1306
ViewController.hh	1307
Visual.hh	1308
WindowManager.hh	1309
WireBox.hh	1310
WirelessReceiver.hh	1311
WirelessTransceiver.hh	1311
WirelessTransmitter.hh	1312
World.hh	1313
WorldState.hh	1314
WrenchVisual.hh	1316

Chapter 8

Module Documentation

8.1 Common

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**
Message, error, warning functionality.
- class **gazebo::common::Exception**
Class for generating exceptions.
- class **gazebo::common::Image**
Encapsulates an image.
- class **gazebo::common::InternalError**

Class for generating Internal Gazebo Errors: those errors which should never happen 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::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. 537).*
- class **gazebo::common::MouseEvent**
Generic description of a mouse event.
- class **gazebo::common::NodeAnimation**
Node animation.
- struct **gazebo::common::NodeAssignment**
Vertex to node weighted assignment for skeleton animation visualization.
- class **gazebo::common::NodeTransform**
***NodeTransform** (p. 598) **Skeleton.hh** (p. 1268) *common/common.hh**
- class **gazebo::common::NumericAnimation**
A numeric animation.
- class **gazebo::common::NumericKeyFrame**
*A keyframe for a **NumericAnimation** (p. 606).*
- class **gazebo::common::PID**
*Generic **PID** (p. 636) 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. 657).*
- 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. 866) animation.*

- class **gazebo::common::SkeletonNode**
A skeleton node.
- class **gazebo::common::SphericalCoordinates**
Convert spherical coordinates for planetary surfaces.
- 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. 944) 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. 1070).*

Macros

- **#define gzclr_end** "\033[0m"
End marker.
- **#define gzclr_start**(clr) "\033[1;33m"
Start marker.
- **#define gzdbg** (gazebo::common::Console::Instance()->ColorMsg("Dbg", 36))
Output a debug message.
- **#define gzerr**
Output an error message.
- **#define gzlog** (gazebo::common::Console::Instance()->Log())
Output a message to a log file.
- **#define gzmsg** (gazebo::common::Console::Instance()->ColorMsg("Msg", 32))
Output a message.
- **#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**
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::Console::NullStream::NullStream ()**
constructor
- void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)
*add path prefix to **common::SystemPaths** (p. 937)*
- std::ostream & **gazebo::common::Console::ColorErr** (const std::string &_lbl, const std::string &_file, unsigned int _line, int _color)
Use this to output an error to the terminal.
- std::ostream & **gazebo::common::Console::ColorMsg** (const std::string &_lbl, int _color)
Use this to output a colored message to the terminal.
- void **gazebo::common::ModelDatabase::DownloadDependencies** (const std::string &_path)
Download all dependencies for a give model path.
- std::string **gazebo::common::find_file** (const std::string &_file)
*search for file in **common::SystemPaths** (p. 937)*
- std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath)
*search for file in **common::SystemPaths** (p. 937)*
- std::string **gazebo::common::find_file_path** (const std::string &_file)
*search for a file in **common::SystemPaths** (p. 937)*
- void **gazebo::common::ModelDatabase::Fini** ()
Finalize the model database.
- 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.
- void **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.
- bool **gazebo::common::Console::GetQuiet** () const
Get whether quiet output is set.
- std::string **gazebo::common::ModelDatabase::GetURI** ()
Returns the the global model database URI.
- bool **gazebo::common::ModelDatabase::HasModel** (const std::string &_modelName)
Returns true if the model exists on the database.
- void **gazebo::common::Console::Init** (const std::string &_logFilename)
Load the message parameters.
- bool **gazebo::common::Console::IsInitialized** () const
Return true if Init has been called.

- void **gazebo::common::load** ()
Load the common library.
- std::ofstream & **gazebo::common::Console::Log** ()
Use this to output a colored message to the terminal.
- void **gazebo::common::Console::SetQuiet** (bool _q)
Set quiet output.
- void **gazebo::common::ModelDatabase::Start** (bool _fetchImmediately=false)
Start the model database.

Variables

- static std::string **gazebo::common::PixelFormatNames** []
String names for the pixel formats.

8.1.1 Detailed Description

8.1.2 Macro Definition Documentation

8.1.2.1 #define gzclr_end "\033[0m"

End marker.

8.1.2.2 #define gzclr_start(clr) "\033[1;33m"

Start marker.

8.1.2.3 #define gzdbg (gazebo::common::Console::Instance()->ColorMsg("Dbg", 36))

Output a debug message.

Referenced by Joint_TEST::Setup().

8.1.2.4 #define gzerr

Value:

```
(gazebo::common::Console::Instance()->ColorErr("Error", \
    __FILE__, __LINE__, 31))
```

Output an error message.

Referenced by gazebo::transport::Connection::AsyncRead(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::physics::SimbodySphereShape::SetRadius(), gazebo::physics::SimbodyBoxShape::SetSize(), and gazebo::physics::SimbodyCylinderShape::SetSize().

8.1.2.5 #define gzlog (gazebo::common::Console::Instance()->Log())

Output a message to a log file.

8.1.2.6 #define gzmsg (gazebo::common::Console::Instance()->ColorMsg("Msg", 32))

Output a message.

8.1.2.7 #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::PluginT< ModelPlugin >::Create(), gazebo::transport::CallbackHelperT< M >::GetMsgType(), and gazebo::transport::SubscribeOptions::Init().

8.1.2.8 #define gzwarn

Value:

```
(gazebo::common::Console::Instance()->ColorErr("Warning", \
    __FILE__, __LINE__, 33))
```

Output a warning message.

Referenced by gazebo::physics::ScrewJoint< SimbodyJoint >::Load(), gazebo::physics::SimbodySphereShape::SetRadius(), gazebo::physics::SimbodyBoxShape::SetSize(), gazebo::physics::SimbodyCylinderShape::SetSize(), and Joint_TEST::SpawnJoint().

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 gazebo::common::Console::NullStream::NullStream() [inline]

constructor

8.1.4.2 `void gazebo::common::add_search_path_suffix (const std::string & _suffix)`

add path prefix to **common::SystemPaths** (p. 937)

8.1.4.3 `std::ostream& gazebo::common::Console::ColorErr (const std::string & _lbl, const std::string & _file, unsigned int _line, int _color)`

Use this to output an error to the terminal.

Parameters

<code>in</code>	<code><i>_lbl</i></code>	Text label
<code>in</code>	<code><i>_file</i></code>	File containing the error
<code>in</code>	<code><i>_line</i></code>	Line containing the error
<code>in</code>	<code><i>_color</i></code>	Color (p. 226) to make the label

Returns

Reference to an output stream

8.1.4.4 `std::ostream& gazebo::common::Console::ColorMsg (const std::string & _lbl, int _color)`

Use this to output a colored message to the terminal.

Parameters

<code>in</code>	<code><i>_lbl</i></code>	Text label
<code>in</code>	<code><i>_color</i></code>	Color (p. 226) to make the label

Returns

Reference to an output stream

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

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

search for file in **common::SystemPaths** (p. 937)

Parameters

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

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

search for file in **common::SystemPaths** (p. 937)

Parameters

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

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

search for a file in **common::SystemPaths** (p. 937)

Parameters

<code>in</code>	<code>_file</code>	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 `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.

8.1.4.11 `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.

8.1.4.12 `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

8.1.4.13 `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://gazebo.org/gazebo_models/ground_plane` or `model://gazebo_models`

Parameters

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

Returns

the model's name.

8.1.4.14 `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

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

Returns

path to a model directory

8.1.4.15 `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. 551) to download the model list.

Returns

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

8.1.4.16 `void 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. 39)

Parameters

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

8.1.4.17 `bool gazebo::common::Console::GetQuiet () const`

Get whether quiet output is set.

Returns

True to if quiet output is set

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

Returns the the global model database URI.

Returns

the URI.

8.1.4.19 `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.

8.1.4.20 `void gazebo::common::Console::Init (const std::string & _logFilename)`

Load the message parameters.

8.1.4.21 `bool gazebo::common::Console::IsInitialized () const`

Return true if Init has been called.

Returns

True is initialized.

8.1.4.22 void gazebo::common::load ()

Load the common library.

8.1.4.23 std::ofstream& gazebo::common::Console::Log ()

Use this to output a colored message to the terminal.

Parameters

in	<i>_lbl</i>	Text label
----	-------------	------------

Returns

Reference to an output stream

8.1.4.24 void gazebo::common::Console::SetQuiet (bool *_q*)

Set quiet output.

Parameters

in	<i>q</i>	True to prevent warning
----	----------	-------------------------

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

Start the model database.

Parameters

in	<i>_fetch-Immediately</i>	True to fetch the models without waiting.
----	---------------------------	---

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. 391).

8.2 Events

Namespaces

- namespace **gazebo::event**
Event (p. 305) namespace.

Classes

- class **gazebo::event::Connection**
A class that encapsulates a connection.
- class **gazebo::event::Event**
Base class for all events.
- class **gazebo::event::Events**
*An **Event** (p. 305) class to get notifications for simulator events.*
- class **gazebo::event::EventT< T >**
A class for event processing.

Functions

- 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 () [virtual]`

Destructor.

Destructor. Deletes all the associated connections.

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

Connect a callback to this event.

Adds a connection.

Parameters

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

Returns

A **Connection** (p. 239) object, which will automatically call `Disconnect` when it goes out of scope

Parameters

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

Referenced by `gazebo::event::Events::ConnectAddEntity()`, `gazebo::event::Events::ConnectCreateEntity()`, `gazebo::rendering::Events::ConnectCreateScene()`, `gazebo::event::Events::ConnectDeleteEntity()`, `gazebo::event::Events::ConnectDiagTimerStart()`, `gazebo::event::Events::ConnectDiagTimerStop()`, `gazebo::physics::Link::ConnectEnabled()`, `gazebo::physics::Joint::ConnectJointUpdate()`, `gazebo::rendering::DepthCamera::ConnectNewDepthFrame()`, `gazebo::rendering::Camera::ConnectNewImageFrame()`, `gazebo::rendering::GpuLaser::ConnectNewLaserFrame()`, `gazebo::physics::MultiRayShape::ConnectNewLaserScans()`, `gazebo::rendering::DepthCamera::ConnectNewRGBPointCloud()`, `gazebo::event::Events::ConnectPause()`, `gazebo::event::Events::ConnectPostRender()`, `gazebo::event::Events::ConnectPreRender()`, `gazebo::rendering::Events::ConnectRemoveScene()`, `gazebo::event::Events::ConnectRender()`, `gazebo::event::Events::ConnectSetSelectedEntity()`, `gazebo::event::Events::ConnectSigInt()`, `gazebo::event::Events::ConnectStep()`, `gazebo::event::Events::ConnectStop()`, `gazebo::transport::Connection::ConnectToShutdown()`, `gazebo::sensors::ForceTorqueSensor::ConnectUpdate()`, `gazebo::sensors::SonarSensor::ConnectUpdate()`, `gazebo::sensors::Sensor::ConnectUpdated()`, `gazebo::event::Events::ConnectWorldCreated()`, `gazebo::event::Events::ConnectWorldUpdateBegin()`, and `gazebo::event::Events::ConnectWorldUpdateEnd()`.

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

Get the number of connections.

Returns

Number of connection to this **Event** (p. 305).

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

Disconnect a callback to this event.

Removes a connection.

Parameters

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

Implements **gazebo::event::Event** (p. 307).

References NULL.

Referenced by `gazebo::event::Events::DisconnectAddEntity()`, `gazebo::event::Events::DisconnectCreateEntity()`, `gazebo::rendering::Events::DisconnectCreateScene()`, `gazebo::event::Events::DisconnectDeleteEntity()`, `gazebo::event::Events::DisconnectDiagTimerStart()`, `gazebo::event::Events::DisconnectDiagTimerStop()`, `gazebo::physics::Link::DisconnectEnabled()`, `gazebo::physics::Joint::DisconnectJointUpdate()`, `gazebo::rendering::DepthCamera::DisconnectNewDepthFrame()`, `gazebo::rendering::Camera::DisconnectNewImageFrame()`, `gazebo::rendering::-`

GpuLaser::DisconnectNewLaserFrame(), gazebo::physics::MultiRayShape::DisconnectNewLaserScans(), gazebo::rendering::DepthCamera::DisconnectNewRGBPointCloud(), gazebo::event::Events::DisconnectPause(), gazebo::event::Events::DisconnectPostRender(), gazebo::event::Events::DisconnectPreRender(), gazebo::rendering::Events::DisconnectRemoveScene(), gazebo::event::Events::DisconnectRender(), gazebo::event::Events::DisconnectSetSelectedEntity(), gazebo::transport::Connection::DisconnectShutdown(), gazebo::event::Events::DisconnectSigInt(), gazebo::event::Events::DisconnectStep(), gazebo::event::Events::DisconnectStop(), gazebo::sensors::ForceTorqueSensor::DisconnectUpdate(), gazebo::sensors::SonarSensor::DisconnectUpdate(), gazebo::sensors::Sensor::DisconnectUpdated(), gazebo::event::Events::DisconnectWorldCreated(), and gazebo::event::Events::DisconnectWorldUpdateEnd().

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

Disconnect a callback to this event.

Removes a connection.

Parameters

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

Implements `gazebo::event::Event` (p. 307).

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. 906) 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. 1004) 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
- **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
- `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::SimbodySphereShape::SetRadius()`, `gazebo::physics::SimbodyBoxShape::SetSize()`, and `gazebo::physics::SimbodyCylinderShape::SetSize()`.

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

check if a float is NaN

Parameters

in	<code>_v</code>	the value
----	-----------------	-----------

Returns

true if `_v` is not a number, false otherwise

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

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

check if a double is NaN

Parameters

in	<code>_v</code>	the value
----	-----------------	-----------

Returns

true if `_v` is not a number, false otherwise

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

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

is this a power of 2?

Parameters

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

Returns

true if `_x` is a power of 2, false otherwise

8.3.2.6 `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()`.

8.3.2.7 `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

8.3.2.8 `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().

8.3.2.9 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.

8.3.2.10 int gazebo::math::parseInt (const std::string & *_input*) [inline]

parse string into an integer

Parameters

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

Returns

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

References gazebo::math::NAN_I.

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

Returns

the value for the specified precision

8.3.2.12 `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. 102).*
- 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

- `msgs::Vector3d gazebo::msgs::Convert` (const math::Vector3 &_v)
*Convert a **math::Vector3** (p. 1004) to a `msgs::Vector3d`.*
- `msgs::Quaternion gazebo::msgs::Convert` (const math::Quaternion &_q)
*Convert a **math::Quaternion** (p. 675) to a `msgs::Quaternion`.*
- `msgs::Pose gazebo::msgs::Convert` (const math::Pose &_p)
*Convert a **math::Pose** (p. 648) to a `msgs::Pose`.*
- `msgs::Color gazebo::msgs::Convert` (const common::Color &_c)
*Convert a **common::Color** (p. 226) to a `msgs::Color`.*
- `msgs::Time gazebo::msgs::Convert` (const common::Time &_t)
*Convert a **common::Time** (p. 944) to a `msgs::Time`.*
- `msgs::PlaneGeom gazebo::msgs::Convert` (const math::Plane &_p)
*Convert a **math::Plane** (p. 640) to a `msgs::PlaneGeom`.*
- `math::Vector3 gazebo::msgs::Convert` (const msgs::Vector3d &_v)
*Convert a `msgs::Vector3d` to a **math::Vector**.*
- `math::Quaternion gazebo::msgs::Convert` (const msgs::Quaternion &_q)
*Convert a `msgs::Quaternion` to a **math::Quaternion** (p. 675).*
- `math::Pose gazebo::msgs::Convert` (const msgs::Pose &_p)
*Convert a `msgs::Pose` to a **math::Pose** (p. 648).*
- `common::Color gazebo::msgs::Convert` (const msgs::Color &_c)
*Convert a `msgs::Color` to a **common::Color** (p. 226).*
- `common::Time gazebo::msgs::Convert` (const msgs::Time &_t)
*Convert a `msgs::Time` to a **common::Time** (p. 944).*
- `math::Plane gazebo::msgs::Convert` (const msgs::PlaneGeom &_p)

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

- `msgs::Request * gazebo::msgs::CreateRequest` (const std::string &_request, const std::string &_data="")
Create a request message.
- `msgs::Fog gazebo::msgs::FogFromSDF` (sdf::ElementPtr _sdf)
Create a `msgs::Fog` from a fog SDF element.
- `msgs::Geometry gazebo::msgs::GeometryFromSDF` (sdf::ElementPtr _sdf)
Create a `msgs::Geometry` from a geometry SDF element.
- `msgs::Header * gazebo::msgs::GetHeader` (google::protobuf::Message &_message)
Get the header from a protobuf message.
- `msgs::GUI gazebo::msgs::GUIFromSDF` (sdf::ElementPtr _sdf)
Create a `msgs::GUI` from a GUI SDF element.
- `void gazebo::msgs::Init` (google::protobuf::Message &_message, const std::string &_id="")
Initialize a message.
- `msgs::Light gazebo::msgs::LightFromSDF` (sdf::ElementPtr _sdf)
Create a `msgs::Light` from a light SDF element.
- `msgs::MeshGeom gazebo::msgs::MeshFromSDF` (sdf::ElementPtr _sdf)
Create a `msgs::MeshGeom` from a mesh SDF element.
- `msgs::Scene gazebo::msgs::SceneFromSDF` (sdf::ElementPtr _sdf)
Create a `msgs::Scene` from a scene SDF element.
- `void gazebo::msgs::Set` (common::Image &_img, const msgs::Image &_msg)
Convert a `msgs::Image` to a `common::Image` (p. 389).
- `void gazebo::msgs::Set` (msgs::Image *_msg, const common::Image &_i)
Set a `msgs::Image` from a `common::Image` (p. 389).
- `void gazebo::msgs::Set` (msgs::Vector3d *_pt, const math::Vector3 &_v)
Set a `msgs::Vector3d` from a `math::Vector3` (p. 1004).
- `void gazebo::msgs::Set` (msgs::Vector2d *_pt, const math::Vector2d &_v)
Set a `msgs::Vector2d` from a `math::Vector3` (p. 1004).
- `void gazebo::msgs::Set` (msgs::Quaternion *_q, const math::Quaternion &_v)
Set a `msgs::Quaternion` from a `math::Quaternion` (p. 675).
- `void gazebo::msgs::Set` (msgs::Pose *_p, const math::Pose &_v)
Set a `msgs::Pose` from a `math::Pose` (p. 648).
- `void gazebo::msgs::Set` (msgs::Color *_c, const common::Color &_v)
Set a `msgs::Color` from a `common::Color` (p. 226).
- `void gazebo::msgs::Set` (msgs::Time *_t, const common::Time &_v)
Set a `msgs::Time` from a `common::Time` (p. 944).
- `void gazebo::msgs::Set` (msgs::PlaneGeom *_p, const math::Plane &_v)
Set a `msgs::Plane` from a `math::Plane` (p. 640).
- `void gazebo::msgs::Stamp` (msgs::Header *_header)
Time stamp a header.
- `void gazebo::msgs::Stamp` (msgs::Time *_time)
Set the time in a time message.
- `msgs::TrackVisual gazebo::msgs::TrackVisualFromSDF` (sdf::ElementPtr _sdf)
Create a `msgs::TrackVisual` from a track visual SDF element.
- `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:

```
boost::shared_ptr<google::protobuf::Message> New##_classname() \
{ \
    return boost::shared_ptr<gazebo::msgs::_classname>(\
        new gazebo::msgs::_classname); \
} \
class 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 msgs::Vector3d gazebo::msgs::Convert (const math::Vector3 & *_v*)

Convert a **math::Vector3** (p. 1004) to a msgs::Vector3d.

Parameters

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

Returns

A msgs::Vector3d object

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

Convert a **math::Quaternion** (p. 675) to a msgs::Quaternion.

Parameters

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

Returns

A `msgs::Quaternion` object

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

Convert a `math::Pose` (p. 648) to a `msgs::Pose`.

Parameters

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

Returns

A `msgs::Pose` object

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

Convert a `common::Color` (p. 226) to a `msgs::Color`.

Parameters

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

Returns

A `msgs::Color` object

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

Convert a `common::Time` (p. 944) to a `msgs::Time`.

Parameters

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

Returns

A `msgs::Time` object

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

Convert a `math::Plane` (p. 640) to a `msgs::PlaneGeom`.

Parameters

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

Returns

A `msgs::PlaneGeom` object

8.4.3.7 `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. 1004) object

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

Convert a `msgs::Quaternion` to a `math::Quaternion` (p. 675).

Parameters

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

Returns

A `math::Quaternion` (p. 675) object

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

Convert a `msgs::Pose` to a `math::Pose` (p. 648).

Parameters

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

Returns

A `math::Pose` (p. 648) object

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

Convert a `msgs::Color` to a `common::Color` (p. 226).

Parameters

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

Returns

A **common::Color** (p. 226) object

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

Convert a `msgs::Time` to a **common::Time** (p. 944).

Parameters

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

Returns

A **common::Time** (p. 944) object

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

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

Parameters

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

Returns

A `common::Plane` object

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

Create a request message.

Parameters

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

Returns

A Request message

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

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

Parameters

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

Returns

The new msgs::Fog object

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

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

Parameters

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

Returns

The new msgs::Geometry object

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

Get the header from a protobuf message.

Parameters

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

Returns

A pointer to the message's header

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

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

Parameters

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

Returns

The new msgs::GUI object

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

Initialize a message.

Parameters

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

Referenced by gazebo::physics::HingeJoint< SimbodyJoint >::Init().

8.4.3.19 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 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 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 void gazebo::msgs::Set (common::Image & *_img*, const msgs::Image & *_msg*)

Convert a msgs::Image to a **common::Image** (p. 389).

Parameters

out	<i>_img</i>	The common::Image (p. 389) container
in	<i>_msg</i>	The Image message to convert

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

Set a `msgs::Image` from a **common::Image** (p. 389).

Parameters

out	<code>_msg</code>	A <code>msgs::Image</code> pointer
in	<code>_i</code>	A common::Image (p. 389) reference

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

Set a `msgs::Vector3d` from a **math::Vector3** (p. 1004).

Parameters

out	<code>_pt</code>	A <code>msgs::Vector3d</code> pointer
in	<code>_v</code>	A math::Vector3 (p. 1004) reference

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

Set a `msgs::Vector2d` from a **math::Vector3** (p. 1004).

Parameters

out	<code>_pt</code>	A <code>msgs::Vector2d</code> pointer
in	<code>_v</code>	A math::Vector2d (p. 987) reference

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

Set a `msgs::Quaternion` from a **math::Quaternion** (p. 675).

Parameters

out	<code>_q</code>	A <code>msgs::Quaternion</code> pointer
in	<code>_v</code>	A math::Quaternion (p. 675) reference

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

Set a `msgs::Pose` from a **math::Pose** (p. 648).

Parameters

out	<code>_p</code>	A <code>msgs::Pose</code> pointer
in	<code>_v</code>	A math::Pose (p. 648) reference

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

Set a `msgs::Color` from a **common::Color** (p. 226).

Parameters

out	<code>_p</code>	A <code>msgs::Color</code> pointer
in	<code>_v</code>	A common::Color (p. 226) reference

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

Set a `msgs::Time` from a **common::Time** (p. 944).

Parameters

out	<code>_p</code>	A <code>msgs::Time</code> pointer
in	<code>_v</code>	A common::Time (p. 944) reference

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

Set a `msgs::Plane` from a **math::Plane** (p. 640).

Parameters

out	<code>_p</code>	A <code>msgs::Plane</code> pointer
in	<code>_v</code>	A math::Plane (p. 640) reference

8.4.3.31 `void gazebo::msgs::Stamp (msgs::Header * _header)`

Time stamp a header.

Parameters

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

8.4.3.32 `void gazebo::msgs::Stamp (msgs::Time * _time)`

Set the time in a time message.

Parameters

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

8.4.3.33 `msgs::TrackVisual gazebo::msgs::TrackVisualFromSDF (sdf::ElementPtr _sdf)`

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

Parameters

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

Returns

The new `msgs::TrackVisual` object

8.4.3.34 `msgs::Visual gazebo::msgs::VisualFromSDF (sdf::ElementPtr _sdf)`

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

Parameters

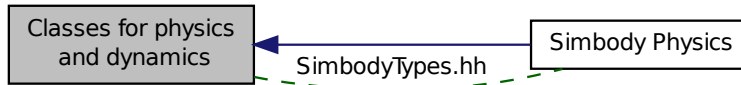
<code>in</code>	<code>_sdf</code>	The sdf element
-----------------	-------------------	-----------------

Returns

The new `msgs::Visual` object

8.5 Classes for physics and dynamics

Collaboration diagram for Classes for physics and dynamics:



Modules

- **Simbody Physics**
simbody physics engine wrapper

Files

- 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. 125) class enables GPU based mesh model / skeleton scriptable animation.
- class **gazebo::physics::BallJoint< T >**
Base (p. 153) class for a ball joint.
- class **gazebo::physics::Base**
Base (p. 153) class for most physics classes.
- class **gazebo::physics::BoxShape**
Box geometry primitive.
- class **gazebo::physics::Collision**
Base (p. 153) class for all collision entities.
- class **gazebo::physics::CollisionState**
*Store state information of a **physics::Collision** (p. 213) 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. 153) class for all physics objects in Gazebo.*
- class **gazebo::physics::Gripper**
A gripper abstraction.
- class **gazebo::physics::HeightmapShape**
***HeightmapShape** (p. 380) 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. 153) class for all joints.*
- class **gazebo::physics::JointController**
*A class for manipulating **physics::Joint** (p. 411).*
- class **gazebo::physics::JointState**
*keeps track of state of a **physics::Joint** (p. 411)*
- class **gazebo::physics::JointWrench**
Wrench information from a joint.
- class **gazebo::physics::Link**
***Link** (p. 455) 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. 455) object.*
- class **Logplay**
Open and playback log files that were recorded using LogRecord.
- class **gazebo::util::LogPlay**
- class **gazebo::physics::MapShape**
Creates box extrusions based on an image.
- 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. 537) 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. 153) class for a physics engine.*
- class **gazebo::physics::PhysicsFactory**
The physics factory instantiates different physics engines.
- class **gazebo::physics::PlaneShape**

Collision (p. 213) for an infinite plane.

- class **QuadNode**
- class **gazebo::physics::RayShape**
 - Base* (p. 153) class for Ray collision geometry.
- class **gazebo::physics::Road**
 - for building a **Road** (p. 718) from SDF
- class **gazebo::physics::ScrewJoint**< T >
 - A screw joint, which has both prismatic and rotational DOFs.
- class **gazebo::physics::Shape**
 - Base* (p. 153) 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. 910) of an entity.
- class **gazebo::physics::SurfaceParams**
 - SurfaceParams* (p. 933) 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. 1070) object.

Macros

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

Typedefs

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

Functions

- WorldPtr **gazebo::physics::create_world** (const std::string &_name="")
 - Create a world given a name.
- bool **gazebo::physics::fini** ()
 - Finalize transport by calling **gazebo::transport::fini** (p. 87).
- WorldPtr **gazebo::physics::get_world** (const std::string &_name="")
 - Returns a pointer to a world by name.
- uint32_t **gazebo::physics::getUniqueld** ()
 - Get a unique ID.

- void **gazebo::physics::init_world** (WorldPtr _world)
Init world given a pointer to it.
- void **gazebo::physics::init_worlds** ()
initialize multiple worlds stored in static variable gazebo::g_worlds
- bool **gazebo::physics::load** ()
Setup gazebo::SystemPlugin (p. 942)'s and call gazebo::transport::init (p. 88).
- void **gazebo::physics::load_world** (WorldPtr _world, sdf::ElementPtr _sdf)
Load world from sdf::Element pointer.
- void **gazebo::physics::load_worlds** (sdf::ElementPtr _sdf)
load multiple worlds from single sdf::Element pointer
- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)
Pause world by calling World::SetPaused (p. 1080).
- void **gazebo::physics::pause_worlds** (bool pause)
pause multiple worlds stored in static variable gazebo::g_worlds
- void **gazebo::physics::remove_worlds** ()
remove multiple worlds stored in static variable gazebo::g_worlds
- void **gazebo::physics::run_world** (WorldPtr _world, unsigned int _iterations=0)
Run world by calling World::Run() (p. 1080) given a pointer to it.
- void **gazebo::physics::run_worlds** (unsigned int _iterations=0)
Run multiple worlds stored in static variable gazebo::g_worlds.
- void **gazebo::physics::stop_world** (WorldPtr _world)
Stop world by calling World::Stop() (p. 1081) given a pointer to it.
- void **gazebo::physics::stop_worlds** ()
stop multiple worlds stored in static variable gazebo::g_worlds
- 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:

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

Static physics registration macro.

Use this macro to register physics engine with the server.

Parameters

<code>in</code>	<code>name</code>	Physics type name, as it appears in the world file.
<code>in</code>	<code>classname</code>	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 `WorldPtr gazebo::physics::create_world (const std::string & _name = " ")`

Create a world given a name.

Parameters

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

Returns

Pointer to the new world.

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

Finalize transport by calling `gazebo::transport::fini` (p. 87).

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

Returns a pointer to a world by name.

Parameters

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

Returns

Pointer to the world.

Referenced by `Joint_TEST::SpawnJoint()`.

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

Get a unique ID.

Returns

A unique integer

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

Init world given a pointer to it.

Parameters

<i>in</i>	<i>_world</i>	World (p. 1070) to initialize.
-----------	---------------	---------------------------------------

8.5.4.6 void gazebo::physics::init_worlds ()

initialize multiple worlds stored in static variable gazebo::g_worlds

8.5.4.7 bool gazebo::physics::load ()

Setup **gazebo::SystemPlugin** (p. 942)'s and call **gazebo::transport::init** (p. 88).

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

Load world from sdf::Element pointer.

Parameters

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

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

load multiple worlds from single sdf::Element pointer

Parameters

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

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

Pause world by calling **World::SetPaused** (p. 1080).

Parameters

<i>in</i>	<i>_world</i>	World (p. 1070) to pause or unpause.
<i>in</i>	<i>_pause</i>	True to pause, False to unpause.

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

pause multiple worlds stored in static variable gazebo::g_worlds

Parameters

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

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

remove multiple worlds stored in static variable `gazebo::g_worlds`

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

Run world by calling **World::Run()** (p. 1080) given a pointer to it.

Parameters

in	<code>_world</code>	World (p. 1070) to run.
in	<code>_iterations</code>	Number of iterations for each world to take. Zero indicates that each world should continue forever.

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

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

Parameters

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

8.5.4.15 `void gazebo::physics::stop_world (WorldPtr _world)`

Stop world by calling **World::Stop()** (p. 1081) given a pointer to it.

Parameters

in	<code>_world</code>	World (p. 1070) to stop.
----	---------------------	---------------------------------

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

stop multiple worlds stored in static variable `gazebo::g_worlds`

8.5.4.17 `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:

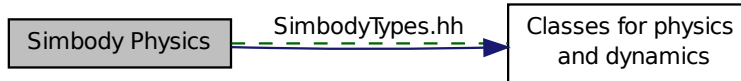
```
= {  
    "common",  
    "entity",  
    "model",  
    "actor",  
    "link",  
    "collision",  
    "light",  
    "visual",  
    "joint",  
    "ball",  
    "hinge2",  
    "hinge",  
    "slider",  
    "universal",  
    "shape",  
    "box",  
    "cylinder",  
    "heightmap",  
    "map",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

String names for the different entity types.

8.6 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. 778) 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. 153) class for all joints.*
- class **gazebo::physics::SimbodyLink**
*Simbody **Link** (p. 455) class.*
- class **gazebo::physics::SimbodyMeshShape**
Triangle mesh collision.
- class **gazebo::physics::SimbodyMultiRayShape**
*Simbody specific version of **MultiRayShape** (p. 578).*
- 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.

8.6.1 Detailed Description

simbody physics engine wrapper

8.7 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. 266) Conversions.hh (p. 1132) rendering/Conversions.hh (p. 1132).
- 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. 123)'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.

Enumerations

- enum **gazebo::rendering::SelectionObj::SelectionMode** {
gazebo::rendering::SelectionObj::SELECTION_NONE = 0, **gazebo::rendering::SelectionObj::TRANS**,
gazebo::rendering::SelectionObj::ROT, **gazebo::rendering::SelectionObj::SCALE**,
gazebo::rendering::SelectionObj::TRANS_X, **gazebo::rendering::SelectionObj::TRANS_Y**, **gazebo-
::rendering::SelectionObj::TRANS_Z**, **gazebo::rendering::SelectionObj::ROT_X**,
gazebo::rendering::SelectionObj::ROT_Y, **gazebo::rendering::SelectionObj::ROT_Z**, **gazebo::rendering-
::SelectionObj::SCALE_X**, **gazebo::rendering::SelectionObj::SCALE_Y**,
gazebo::rendering::SelectionObj::SCALE_Z }

Functions

- **gazebo::rendering::SelectionObj::SelectionObj** (const std::string &_name, VisualPtr _vis)
Constructor.
- virtual **gazebo::rendering::SelectionObj::~~SelectionObj** ()
Destructor.
- void **gazebo::rendering::SelectionObj::Attach** (rendering::VisualPtr _vis)
Attach the selection object to the given visual.
- rendering::ScenePtr **gazebo::rendering::create_scene** (const std::string &_name, bool _enableVisualizations, bool _isServer=false)
*create **rendering::Scene** (p. 728) by name.*
- void **gazebo::rendering::SelectionObj::Detach** ()
Detach the selection object from the current visual.
- bool **gazebo::rendering::fini** ()
teardown rendering engine.
- rendering::ScenePtr **gazebo::rendering::get_scene** (const std::string &_name="")
*get pointer to **rendering::Scene** (p. 728) by name.*
- SelectionMode **gazebo::rendering::SelectionObj::GetMode** ()
Get the current selection mode.
- SelectionMode **gazebo::rendering::SelectionObj::GetState** ()
Get the current selection state.
- bool **gazebo::rendering::init** ()
init rendering engine.
- bool **gazebo::rendering::load** ()
load rendering engine.
- void **gazebo::rendering::SelectionObj::Load** ()
Load.
- void **gazebo::rendering::remove_scene** (const std::string &_name)
*remove a **rendering::Scene** (p. 728) by name*
- void **gazebo::rendering::SelectionObj::SetGlobal** (bool _global)
Set selection object to ignore local transforms.
- void **gazebo::rendering::SelectionObj::SetMode** (const std::string &_mode)
Set the manipulation mode.
- void **gazebo::rendering::SelectionObj::SetMode** (SelectionMode _mode)
Set the selection mode.
- void **gazebo::rendering::SelectionObj::SetState** (const std::string &_state)
Set state by highlighting the corresponding selection object visual.

- void **gazebo::rendering::SelectionObj::SetState** (SelectionMode _state)
Set state by highlighting the corresponding selection object visual.
- void **gazebo::rendering::SelectionObj::UpdateSize** ()
Update selection object size to match the parent visual.

8.7.1 Detailed Description

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

8.7.2 Enumeration Type Documentation

8.7.2.1 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.

8.7.3 Function Documentation

8.7.3.1 gazebo::rendering::SelectionObj::SelectionObj (const std::string & _name, VisualPtr _vis)

Constructor.

Parameters

in	_name	Name of selection object.
in	_vis	Parent visual that the selection object is attached to.

8.7.3.2 virtual gazebo::rendering::SelectionObj::~SelectionObj () [virtual]

Destructor.

8.7.3.3 void gazebo::rendering::SelectionObj::Attach (rendering::VisualPtr _vis)

Attach the selection object to the given visual.

Parameters

in	<i>_vis</i>	Pointer to visual to which the selection object will be attached.
----	-------------	---

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

create **rendering::Scene** (p. 728) 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.

8.7.3.5 void gazebo::rendering::SelectionObj::Detach ()

Detach the selection object from the current visual.

8.7.3.6 bool gazebo::rendering::fini ()

teardown rendering engine.

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

get pointer to **rendering::Scene** (p. 728) by name.

Parameters

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

8.7.3.8 SelectionMode gazebo::rendering::SelectionObj::GetMode ()

Get the current selection mode.

8.7.3.9 SelectionMode gazebo::rendering::SelectionObj::GetState ()

Get the current selection state.

8.7.3.10 bool gazebo::rendering::init ()

init rendering engine.

8.7.3.11 `bool gazebo::rendering::load ()`

load rendering engine.

8.7.3.12 `void gazebo::rendering::SelectionObj::Load () [virtual]`

Load.

Reimplemented from `gazebo::rendering::Visual` (p. 1048).

8.7.3.13 `void gazebo::rendering::remove_scene (const std::string & _name)`

remove a `rendering::Scene` (p. 728) by name

Parameters

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

8.7.3.14 `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.
-----------------	----------------------	--

8.7.3.15 `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.
-----------------	--------------------	---

8.7.3.16 `void gazebo::rendering::SelectionObj::SetMode (SelectionMode _mode)`

Set the selection mode.

`_name` Selection mode: TRANS, ROT, SCALE.

8.7.3.17 `void gazebo::rendering::SelectionObj::SetState (const std::string & _state)`

Set state by highlighting the corresponding selection object visual.

Parameters

<code>in</code>	<code>_state</code>	Selection state in string format.
-----------------	---------------------	-----------------------------------

8.7.3.18 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. 76)

8.7.3.19 void gazebo::rendering::SelectionObj::UpdateSize ()

Update selection object size to match the parent visual.

8.8 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. 751) for measure force and torque on a joint.
- class **gazebo::sensors::GpsSensor**
GpsSensor (p. 341) 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. 603) models for sensor output signals.
- class **gazebo::sensors::RaySensor**
Sensor (p. 751) with one or more rays.
- class **gazebo::sensors::RFIDSensor**
Sensor (p. 751) class for RFID type of sensor.
- class **gazebo::sensors::RFIDTag**
RFIDTag (p. 712) 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. 751) with sonar cone.

- class **gazebo::sensors::WirelessReceiver**
Sensor (p. 751) class for receiving wireless signals.
- class **gazebo::sensors::WirelessTransceiver**
Sensor (p. 751) 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

- std::string **gazebo::sensors::create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName) **GAZEBO_DEPRECATED**(1.10)
Deprecated.
- 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.
- bool **gazebo::sensors::fini** ()
shutdown the sensor generation loop.
- SensorPtr **gazebo::sensors::get_sensor** (const std::string &_name)
Get a sensor using by name.
- bool **gazebo::sensors::init** ()
initialize the sensor generation loop.
- bool **gazebo::sensors::load** ()
Load the sensor library.
- void **gazebo::sensors::remove_sensor** (const std::string &_sensorName)
Remove a sensor by name.
- bool **gazebo::sensors::remove_sensors** ()
Remove all sensors.
- void **gazebo::sensors::run_once** (bool _force=false)
Run the sensor generation one step.
- void **gazebo::sensors::run_threads** ()
Run sensors in a threads. This is a non-blocking call.
- void **gazebo::sensors::stop** ()
Stop the sensor generation loop.

8.8.1 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).

8.8.2 Macro Definition Documentation

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

Value:

```
Sensor *New##classname() \
{ \
    return new gazebo::sensors::classname(); \
} \
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.

8.8.3 Function Documentation

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

Deprecated.

8.8.3.2 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.

8.8.3.3 bool gazebo::sensors::fini ()

shutdown the sensor generation loop.

Returns

True if successfully finalized, false if not

8.8.3.4 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.

8.8.3.5 bool gazebo::sensors::init ()

initialize the sensor generation loop.

Returns

True if successfully initialized, false if not

8.8.3.6 bool gazebo::sensors::load ()

Load the sensor library.

Returns

True if successfully loaded, false if not.

8.8.3.7 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
-----------	--------------------	--------------------------

8.8.3.8 bool gazebo::sensors::remove_sensors ()

Remove all sensors.

Returns

True if all successfully removed, false if not

8.8.3.9 `void gazebo::sensors::run_once (bool _force = false)`

Run the sensor generation one step.

Parameters

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

8.8.3.10 `void gazebo::sensors::run_threads ()`

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

8.8.3.11 `void gazebo::sensors::stop ()`

Stop the sensor generation loop.

8.9 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. 173)*

Functions

- void **gazebo::transport::clear_buffers** ()
Clear any remaining communication buffers.
- void **gazebo::transport::fini** ()
Cleanup the transport component.
- 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.
- void **gazebo::transport::get_topic_namespaces** (std::list< std::string > &_namespaces)
Return all the namespace (world names) on the master.
- std::map< std::string, std::list< std::string > > **gazebo::transport::getAdvertisedTopics** ()
Get a list of all the topics and their message types.
- std::list< std::string > **gazebo::transport::getAdvertisedTopics** (const std::string &_msgType)
Get a list of all the unique advertised topic names.
- bool **gazebo::transport::getMinimalComms** ()
Get whether minimal comms has been enabled.
- std::string **gazebo::transport::getTopicMsgType** (const std::string &_topicName)
Get the message typename that is published on the given topic.
- bool **gazebo::transport::init** (const std::string &_master_host="", unsigned int _master_port=0)
Initialize the transport system.
- bool **gazebo::transport::is_stopped** ()
Is the transport system stopped?
- void **gazebo::transport::pause_incoming** (bool _pause)
Pause or unpaue incoming messages.
- template<typename M >
void **gazebo::transport::publish** (const std::string &_topic, const google::protobuf::Message &_message)
A convenience function for a one-time publication of a message.
- 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.
- 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.
- void **gazebo::transport::requestNoReply** (NodePtr _node, const std::string &_request, const std::string &_data="")
Send a request and don't wait for a response.
- void **gazebo::transport::run** ()
Run the transport component.
- void **gazebo::transport::setMinimalComms** (bool _enabled)
Set whether minimal comms should be used.
- void **gazebo::transport::stop** ()
Stop the transport component from running.

8.9.1 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.

8.9.2 Typedef Documentation

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

boost shared pointer to **transport::CallbackHelper** (p. 173)

8.9.3 Function Documentation

8.9.3.1 `void gazebo::transport::clear_buffers ()`

Clear any remaining communication buffers.

8.9.3.2 `void gazebo::transport::fini ()`

Cleanup the transport component.

8.9.3.3 `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	<code>_master_host</code>	The hostname of the master is set to this param
out	<code>_master_port</code>	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)

8.9.3.4 `void gazebo::transport::get_topic_namespaces (std::list< std::string > & _namespaces)`

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

Parameters

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

8.9.3.5 `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.

8.9.3.6 `std::list<std::string> gazebo::transport::getAdvertisedTopics (const std::string & _msgType)`

Get a list of all the unique advertised topic names.

Parameters

in	<code>_msgType</code>	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`.

8.9.3.7 `bool gazebo::transport::getMinimalComms ()`

Get whether minimal comms has been enabled.

Returns

True if minimal comms is enabled.

8.9.3.8 `std::string gazebo::transport::getTopicMsgType (const std::string & _topicName)`

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

Parameters

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

Returns

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

8.9.3.9 `bool gazebo::transport::init (const std::string & _master_host = " ", unsigned int _master_port = 0)`

Initialize the transport system.

Parameters

in	<code>_master_host</code>	The hostname or IP of the master. Leave empty to use pull address from the GAZEBO_MASTER_URI env var.
in	<code>_master_port</code>	The port of the master. Leave empty to use pull address from the GAZEBO_MASTER_URI env var.

Returns

true if initialization succeeded; false otherwise

8.9.3.10 `bool gazebo::transport::is_stopped ()`

Is the transport system stopped?

Returns

true if the transport system is stopped; false otherwise

8.9.3.11 `void gazebo::transport::pause_incoming (bool _pause)`

Pause or unpause incoming messages.

When paused, messages are queued for later delivery

Parameters

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

8.9.3.12 `template<typename M > 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. 589) followed by **Publisher::Publish** (p. 673). This function should only be used when sending a message very infrequently.

Parameters

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

8.9.3.13 `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

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

Returns

The response to the request. Can be empty.

8.9.3.14 `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.

8.9.3.15 `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

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

8.9.3.16 `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

8.9.3.17 `void gazebo::transport::setMinimalComms (bool _enabled)`

Set whether minimal comms should be used.

This will be used to reduce network traffic.

8.9.3.18 `void gazebo::transport::stop ()`

Stop the transport component from running.

8.10 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. 609) Listener.
- class **gazebo::util::OpenALSource**
OpenAL (p. 609) Source.

Macros

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

8.10.1 Detailed Description

8.10.2 Macro Definition Documentation

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

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

8.10.2.3 #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. 305) 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 ROS Master-like manager that directs gztopic 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. 537).*

- 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. 1070).

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

- void **add_plugin** (const std::string &_filename)
- std::string **find_file** (const std::string &_file)
Find a file in the gazebo search paths.
- void **fini** ()
- bool **init** ()
- bool **load** (int _argc=0, char **_argv=0)
- void **print_version** ()
- void **run** ()
- void **stop** ()

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 void gazebo::add_plugin (const std::string & *filename*)

9.2.3.2 std::string gazebo::find_file (const std::string & *file*)

Find a file in the gazebo search paths.

9.2.3.3 void gazebo::fini ()

9.2.3.4 bool gazebo::init ()

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

9.2.3.6 void gazebo::print_version ()

9.2.3.7 void gazebo::run ()

9.2.3.8 void gazebo::stop ()

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**

Message, error, warning functionality.
- class **Exception**

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

Encapsulates an image.
- 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 **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 **MouseEvent**

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

Node animation.
- struct **NodeAssignment**

Vertex to node weighted assignement for skeleton animation visualization.
- class **NodeTransform**

NodeTransform (p. 598) **Skeleton.hh** (p. 1268) *common/common.hh*
- class **NumericAnimation**

A numeric animation.
- class **NumericKeyFrame**

*A keyframe for a **NumericAnimation** (p. 606).*
- class **ParamT**
- class **PID**

*Generic **PID** (p. 636) controller class.*
- class **PoseAnimation**

A pose animation.
- class **PoseKeyFrame**

*A keyframe for a **PoseAnimation** (p. 657).*
- class **Skeleton**

- A skeleton.*
- class **SkeletonAnimation**
 - Skeleton** (p. 866) animation.*
- class **SkeletonNode**
 - A skeleton node.*
- class **SphericalCoordinates**
 - Convert spherical coordinates for planetary surfaces.*
- 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. 944) 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

- void **add_search_path_suffix** (const std::string &_suffix)
*add path prefix to **common::SystemPaths** (p. 937)*
- std::string **find_file** (const std::string &_file)
*search for file in **common::SystemPaths** (p. 937)*
- std::string **find_file** (const std::string &_file, bool _searchLocalPath)
*search for file in **common::SystemPaths** (p. 937)*
- std::string **find_file_path** (const std::string &_file)
*search for a file in **common::SystemPaths** (p. 937)*
- 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. 305) namespace.

Classes

- class **Connection**
A class that encapsulates a connection.
- class **Event**
Base class for all events.
- class **Events**
*An **Event** (p. 305) class to get notifications for simulator events.*
- class **EventT**
A class for event processing.

Typedefs

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

9.4.1 Detailed Description

Event (p. 305) 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. 906) for rotations.*
- class **Spline**
Splines.
- class **Vector2d**
Generic double x, y vector.
- class **Vector2i**
Generic integer x, y vector.
- class **Vector3**
*The **Vector3** (p. 1004) 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
- 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
- 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

- msgs::Vector3d **Convert** (const **math::Vector3** &_v)
*Convert a **math::Vector3** (p. 1004) to a msgs::Vector3d.*
- msgs::Quaternion **Convert** (const **math::Quaternion** &_q)
*Convert a **math::Quaternion** (p. 675) to a msgs::Quaternion.*
- msgs::Pose **Convert** (const **math::Pose** &_p)
*Convert a **math::Pose** (p. 648) to a msgs::Pose.*
- msgs::Color **Convert** (const **common::Color** &_c)
*Convert a **common::Color** (p. 226) to a msgs::Color.*
- msgs::Time **Convert** (const **common::Time** &_t)
*Convert a **common::Time** (p. 944) to a msgs::Time.*
- msgs::PlaneGeom **Convert** (const **math::Plane** &_p)

- Convert a **math::Plane** (p. 640) to a `msgs::PlaneGeom`.*

 - **math::Vector3 Convert** (const msgs::Vector3d &_v)
 - Convert a `msgs::Vector3d` to a `math::Vector`.*
 - **math::Quaternion Convert** (const msgs::Quaternion &_q)
 - Convert a `msgs::Quaternion` to a **math::Quaternion** (p. 675).*
 - **math::Pose Convert** (const msgs::Pose &_p)
 - Convert a `msgs::Pose` to a **math::Pose** (p. 648).*
 - **common::Color Convert** (const msgs::Color &_c)
 - Convert a `msgs::Color` to a **common::Color** (p. 226).*
 - **common::Time Convert** (const msgs::Time &_t)
 - Convert a `msgs::Time` to a **common::Time** (p. 944).*
 - **math::Plane Convert** (const msgs::PlaneGeom &_p)
 - Convert a `msgs::PlaneGeom` to a `common::Plane`.*
 - msgs::Request * **CreateRequest** (const std::string &_request, const std::string &_data="")
 - Create a request message.*
 - msgs::Fog **FogFromSDF** (sdf::ElementPtr _sdf)
 - Create a `msgs::Fog` from a fog SDF element.*
 - msgs::Geometry **GeometryFromSDF** (sdf::ElementPtr _sdf)
 - Create a `msgs::Geometry` from a geometry SDF element.*
 - msgs::Header * **GetHeader** (google::protobuf::Message &_message)
 - Get the header from a protobuf message.*
 - msgs::GUI **GUIFromSDF** (sdf::ElementPtr _sdf)
 - Create a `msgs::GUI` from a GUI SDF element.*
 - void **Init** (google::protobuf::Message &_message, const std::string &_id="")
 - Initialize a message.*
 - msgs::Light **LightFromSDF** (sdf::ElementPtr _sdf)
 - Create a `msgs::Light` from a light SDF element.*
 - msgs::MeshGeom **MeshFromSDF** (sdf::ElementPtr _sdf)
 - Create a `msgs::MeshGeom` from a mesh SDF element.*
 - msgs::Scene **SceneFromSDF** (sdf::ElementPtr _sdf)
 - Create a `msgs::Scene` from a scene SDF element.*
 - void **Set (common::Image &_img, const msgs::Image &_msg)**
 - Convert a `msgs::Image` to a **common::Image** (p. 389).*
 - void **Set** (msgs::Image *_msg, const **common::Image** &_i)
 - Set a `msgs::Image` from a **common::Image** (p. 389).*
 - void **Set** (msgs::Vector3d *_pt, const **math::Vector3** &_v)
 - Set a `msgs::Vector3d` from a **math::Vector3** (p. 1004).*
 - void **Set** (msgs::Vector2d *_pt, const **math::Vector2d** &_v)
 - Set a `msgs::Vector2d` from a **math::Vector3** (p. 1004).*
 - void **Set** (msgs::Quaternion *_q, const **math::Quaternion** &_v)
 - Set a `msgs::Quaternion` from a **math::Quaternion** (p. 675).*
 - void **Set** (msgs::Pose *_p, const **math::Pose** &_v)
 - Set a `msgs::Pose` from a **math::Pose** (p. 648).*
 - void **Set** (msgs::Color *_c, const **common::Color** &_v)
 - Set a `msgs::Color` from a **common::Color** (p. 226).*
 - void **Set** (msgs::Time *_t, const **common::Time** &_v)
 - Set a `msgs::Time` from a **common::Time** (p. 944).*

- void **Set** (msgs::PlaneGeom *_p, const **math::Plane** &_v)
*Set a msgs::Plane from a **math::Plane** (p. 640).*
- void **Stamp** (msgs::Header *_header)
Time stamp a header.
- void **Stamp** (msgs::Time *_time)
Set the time in a time message.
- msgs::TrackVisual **TrackVisualFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::TrackVisual from a track visual SDF element.
- 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. 125) class enables GPU based mesh model / skeleton scriptable animation.*
- class **BallJoint**
***Base** (p. 153) class for a ball joint.*
- class **Base**
***Base** (p. 153) class for most physics classes.*
- class **BoxShape**
Box geometry primitive.
- class **Collision**
***Base** (p. 153) class for all collision entities.*
- class **CollisionState**
*Store state information of a **physics::Collision** (p. 213) 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. 253) Manager.*
- class **CylinderShape**
Cylinder collision.
- class **Entity**

- Base** (p. 153) class for all physics objects in Gazebo.*
- class **Gripper**
 - A gripper abstraction.*
- class **HeightmapShape**
 - HeightmapShape** (p. 380) 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. 153) class for all joints.*
- class **JointController**
 - A class for manipulating **physics::Joint** (p. 411).*
- class **JointState**
 - keeps track of state of a **physics::Joint** (p. 411)*
- class **JointWrench**
 - Wrench information from a joint.*
- class **Link**
 - Link** (p. 455) 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. 455) object.*
- class **MapShape**
 - Creates box extrusions based on an image.*
- 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. 537) object.*
- class **MultiRayShape**
 - Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.*
- class **PhysicsEngine**
 - Base** (p. 153) class for a physics engine.*
- class **PhysicsFactory**
 - The physics factory instantiates different physics engines.*
- class **PlaneShape**
 - Collision** (p. 213) for an infinite plane.*
- class **RayShape**
 - Base** (p. 153) class for Ray collision geometry.*
- class **Road**
 - for building a **Road** (p. 718) from SDF*
- class **ScrewJoint**
 - A screw joint, which has both prismatic and rotational DOFs.*
- class **Shape**

- **Base** (p. 153) class for all shapes.
- class **SimbodyBallJoint**
 - **SimbodyBallJoint** (p. 778) 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. 153) class for all joints.
- class **SimbodyLink**
 - *Simbody Link* (p. 455) 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. 578).*
- 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. 910) of an entity.
- class **SurfaceParams**
 - **SurfaceParams** (p. 933) defines various Surface contact parameters.

- struct **TrajectoryInfo**
- 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. 1070) 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< **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

- **WorldPtr create_world** (const std::string &_name="")
 Create a world given a name.
- bool **fini** ()
 Finalize transport by calling `gazebo::transport::fini` (p. 87).
- **WorldPtr get_world** (const std::string &_name="")
 Returns a pointer to a world by name.
- uint32_t **getUniqueId** ()
 Get a unique ID.
- void **init_world** (WorldPtr _world)
 Init world given a pointer to it.
- void **init_worlds** ()
 initialize multiple worlds stored in static variable `gazebo::g_worlds`
- bool **load** ()
 Setup `gazebo::SystemPlugin` (p. 942)'s and call `gazebo::transport::init` (p. 88).
- void **load_world** (WorldPtr _world, sdf::ElementPtr _sdf)
 Load world from `sdf::Element` pointer.
- void **load_worlds** (sdf::ElementPtr _sdf)
 load multiple worlds from single `sdf::Element` pointer
- void **pause_world** (WorldPtr _world, bool _pause)
 Pause world by calling `World::SetPaused` (p. 1080).
- void **pause_worlds** (bool pause)
 pause multiple worlds stored in static variable `gazebo::g_worlds`
- void **remove_worlds** ()
 remove multiple worlds stored in static variable `gazebo::g_worlds`

- void **run_world** (**WorldPtr** _world, unsigned int _iterations=0)
*Run world by calling **World::Run()** (p. 1080) given a pointer to it.*
- void **run_worlds** (unsigned int _iterations=0)
Run multiple worlds stored in static variable gazebo::g_worlds.
- void **stop_world** (**WorldPtr** _world)
*Stop world by calling **World::Stop()** (p. 1081) given a pointer to it.*
- void **stop_worlds** ()
stop multiple worlds stored in static variable gazebo::g_worlds
- 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<Entity> gazebo::physics::EntityPtr

9.7.2.11 typedef boost::shared_ptr<Gripper> gazebo::physics::GripperPtr

9.7.2.12 typedef boost::shared_ptr<HeightmapShape> gazebo::physics::HeightmapShapePtr

9.7.2.13 typedef boost::shared_ptr<Inertial> gazebo::physics::InertialPtr

- 9.7.2.14 `typedef std::vector<JointPtr> gazebo::physics::Joint_V`
- 9.7.2.15 `typedef std::vector<JointControllerPtr> gazebo::physics::JointController_V`
- 9.7.2.16 `typedef boost::shared_ptr<JointController> gazebo::physics::JointControllerPtr`
- 9.7.2.17 `typedef boost::shared_ptr<Joint> gazebo::physics::JointPtr`
- 9.7.2.18 `typedef std::map<std::string, JointState> gazebo::physics::JointState_M`
- 9.7.2.19 `typedef std::vector<LinkPtr> gazebo::physics::Link_V`
- 9.7.2.20 `typedef boost::shared_ptr<Link> gazebo::physics::LinkPtr`
- 9.7.2.21 `typedef std::map<std::string, LinkState> gazebo::physics::LinkState_M`
- 9.7.2.22 `typedef boost::shared_ptr<MeshShape> gazebo::physics::MeshShapePtr`
- 9.7.2.23 `typedef std::vector<ModelPtr> gazebo::physics::Model_V`
- 9.7.2.24 `typedef boost::shared_ptr<Model> gazebo::physics::ModelPtr`
- 9.7.2.25 `typedef std::map<std::string, ModelState> gazebo::physics::ModelState_M`
- 9.7.2.26 `typedef boost::shared_ptr<MultiRayShape> gazebo::physics::MultiRayShapePtr`
- 9.7.2.27 `typedef boost::shared_ptr<PhysicsEngine> gazebo::physics::PhysicsEnginePtr`
- 9.7.2.28 `typedef boost::shared_ptr<RayShape> gazebo::physics::RayShapePtr`
- 9.7.2.29 `typedef boost::shared_ptr<Road> gazebo::physics::RoadPtr`
- 9.7.2.30 `typedef boost::shared_ptr<Shape> gazebo::physics::ShapePtr`
- 9.7.2.31 `typedef boost::shared_ptr<SimbodyCollision> gazebo::physics::SimbodyCollisionPtr`
- 9.7.2.32 `typedef boost::shared_ptr<SimbodyLink> gazebo::physics::SimbodyLinkPtr`
- 9.7.2.33 `typedef boost::shared_ptr<SimbodyModel> gazebo::physics::SimbodyModelPtr`
- 9.7.2.34 `typedef boost::shared_ptr<SimbodyPhysics> gazebo::physics::SimbodyPhysicsPtr`
- 9.7.2.35 `typedef boost::shared_ptr<SimbodyRayShape> gazebo::physics::SimbodyRayShapePtr`
- 9.7.2.36 `typedef boost::shared_ptr<SphereShape> gazebo::physics::SphereShapePtr`
- 9.7.2.37 `typedef boost::shared_ptr<SurfaceParams> gazebo::physics::SurfaceParamsPtr`
- 9.7.2.38 `typedef boost::shared_ptr<World> gazebo::physics::WorldPtr`

9.8 gazebo::rendering Namespace Reference

Rendering namespace.

Classes

- class **ArrowVisual**
Basic arrow visualization.
- class **AxisVisual**
Basic axis visualization.
- class **Camera**
Basic camera sensor.
- class **CameraVisual**
Basic camera visualization.
- class **COMVisual**
Basic Center of Mass visualization.
- class **ContactVisual**
Contact visualization.
- class **Conversions**
Conversions (p. 266) *Conversions.hh* (p. 1132) *rendering/Conversions.hh* (p. 1132).
- 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 **GzTerrainMatGen**
- class **Heightmap**
Rendering a terrain using heightmap information.
- class **JointVisual**
Visualization for joints.
- class **LaserVisual**
Visualization for laser data.
- 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 **RFIDVisual**
Visualization for RFID sensor.
- class **Road2d**
- class **RTShaderSystem**
*Implements **Ogre** (p. 123)'s Run-Time Shader system.*
- class **Scene**
Representation of an entire scene graph.
- class **SelectionObj**
Interactive selection object for models and links.
- class **SonarVisual**
Visualization for sonar data.
- class **TransmitterVisual**
Visualization for the wireless propagation data.
- class **UserCamera**
A camera used for user visualization of a scene.
- class **VideoVisual**
A visual element that displays a video as a texture.
- class **ViewController**
Base class for view controllers.
- class **Visual**
A renderable object.
- class **WindowManager**
Class to manage render windows.
- class **WireBox**
Draws a wireframe box.
- class **WrenchVisual**
Visualization for sonar data.

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

- **rendering::ScenePtr create_scene** (const std::string &_name, bool _enableVisualizations, bool _isServer=false)
*create **rendering::Scene** (p. 728) by name.*
- bool **fini** ()
teardown rendering engine.
- **rendering::ScenePtr get_scene** (const std::string &_name="")
*get pointer to **rendering::Scene** (p. 728) by name.*

- bool **init** ()
init rendering engine.
- bool **load** ()
load rendering engine.
- void **remove_scene** (const std::string &_name)
*remove a **rendering::Scene** (p. 728) 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. 751) for measure force and torque on a joint.
- class **GpsSensor**
GpsSensor (p. 341) to provide position measurement.
- class **GpuRaySensor**
- class **ImuSensor**
An IMU sensor.
- class **MultiCameraSensor**
Multiple camera sensor.
- class **Noise**
Noise (p. 603) models for sensor output signals.
- class **RaySensor**
Sensor (p. 751) with one or more rays.
- class **RFIDSensor**
Sensor (p. 751) class for RFID type of sensor.
- class **RFIDTag**

RFIDTag (p. 712) to interact with *RFIDTagSensors*.

- class **Sensor**
Base class for sensors.
- class **SensorFactory**
- class **SensorManager**
Class to manage and update all sensors.
- class **SonarSensor**
Sensor (p. 751) with sonar cone.
- class **WirelessReceiver**
Sensor (p. 751) class for receiving wireless signals.
- class **WirelessTransceiver**
Sensor (p. 751) 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
< **GpsSensor** > **GpsSensorPtr**
- typedef std::vector
< **GpuRaySensorPtr** > **GpuRaySensor_V**
- typedef boost::shared_ptr
< **GpuRaySensor** > **GpuRaySensorPtr**
- typedef std::vector< **ImuSensorPtr** > **ImuSensor_V**
- typedef boost::shared_ptr
< **ImuSensor** > **ImuSensorPtr**
- 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

- std::string **create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName) **GAZEBO_DEPRECATED**(1.10)
Deprecated.
- std::string **create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName, uint32_t _parentId)
Create a sensor using SDF.
- bool **fini** ()
shutdown the sensor generation loop.
- **SensorPtr** **get_sensor** (const std::string &_name)
Get a sensor using by name.
- bool **init** ()
initialize the sensor generation loop.
- bool **load** ()
Load the sensor library.
- void **remove_sensor** (const std::string &_sensorName)
Remove a sensor by name.
- bool **remove_sensors** ()
Remove all sensors.
- void **run_once** (bool _force=false)
Run the sensor generation one step.
- void **run_threads** ()
Run sensors in a threads. This is a non-blocking call.
- 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<GpsSensor> gazebo::sensors::GpsSensorPtr`

9.9.2.9 `typedef std::vector<GpuRaySensorPtr> gazebo::sensors::GpuRaySensor_V`

9.9.2.10 `typedef boost::shared_ptr<GpuRaySensor> gazebo::sensors::GpuRaySensorPtr`

9.9.2.11 `typedef std::vector<ImuSensorPtr> gazebo::sensors::ImuSensor_V`

9.9.2.12 `typedef boost::shared_ptr<ImuSensor> gazebo::sensors::ImuSensorPtr`

9.9.2.13 `typedef boost::shared_ptr<Noise> gazebo::sensors::NoisePtr`

9.9.2.14 `typedef std::vector<RaySensorPtr> gazebo::sensors::RaySensor_V`

9.9.2.15 `typedef boost::shared_ptr<RaySensor> gazebo::sensors::RaySensorPtr`

9.9.2.16 `typedef std::vector<RFIDSensor> gazebo::sensors::RFIDSensor_V`

9.9.2.17 `typedef boost::shared_ptr<RFIDSensor> gazebo::sensors::RFIDSensorPtr`

9.9.2.18 `typedef std::vector<RFIDTag> gazebo::sensors::RFIDTag_V`

9.9.2.19 `typedef boost::shared_ptr<RFIDTag> gazebo::sensors::RFIDTagPtr`

9.9.2.20 `typedef std::vector<SensorPtr> gazebo::sensors::Sensor_V`

9.9.2.21 `typedef Sensor*(* gazebo::sensors::SensorFactoryFn)()`

9.9.2.22 `typedef boost::shared_ptr<Sensor> gazebo::sensors::SensorPtr`

9.9.2.23 `typedef boost::shared_ptr<SonarSensor> gazebo::sensors::SonarSensorPtr`

- 9.9.2.24 `typedef std::vector<WirelessReceiver> gazebo::sensors::WirelessReceiver_V`
- 9.9.2.25 `typedef boost::shared_ptr<WirelessReceiver> gazebo::sensors::WirelessReceiverPtr`
- 9.9.2.26 `typedef std::vector<WirelessTransceiver> gazebo::sensors::WirelessTransceiver_V`
- 9.9.2.27 `typedef boost::shared_ptr<WirelessTransceiver> gazebo::sensors::WirelessTransceiverPtr`
- 9.9.2.28 `typedef std::vector<WirelessTransmitter> gazebo::sensors::WirelessTransmitter_V`
- 9.9.2.29 `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. 751) 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 **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 **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. 173)
- 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

- void **clear_buffers** ()
Clear any remaining communication buffers.
- void **fini** ()
Cleanup the transport component.
- 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.
- void **get_topic_namespaces** (std::list< std::string > &_namespaces)
Return all the namespace (world names) on the master.
- std::map< std::string,
std::list< std::string > > **getAdvertisedTopics** ()
Get a list of all the topics and their message types.
- std::list< std::string > **getAdvertisedTopics** (const std::string &_msgType)
Get a list of all the unique advertised topic names.
- bool **getMinimalComms** ()

- Get whether minimal comms has been enabled.*

 - std::string **getTopicMsgType** (const std::string &_topicName)
 - Get the message typename that is published on the given topic.*
 - bool **init** (const std::string &_master_host="", unsigned int _master_port=0)
 - Initialize the transport system.*
 - bool **is_stopped** ()
 - Is the transport system stopped?*
 - void **pause_incoming** (bool _pause)
 - Pause or unpauses incoming messages.*
 - 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.*
 - 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.*
 - void **requestNoReply** (const std::string &_worldName, const std::string &_request, const std::string &_data="")
 - Send a request and don't wait for a response.*
 - void **requestNoReply (NodePtr _node, const std::string &_request, const std::string &_data="")**
 - Send a request and don't wait for a response.*
 - void **run** ()
 - Run the transport component.*
 - void **setMinimalComms** (bool _enabled)
 - Set whether minimal comms should be used.*
 - void **stop** ()
 - Stop the transport component from running.*

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. 609) Listener.*
- class **OpenALSource**
 - *OpenAL (p. 609) 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. 102).*

9.16 Ogre Namespace Reference

9.17 ogre Namespace Reference

9.18 SimTK Namespace Reference

9.19 SkyX Namespace Reference

Chapter 10

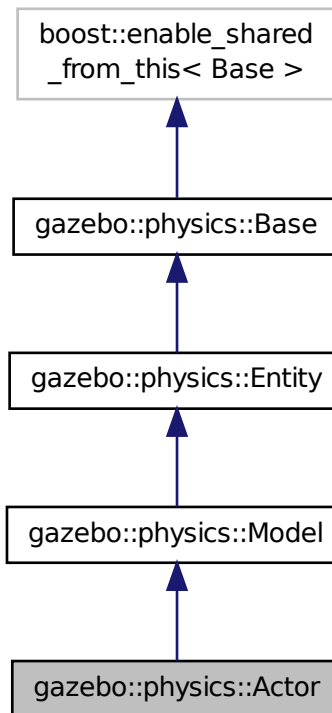
Class Documentation

10.1 gazebo::physics::Actor Class Reference

Actor (p. 125) 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. 153) 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. 125) 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

in	<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. 542).

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. 545).

10.1.3.3 virtual void `gazebo::physics::Actor::Init ()` [virtual]

Initialize the actor.

Reimplemented from `gazebo::physics::Model` (p. 546).

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	<code>_sdf</code>	SDF parameters
----	-------------------	----------------

Reimplemented from `gazebo::physics::Entity` (p. 301).

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::Base** (p. 163).

10.1.3.9 `virtual void gazebo::physics::Actor::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::Model** (p. 551).

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. 153) 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 naode 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 Pi.

- 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

Substraction, 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. 131)($M_PI * 0.5$)
- 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. 131) 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. 131) 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. 131) 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. 131) 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. 131) 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. 131) 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. 131) for subtraction
----	---------------	---------------------------------------

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. 131) 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. 131) 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. 131) 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. 131) 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

in	_angle	Angle (p. 131) 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

in	_angle	Angle (p. 131) 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

in	_angle	Angle (p. 131) 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

in	_angle	Angle (p. 131) 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. 131)($M_PI * 0.5$)

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:

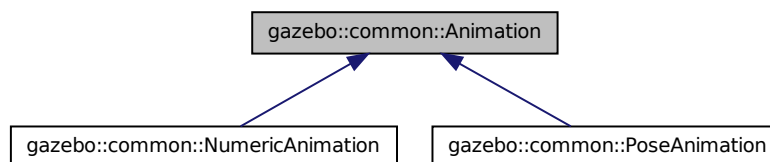
- **Angle.hh**

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:

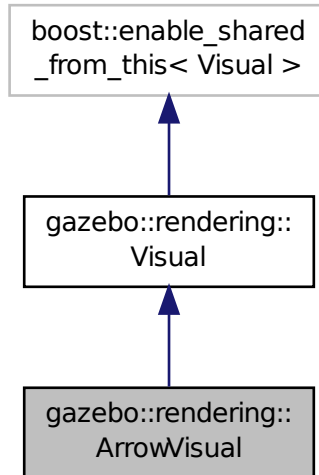
- **Animation.hh**

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. 1048).

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:

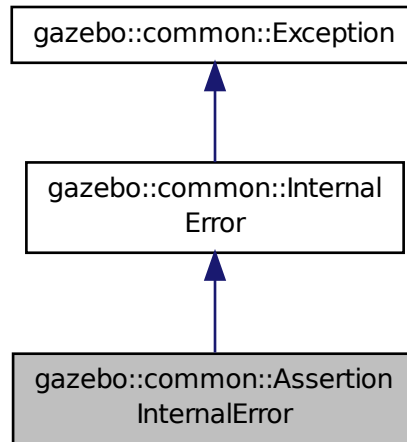
- **ArrowVisual.hh**

10.5 gazebo::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.

10.5.1 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.

10.5.2 Constructor & Destructor Documentation

- 10.5.2.1 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

<code>in</code>	<code>_function</code>	Function where assertion failed
<code>in</code>	<code>_msg</code>	Function where assertion failed

10.5.2.2 virtual gazebo::common::AssertionInternalError::~AssertionInternalError () [virtual]

Destructor.

The documentation for this class was generated from the following file:

- **Exception.hh**

10.6 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.

10.6.1 Detailed Description

An audio decoder based on FFmpeg.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 gazebo::common::AudioDecoder::AudioDecoder ()

Constructor.

10.6.2.2 virtual gazebo::common::AudioDecoder::~AudioDecoder () [virtual]

Destructor.

10.6.3 Member Function Documentation

10.6.3.1 `bool gazebo::common::AudioDecoder::Decode (uint8_t ** _outBuffer, unsigned int * _outBufferSize)`

Decode the loaded audio file.

See Also

AudioDecoder::SetFile (p. 148)

Parameters

out	<code><i>_outBuffer</i></code>	Buffer that holds the decoded audio data.
out	<code><i>_outBufferSize</i></code>	Size of the <code><i>_outBuffer</i></code> .

Returns

True if decoding was succesful.

10.6.3.2 `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. 148)

10.6.3.3 `int gazebo::common::AudioDecoder::GetSampleRate ()`

Get the sample rate from the latest decoded file.

Returns

Integer sample rate, such as 44100.

10.6.3.4 `bool gazebo::common::AudioDecoder::SetFile (const std::string & _filename)`

Set the file to decode.

Parameters

in	<code><i>_filename</i></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:

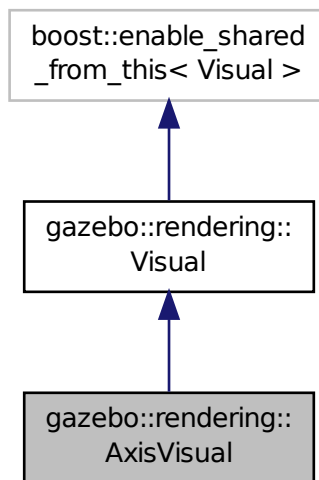
- **AudioDecoder.hh**

10.7 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

10.7.1 Detailed Description

Basic axis visualization.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 gazebo::rendering::AxisVisual::AxisVisual (const std::string & *_name*, VisualPtr *_vis*)

Constructor.

Parameters

in	<i>_name</i>	Name of the AxisVisual (p. 149)
in	<i>_vis</i>	Parent visual

10.7.2.2 virtual gazebo::rendering::AxisVisual::~~AxisVisual () [virtual]

Destructor.

10.7.3 Member Function Documentation

10.7.3.1 virtual void gazebo::rendering::AxisVisual::Load () [virtual]

Load the axis visual.

Reimplemented from **gazebo::rendering::Visual** (p. 1048).

10.7.3.2 void gazebo::rendering::AxisVisual::ScaleXAxis (const math::Vector3 & *_scale*)

Scale the X axis.

Parameters

in	<i>_scale</i>	Scaling factor
----	---------------	----------------

10.7.3.3 void gazebo::rendering::AxisVisual::ScaleYAxis (const math::Vector3 & *_scale*)

Scale the Y axis.

Parameters

in	<i>_scale</i>	Scaling factor
----	---------------	----------------

10.7.3.4 void gazebo::rendering::AxisVisual::ScaleZAxis (const math::Vector3 & *_scale*)

Scale the Z axis.

Parameters

in	<i>_scale</i>	Scaling factor
----	---------------	----------------

10.7.3.5 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

10.7.3.6 void gazebo::rendering::AxisVisual::ShowRotation (unsigned int *_axis*)

Load the rotation tube.

The documentation for this class was generated from the following file:

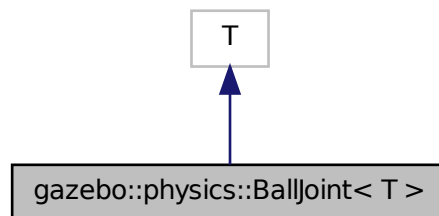
- **AxisVisual.hh**

10.8 gazebo::physics::BallJoint< T > Class Template Reference

Base (p. 153) class for a ball joint.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::BallJoint< T >:



Public Member Functions

- **BallJoint** (BasePtr *_parent*)

Constructor.

- virtual `~BallJoint ()`

Destructor.

- virtual unsigned int `GetAngleCount () const`
- virtual `math::Angle GetHighStop (int)`
- virtual `math::Angle GetLowStop (int)`
- void `Load (sdf::ElementPtr _sdf)`

*Template to ::Load the **BallJoint** (p. 151).*

- virtual void `SetAxis (int, const math::Vector3 &)`
- virtual void `SetHighStop (int, math::Angle)`
- virtual void `SetLowStop (int, math::Angle)`

10.8.1 Detailed Description

```
template<class T>class gazebo::physics::BallJoint< T >
```

Base (p. 153) class for a ball joint.

Each physics engine should implement this class.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 `template<class T> gazebo::physics::BallJoint< T >::BallJoint (BasePtr _parent) [inline], [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Pointer to the parent link.
----	----------------------	-----------------------------

10.8.2.2 `template<class T> virtual gazebo::physics::BallJoint< T >::~~BallJoint () [inline], [virtual]`

Destructor.

10.8.3 Member Function Documentation

10.8.3.1 `template<class T> virtual unsigned int gazebo::physics::BallJoint< T >::GetAngleCount () const [inline], [virtual]`

10.8.3.2 `template<class T> virtual math::Angle gazebo::physics::BallJoint< T >::GetHighStop (int) [inline], [virtual]`

10.8.3.3 `template<class T> virtual math::Angle gazebo::physics::BallJoint< T >::GetLowStop (int) [inline], [virtual]`

10.8.3.4 `template<class T> void gazebo::physics::BallJoint< T >::Load (sdf::ElementPtr _sdf) [inline]`

Template to ::Load the **BallJoint** (p. 151).

Parameters

in	<code>_sdf</code>	SDF to load the joint from.
----	-------------------	-----------------------------

10.8.3.5 `template<class T> virtual void gazebo::physics::BallJoint< T >::SetAxis (int , const math::Vector3 &)`
`[inline], [virtual]`

10.8.3.6 `template<class T> virtual void gazebo::physics::BallJoint< T >::SetHighStop (int , math::Angle)`
`[inline], [virtual]`

10.8.3.7 `template<class T> virtual void gazebo::physics::BallJoint< T >::SetLowStop (int , math::Angle)`
`[inline], [virtual]`

The documentation for this class was generated from the following file:

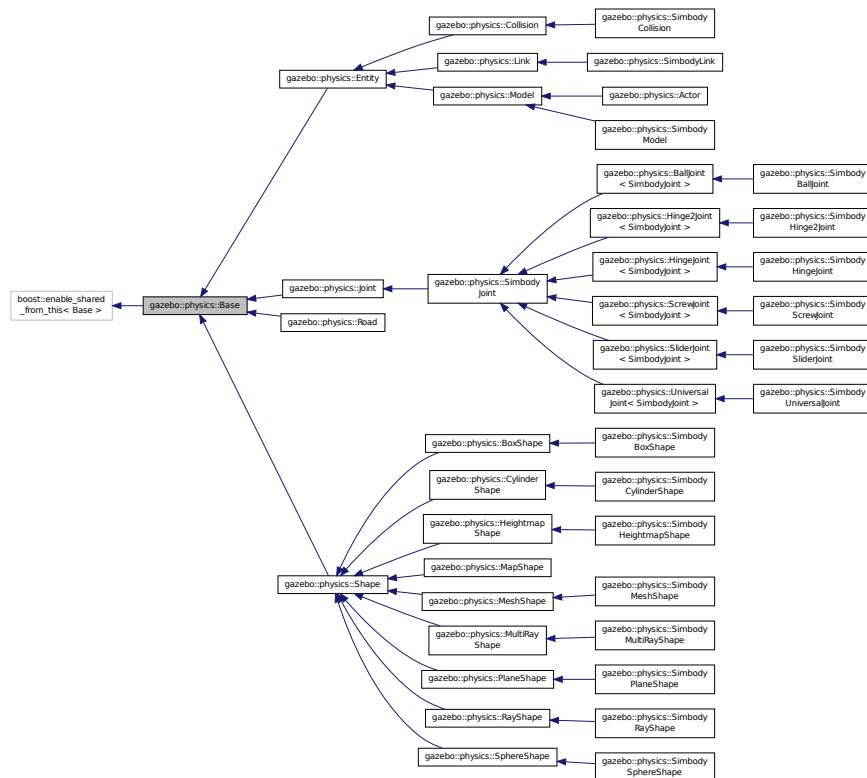
- [BallJoint.hh](#)

10.9 gazebo::physics::Base Class Reference

Base (p. 153) 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, **SHAPE** =
0x00002000,
BOX_SHAPE = 0x00004000, **CYLINDER_SHAPE** = 0x00008000, **HEIGHTMAP_SHAPE** = 0x00010000, **MAP_**
_SHAPE = 0x00020000,
MULTIRAY_SHAPE = 0x00040000, **RAY_SHAPE** = 0x00080000, **PLANE_SHAPE** = 0x00100000, **SPHERE_**
SHAPE = 0x00200000,
MESH_SHAPE = 0x00400000, **SENSOR_COLLISION** = 0x00800000 }

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** **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. 1070) 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. 156)'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.
- **Base_V::iterator childrenEnd**
End of the children vector.
- **BasePtr parent**
Parent of this entity.
- **sdf::ElementPtr sdf**
The SDF values for this object.
- **WorldPtr world**
Pointer to the world.

10.9.1 Detailed Description

Base (p. 153) class for most physics classes.

10.9.2 Member Enumeration Documentation

10.9.2.1 enum gazebo::physics::Base::EntityType

Unique identifiers for all entity types.

Enumerator

- BASE** **Base** (p. 153) type.
- ENTITY** **Entity** (p. 293) type.
- MODEL** **Model** (p. 537) type.
- LINK** **Link** (p. 455) type.
- COLLISION** **Collision** (p. 213) type.
- ACTOR** **Actor** (p. 125) type.
- LIGHT** **Light** type.
- VISUAL** **Visual** type.
- JOINT** **Joint** (p. 411) type.
- BALL_JOINT** **BallJoint** (p. 151) type.
- HINGE2_JOINT** **Hing2Joint** type.
- HINGE_JOINT** **HingeJoint** (p. 387) type.
- SLIDER_JOINT** **SliderJoint** (p. 886) type.
- SCREW_JOINT** **ScrewJoint** (p. 746) type.
- UNIVERSAL_JOINT** **UniversalJoint** (p. 976) type.
- SHAPE** **Shape** (p. 775) type.
- BOX_SHAPE** **BoxShape** (p. 169) type.
- CYLINDER_SHAPE** **CylinderShape** (p. 268) type.
- HEIGHTMAP_SHAPE** **HeightmapShape** (p. 380) type.
- MAP_SHAPE** **MapShape** (p. 492) type.

MULTIRAY_SHAPE MultiRayShape (p. 578) type.

RAY_SHAPE RayShape (p. 700) type.

PLANE_SHAPE PlaneShape (p. 642) type.

SPHERE_SHAPE SphereShape (p. 898) type.

MESH_SHAPE MeshShape (p. 534) type.

SENSOR_COLLISION Indicates a collision shape used for sensing.

10.9.3 Constructor & Destructor Documentation

10.9.3.1 gazebo::physics::Base::Base (BasePtr _parent) [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent of this object
----	----------------	-----------------------

10.9.3.2 virtual gazebo::physics::Base::~Base () [virtual]

Destructor.

10.9.4 Member Function Documentation

10.9.4.1 void gazebo::physics::Base::AddChild (BasePtr _child)

Add a child to this entity.

Parameters

in	<i>_child</i>	Child entity.
----	---------------	---------------

10.9.4.2 void gazebo::physics::Base::AddType (EntityType _type)

Add a type specifier.

Parameters

in	<i>_type</i>	New type to append to this objects type definition.
----	--------------	---

10.9.4.3 void gazebo::physics::Base::ComputeScopedName () [protected]

Compute the scoped name of this object based on its parents.

See Also

Base::GetScopedName (p. 159)

10.9.4.4 virtual void gazebo::physics::Base::Fini () [virtual]

Finalize the object.

Reimplemented in **gazebo::physics::Actor** (p.128), **gazebo::physics::Link** (p.463), **gazebo::physics::Model** (p.542), **gazebo::physics::Entity** (p.297), **gazebo::physics::Collision** (p.216), and **gazebo::physics::Simbody-Link** (p.818).

10.9.4.5 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

10.9.4.6 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.

10.9.4.7 BasePtr gazebo::physics::Base::GetChild (const std::string & *_name*)

Get a child by name.

Parameters

<i>in</i>	<i>_name</i>	Name of the child.
-----------	--------------	--------------------

Returns

A pointer to the object, NULL if not found

10.9.4.8 unsigned int gazebo::physics::Base::GetChildCount () const

Get the number of children.

Returns

The number of children.

10.9.4.9 `uint32_t gazebo::physics::Base::GetId () const`

Return the ID of this entity.

This id is unique.

Returns

Integer ID.

10.9.4.10 `std::string gazebo::physics::Base::GetName () const`

Return the name of the entity.

Returns

Name of the entity.

10.9.4.11 `BasePtr gazebo::physics::Base::GetParent () const`

Get the parent.

Returns

Pointer to the parent entity.

10.9.4.12 `int gazebo::physics::Base::GetParentId () const`

Return the ID of the parent.

Returns

Integer ID.

10.9.4.13 `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.

10.9.4.14 `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.

10.9.4.15 `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. 128), and `gazebo::physics::Model` (p. 545).

10.9.4.16 `unsigned int gazebo::physics::Base::GetType () const`

Get the full type definition.

Returns

The full type definition.

10.9.4.17 `const WorldPtr& gazebo::physics::Base::GetWorld () const`

Get the **World** (p. 1070) this object is in.

Returns

The **World** (p. 1070) this object is part of.

10.9.4.18 `bool gazebo::physics::Base::HasType (const EntityType & _t) const`

Returns true if this object's type definition has the given type.

Parameters

in	_t	Type to check.
----	----	----------------

Returns

True if this object's type definition has the.

10.9.4.19 `virtual void gazebo::physics::Base::Init () [inline],[virtual]`

Initialize the object.

Reimplemented in `gazebo::physics::Joint` (p. 424), `gazebo::physics::RayShape` (p. 704), `gazebo::physics::Link` (p. 469), `gazebo::physics::Actor` (p. 128), `gazebo::physics::Model` (p. 546), `gazebo::physics::MapShape` (p. 495), `gazebo::physics::HingeJoint< SimbodyJoint >` (p. 389), `gazebo::physics::Collision` (p. 220), `gazebo::physics::HeightmapShape` (p. 384), `gazebo::physics::MeshShape` (p. 536), `gazebo::physics::SimbodyHinge2Joint` (p. 797), `gazebo::physics::SimbodyScrewJoint` (p. 847), `gazebo::physics::MultiRayShape` (p. 585), `gazebo::physics::PlaneShape` (p. 645), `gazebo::physics::SimbodyBallJoint` (p. 782), `gazebo::physics::SimbodyLink` (p. 820), `gazebo::physics::Road` (p. 719), `gazebo::physics::Shape` (p. 777), `gazebo::physics::SimbodyUniversalJoint` (p. 862), `gazebo::physics::SphereShape` (p. 900), `gazebo::physics::BoxShape` (p. 171), `gazebo::physics::-`

CylinderShape (p. 271), **gazebo::physics::SimbodyHeightmapShape** (p. 792), **gazebo::physics::SimbodyModel** (p. 826), and **gazebo::physics::SimbodyMeshShape** (p. 824).

10.9.4.20 `bool gazebo::physics::Base::IsSelected () const`

True if the entity is selected by the user.

Returns

True if the entity is selected.

10.9.4.21 `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. 424), **gazebo::physics::SimbodySliderJoint** (p. 853), **gazebo::physics::Link** (p. 469), **gazebo::physics::Actor** (p. 128), **gazebo::physics::Entity** (p. 301), **gazebo::physics::Model** (p. 546), **gazebo::physics::MapShape** (p. 495), **gazebo::physics::Hinge2Joint**< **SimbodyJoint** > (p. 387), **gazebo::physics::Collision** (p. 220), **gazebo::physics::HeightmapShape** (p. 385), **gazebo::physics::BallJoint**< **SimbodyJoint** > (p. 152), **gazebo::physics::ScrewJoint**< **SimbodyJoint** > (p. 748), **gazebo::physics::UniversalJoint**< **SimbodyJoint** > (p. 978), **gazebo::physics::HingeJoint**< **SimbodyJoint** > (p. 389), **gazebo::physics::SliderJoint**< **SimbodyJoint** > (p. 888), **gazebo::physics::SimbodyCollision** (p. 787), **gazebo::physics::SimbodyHingeJoint** (p. 802), **gazebo::physics::SimbodyLink** (p. 820), **gazebo::physics::Road** (p. 719), **gazebo::physics::SimbodyHinge2Joint** (p. 797), **gazebo::physics::SimbodyUniversalJoint** (p. 862), **gazebo::physics::SimbodyJoint** (p. 810), **gazebo::physics::SimbodyScrewJoint** (p. 847), **gazebo::physics::SimbodyBallJoint** (p. 782), **gazebo::physics::SimbodyModel** (p. 826), and **gazebo::physics::SimbodyMeshShape** (p. 824).

10.9.4.22 `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. 153) object to compare with.
-----------------	-------------------	--

Returns

True if the entities are the same.

10.9.4.23 `void gazebo::physics::Base::Print (const std::string & _prefix)`

Print this object to screen via gzmsg.

Parameters

in	<code>_prefix</code>	Usually a set of spaces.
----	----------------------	--------------------------

10.9.4.24 `virtual void gazebo::physics::Base::RemoveChild (unsigned int _id) [virtual]`

Remove a child from this entity.

Parameters

in	<code>_id</code>	ID of the child to remove.
----	------------------	----------------------------

10.9.4.25 `void gazebo::physics::Base::RemoveChild (const std::string & _name)`

Remove a child by name.

Parameters

in	<code>_name</code>	Name of the child.
----	--------------------	--------------------

10.9.4.26 `void gazebo::physics::Base::RemoveChildren ()`

Remove all children.

10.9.4.27 `virtual void gazebo::physics::Base::Reset () [virtual]`

Reset the object.

Reimplemented in `gazebo::physics::Joint` (p. 425), `gazebo::physics::Model` (p. 547), `gazebo::physics::Link` (p. 470), `gazebo::physics::Entity` (p. 301), and `gazebo::physics::SimbodyJoint` (p. 810).

10.9.4.28 `virtual void gazebo::physics::Base::Reset (Base::EntityType _resetType) [virtual]`

Calls recursive Reset on one of the `Base::EntityType` (p. 156)'s.

Parameters

in	<code>_resetType</code>	The type of reset operation
----	-------------------------	-----------------------------

10.9.4.29 `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. 302).

10.9.4.30 void gazebo::physics::Base::SetParent (BasePtr *_parent*)

Set the parent.

Parameters

in	<i>_parent</i>	Parent object.
----	----------------	----------------

10.9.4.31 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	<i>_v</i>	Set to True if the object should be saved.
----	-----------	--

10.9.4.32 virtual bool gazebo::physics::Base::SetSelected (bool *_show*) [virtual]

Set whether this entity has been selected by the user through the gui.

Parameters

in	<i>_show</i>	True to set this entity as selected.
----	--------------	--------------------------------------

Reimplemented in **gazebo::physics::Link** (p. 474).

10.9.4.33 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	<i>_newWorld</i>	The new World (p. 1070) this object is part of.
----	------------------	--

10.9.4.34 virtual void gazebo::physics::Base::Update () [inline],[virtual]

Update the object.

Reimplemented in **gazebo::physics::Joint** (p. 428), **gazebo::physics::MultiRayShape** (p. 585), **gazebo::physics::RayShape** (p. 705), **gazebo::physics::Actor** (p. 129), **gazebo::physics::Model** (p. 551), **gazebo::physics::MapShape** (p. 496), **gazebo::physics::MeshShape** (p. 537), and **gazebo::physics::SimbodyRayShape** (p. 842).

10.9.4.35 virtual void gazebo::physics::Base::UpdateParameters (sdf::ElementPtr *_sdf*) [virtual]

Update the parameters using new sdf values.

Parameters

in	_sdf	Update the object's parameters based on SDF values.
----	------	---

Reimplemented in **gazebo::physics::Joint** (p. 428), **gazebo::physics::Actor** (p. 129), **gazebo::physics::Link** (p. 475), **gazebo::physics::Model** (p. 551), **gazebo::physics::Entity** (p. 303), and **gazebo::physics::Collision** (p. 222).

10.9.5 Member Data Documentation

10.9.5.1 Base_V gazebo::physics::Base::children [protected]

Children of this entity.

10.9.5.2 Base_V::iterator gazebo::physics::Base::childrenEnd [protected]

End of the children vector.

10.9.5.3 BasePtr gazebo::physics::Base::parent [protected]

Parent of this entity.

10.9.5.4 sdf::ElementPtr gazebo::physics::Base::sdf [protected]

The SDF values for this object.

10.9.5.5 WorldPtr gazebo::physics::Base::world [protected]

Pointer to the world.

The documentation for this class was generated from the following file:

- **Base.hh**

10.10 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 \sim **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)
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.

10.10.1 Detailed Description

Mathematical representation of a box and related functions.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 gazebo::math::Box::Box ()

Default constructor.

10.10.2.2 `gazebo::math::Box::Box (const Vector3 & _min, const Vector3 & _max)`

Constructor.

Parameters

in	<code>_min</code>	Minimum corner of the box
in	<code>_max</code>	Maximum corner of the box

10.10.2.3 `gazebo::math::Box::Box (const Box & _b)`

Copy Constructor.

Parameters

in	<code>_b</code>	Box (p. 164) to copy
----	-----------------	-----------------------------

10.10.2.4 `virtual gazebo::math::Box::~~Box () [virtual]`

Destructor.

10.10.3 Member Function Documentation

10.10.3.1 `math::Vector3 gazebo::math::Box::GetCenter () const`

Get the box center.

Returns

The center position of the box

10.10.3.2 `math::Vector3 gazebo::math::Box::GetSize () const`

Get the size of the box.

Returns

Size of the box

10.10.3.3 `double gazebo::math::Box::GetXLength () const`

Get the length along the x dimension.

Returns

Double value of the length in the x dimension

10.10.3.4 `double gazebo::math::Box::GetYLength () const`

Get the length along the y dimension.

Returns

Double value of the length in the y dimension

10.10.3.5 `double gazebo::math::Box::GetZLength () const`

Get the length along the z dimension.

Returns

Double value of the length in the z dimension

10.10.3.6 `void gazebo::math::Box::Merge (const Box & _box)`

Merge a box with this box.

Parameters

<code>in</code>	<code>_box</code>	Box (p. 164) to add to this box
-----------------	-------------------	--

10.10.3.7 `Box gazebo::math::Box::operator+ (const Box & _b) const`

Addition operator.

result = this + `_b`

Parameters

<code>in</code>	<code>_b</code>	Box (p. 164) to add
-----------------	-----------------	----------------------------

Returns

The new box

10.10.3.8 `const Box& gazebo::math::Box::operator+= (const Box & _b)`

Addition set operator.

this = this + `_b`

Parameters

<code>in</code>	<code>_b</code>	Box (p. 164) to add
-----------------	-----------------	----------------------------

Returns

This new box

10.10.3.9 **Box** gazebo::math::Box::operator- (const Vector3 & _v)

Subtract a vector from the min and max values.

Parameters

_v	The vector to use during subtraction
----	--------------------------------------

Returns

The new box

10.10.3.10 **Box&** gazebo::math::Box::operator= (const Box & _b)

Assignment operator.

Set this box to the parameter

Parameters

in	_b	Box (p. 164) to copy
----	----	-----------------------------

Returns

The new box.

10.10.3.11 **bool** gazebo::math::Box::operator== (const Box & _b)

Equality test operator.

Parameters

in	_b	Box (p. 164) to test
----	----	-----------------------------

Returns

True if equal

10.10.4 Friends And Related Function Documentation

10.10.4.1 **std::ostream&** operator<< (**std::ostream & _out**, const gazebo::math::Box & _b) [friend]

Output operator.

Parameters

<i>in</i>	<i>_out</i>	Output stream
<i>in</i>	<i>_b</i>	Box (p. 164) to output to the stream

Returns

The stream

10.10.5 Member Data Documentation

10.10.5.1 Vector3 gazebo::math::Box::max

Maximum corner of the box.

10.10.5.2 Vector3 gazebo::math::Box::min

Minimum corner of the box.

The documentation for this class was generated from the following file:

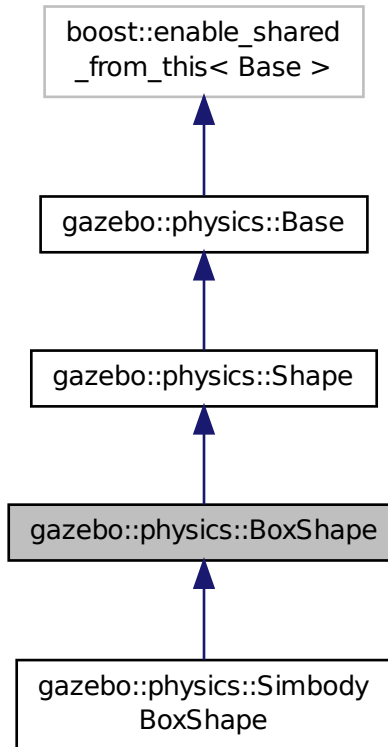
- **Box.hh**

10.11 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

10.11.1 Detailed Description

Box geometry primitive.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 gazebo::physics::BoxShape::BoxShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent Collision (p. 213).
----	----------------	-----------------------------------

10.11.2.2 virtual gazebo::physics::BoxShape::~~BoxShape () [virtual]

Destructor.

10.11.3 Member Function Documentation

10.11.3.1 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. 777).

10.11.3.2 math::Vector3 gazebo::physics::BoxShape::GetSize () const

Get the size of the box.

Returns

The size of each side of the box.

10.11.3.3 virtual void gazebo::physics::BoxShape::Init () [virtual]

Initialize the box.

Implements **gazebo::physics::Shape** (p. 777).

10.11.3.4 virtual void gazebo::physics::BoxShape::ProcessMsg (const msgs::Geometry & *_msg*) [virtual]

Process a geometry message.

Parameters

in	<code>_msg</code>	The message to set values from.
----	-------------------	---------------------------------

Implements `gazebo::physics::Shape` (p. 778).

10.11.3.5 `virtual void gazebo::physics::BoxShape::SetScale (const math::Vector3 & _scale) [virtual]`

Set the scale of the box.

Parameters

in	<code>_scale</code>	Scale of the box.
----	---------------------	-------------------

Implements `gazebo::physics::Shape` (p. 778).

10.11.3.6 `virtual void gazebo::physics::BoxShape::SetSize (const math::Vector3 & _size) [virtual]`

Set the size of the box.

Parameters

in	<code>_size</code>	Size of each side of the box.
----	--------------------	-------------------------------

Reimplemented in `gazebo::physics::SimbodyBoxShape` (p. 785).

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

The documentation for this class was generated from the following file:

- **BoxShape.hh**

10.12 gazebo::common::BVHLoader Class Reference

Handles loading BVH animation files.

```
#include <common/common.hh>
```

Public Member Functions

- **BVHLoader ()**
Constructor.
- **~BVHLoader ()**
Destructor.
- **Skeleton * Load** (const std::string &_filename, double _scale)
Load a BVH file.

10.12.1 Detailed Description

Handles loading BVH animation files.

10.12.2 Constructor & Destructor Documentation

10.12.2.1 gazebo::common::BVHLoader::BVHLoader ()

Constructor.

10.12.2.2 gazebo::common::BVHLoader::~~BVHLoader ()

Desutrctor.

10.12.3 Member Function Documentation

10.12.3.1 Skeleton* gazebo::common::BVHLoader::Load (const std::string & *_filename*, double *_scale*)

Load a BVH file.

Parameters

in	<i>_filename</i>	BVH file to load
in	<i>_scale</i>	Scaling factor to apply to the skeleton

Returns

A pointer to a new **Skeleton** (p. 866)

The documentation for this class was generated from the following file:

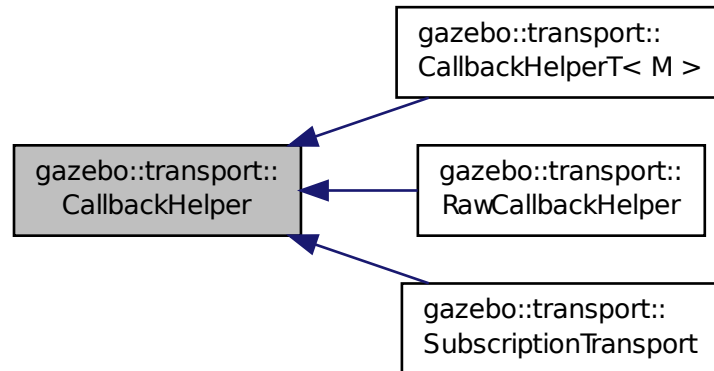
- **BVHLoader.hh**

10.13 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?

Protected Attributes

- bool **latching**
True means that the callback helper will get the last published message on the topic.

10.13.1 Detailed Description

A helper class to handle callbacks when messages arrive.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 gazebo::transport::CallbackHelper::CallbackHelper (bool *_latching* = false)

Constructor.

Parameters

in	<i>_latching</i>	Set to true to make the callback helper latching.
----	------------------	---

10.13.2.2 virtual gazebo::transport::CallbackHelper::~~CallbackHelper () [virtual]

Destructor.

10.13.3 Member Function Documentation

10.13.3.1 unsigned int gazebo::transport::CallbackHelper::GetId () const

Get the unique ID of this callback.

Returns

The unique ID of this callback.

10.13.3.2 bool gazebo::transport::CallbackHelper::GetLatching () const

Is the callback latching?

Returns

true if the callback is latching, false otherwise

10.13.3.3 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. 692), and [gazebo::transport::CallbackHelperT< M >](#) (p. 178).

10.13.3.4 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

in	_newdata	Incoming data to be processed
----	----------	-------------------------------

Returns

true if successfully processed; false otherwise

Parameters

in	_cb	If non-null, callback to be invoked which signals that transmission is complete.
in	_id	ID associated with the message data.

Implemented in [gazebo::transport::RawCallbackHelper](#) (p. 692), [gazebo::transport::CallbackHelperT< M >](#) (p. 178), and [gazebo::transport::SubscriptionTransport](#) (p. 932).

10.13.3.5 `virtual bool gazebo::transport::CallbackHelper::HandleMessage (MessagePtr _newMsg) [pure virtual]`

Process new incoming message.

Parameters

in	_newMsg	Incoming message to be processed
----	---------	----------------------------------

Returns

true if successfully processed; false otherwise

Implemented in [gazebo::transport::RawCallbackHelper](#) (p. 692), [gazebo::transport::CallbackHelperT< M >](#) (p. 178), and [gazebo::transport::SubscriptionTransport](#) (p. 932).

10.13.3.6 `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. 692), [gazebo::transport::CallbackHelperT< M >](#) (p. 179), and [gazebo::transport::SubscriptionTransport](#) (p. 933).

10.13.4 Member Data Documentation

10.13.4.1 `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:

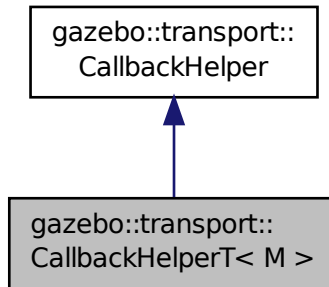
- [CallbackHelper.hh](#)

10.14 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

10.14.1 Detailed Description

```
template<class M>class gazebo::transport::CallbackHelperT< M >
```

Callback helper Template.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 `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.

10.14.3 Member Function Documentation

10.14.3.1 `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. 175).

References `gzthrow`, and `NULL`.

10.14.3.2 `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. 175).

10.14.3.3 `template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::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. 176).

10.14.3.4 `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. 176).

The documentation for this class was generated from the following file:

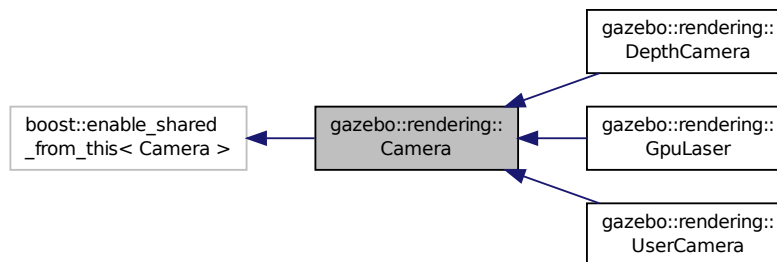
- `CallbackHelper.hh`

10.15 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 **CreateRenderTarget** (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** ()
 - 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 name.
- `double GetNearClip ()`
Get the near clip distance.
- `Ogre::Camera * GetOgreCamera () const`
Get a pointer to the ogre camera.
- `Ogre::SceneNode * GetPitchNode () const`
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 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 ()`
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 ()`
Get the global pose of the camera.
- `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 **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. 202) 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.*
- Ogre::SceneNode * **pitchNode**
 - Scene** (p. 728) nod that controls camera pitch.*

- **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. 728) node that controls camera position.*
- **std::string screenshotPath**
Path to saved screenshots.
- **sdf::ElementPtr sdf**
***Camera** (p. 179)'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.

10.15.1 Detailed Description

Basic camera sensor.

This is the base class for all cameras.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 `gazebo::rendering::Camera::Camera (const std::string & _namePrefix, ScenePtr _scene, bool _autoRender = true)`

Constructor.

Parameters

<code>in</code>	<code>_namePrefix</code>	Unique prefix name for the camera.
<code>in</code>	<code>_scene</code>	Scene (p. 728) that will contain the camera
<code>in</code>	<code>_autoRender</code>	Almost everyone should leave this as true.

10.15.2.2 `virtual gazebo::rendering::Camera::~~Camera () [virtual]`

Destructor.

10.15.3 Member Function Documentation

10.15.3.1 `virtual void gazebo::rendering::Camera::AnimationComplete () [protected],[virtual]`

Internal function used to indicate that an animation has completed.

Reimplemented in `gazebo::rendering::UserCamera` (p. 981).

10.15.3.2 `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	<code>_visualName</code>	Name of the visual to attach the camera to
in	<code>_inheritOrientation</code>	True means camera acquires the visual's orientation
in	<code>_minDist</code>	Minimum distance the camera is allowed to get to the visual
in	<code>_maxDist</code>	Maximum distance the camera is allowed to get from the visual

10.15.3.3 `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	<code>_id</code>	ID of the visual to attach the camera to
in	<code>_inheritOrientation</code>	True means camera acquires the visual's orientation
in	<code>_minDist</code>	Minimum distance the camera is allowed to get to the visual
in	<code>_maxDist</code>	Maximum distance the camera is allowed to get from the visual

10.15.3.4 `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	<code>_visualName</code>	Name of the visual to attach the camera to
in	<code>_inheritOrientation</code>	True means camera acquires the visual's orientation
in	<code>_minDist</code>	Minimum distance the camera is allowed to get to the visual
in	<code>_maxDist</code>	Maximum distance the camera is allowed to get from the visual

Returns

True on success

10.15.3.5 `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

10.15.3.6 `virtual bool gazebo::rendering::Camera::AttachToVisualImpl (VisualPtr _visual, bool _inheritOrientation, double _minDist = 0, double _maxDist = 0)` [protected],[virtual]

Attach the camera to a visual.

Parameters

in	<i>_visual</i>	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

Reimplemented in `gazebo::rendering::UserCamera` (p. 982).

10.15.3.7 `template<typename T> event::ConnectionPtr gazebo::rendering::Camera::ConnectNewImageFrame (T _subscriber)` [inline]

Connect to the new image 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.

References gazebo::event::EventT< T >::Connect(), and newImageFrame.

10.15.3.8 void gazebo::rendering::Camera::CreateRenderTexture (const std::string & *_textureName*)

Set the render target.

Parameters

in	<i>_textureName</i>	Name of the new render texture
----	---------------------	--------------------------------

10.15.3.9 void gazebo::rendering::Camera::DisconnectNewImageFrame (event::ConnectionPtr & *_c*) [inline]

Disconnect from an image frame.

Parameters

in	<i>_c</i>	The connection to disconnect
----	-----------	------------------------------

References gazebo::event::EventT< T >::Disconnect(), and newImageFrame.

10.15.3.10 void gazebo::rendering::Camera::EnableSaveFrame (bool *_enable*)

Enable or disable saving.

Parameters

in	<i>_enable</i>	Set to True to enable saving of frames
----	----------------	--

10.15.3.11 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. 348), **gazebo::rendering::DepthCamera** (p. 275), and **gazebo::rendering::UserCamera** (p. 982).

10.15.3.12 float gazebo::rendering::Camera::GetAspectRatio () const

Get the aspect ratio.

Returns

The aspect ratio (width / height) in pixels

10.15.3.13 virtual float gazebo::rendering::Camera::GetAvgFPS () [inline],[virtual]

Get the average FPS.

Returns

The average frames per second

10.15.3.14 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

10.15.3.15 bool gazebo::rendering::Camera::GetCaptureData () const

Return the value of this->captureData.

Returns

True if the camera is set to capture data.

10.15.3.16 math::Vector3 gazebo::rendering::Camera::GetDirection () const

Get the camera's direction vector.

Returns

Direction the camera is facing

10.15.3.17 double gazebo::rendering::Camera::GetFarClip ()

Get the far clip distance.

Returns

Far clip distance

10.15.3.18 std::string gazebo::rendering::Camera::GetFrameFilename () [protected]

Get the next frame filename based on SDF parameters.

Returns

The frame's filename

10.15.3.19 `math::Angle gazebo::rendering::Camera::GetHFOV () const`

Get the camera FOV (horizontal)

Returns

The horizontal field of view

10.15.3.20 `size_t gazebo::rendering::Camera::GetImageByteSize () const`

Get the image size in bytes.

Returns

Size in bytes

10.15.3.21 `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><i>_width</i></code>	Width of an image
<code>in</code>	<code><i>_height</i></code>	Height of an image
<code>in</code>	<code><i>_format</i></code>	Image format

Returns

Size of an image based on the parameters

10.15.3.22 `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>_i</i></code>	Index of the camera's texture (0 = RGB, 1 = depth).
-----------------	------------------------	---

Returns

Pointer to the raw data, null if data is not available.

10.15.3.23 `unsigned int gazebo::rendering::Camera::GetImageDepth () const`

Get the depth of the image.

Returns

Depth of the image

10.15.3.24 `std::string gazebo::rendering::Camera::GetImageFormat () const`

Get the string representation of the image format.

Returns

String representation of the image format.

10.15.3.25 `virtual unsigned int gazebo::rendering::Camera::GetImageHeight () const` [virtual]

Get the height of the image.

Returns

Image height

Reimplemented in **`gazebo::rendering::UserCamera`** (p. 983).

10.15.3.26 `virtual unsigned int gazebo::rendering::Camera::GetImageWidth () const` [virtual]

Get the width of the image.

Returns

Image width

Reimplemented in **`gazebo::rendering::UserCamera`** (p. 983).

10.15.3.27 `bool gazebo::rendering::Camera::GetInitialized () const`

Return true if the camera has been initialized.

Returns

True if initialized was successful

10.15.3.28 `common::Time gazebo::rendering::Camera::GetLastRenderWallTime ()`

Get the last time the camera was rendered.

Returns

Time the camera was last rendered

10.15.3.29 `std::string gazebo::rendering::Camera::GetName () const`

Get the camera's name.

Returns

The name of the camera

10.15.3.30 `double gazebo::rendering::Camera::GetNearClip ()`

Get the near clip distance.

Returns

Near clip distance

10.15.3.31 `Ogre::Camera* gazebo::rendering::Camera::GetOgreCamera () const`

Get a pointer to the ogre camera.

Returns

Pointer to the OGRE camera

10.15.3.32 `Ogre::SceneNode* gazebo::rendering::Camera::GetPitchNode () const`

Get the camera's pitch scene node.

Returns

The pitch node the camera is attached to

10.15.3.33 `double gazebo::rendering::Camera::GetRenderRate () const`

Get the render Hz rate.

Returns

The Hz rate

10.15.3.34 `Ogre::Texture* gazebo::rendering::Camera::GetRenderTexture () const`

Get the render texture.

Returns

Pointer to the render texture

10.15.3.35 `math::Vector3 gazebo::rendering::Camera::GetRight ()`

Get the viewport right vector.

Returns

The viewport right vector

10.15.3.36 `ScenePtr gazebo::rendering::Camera::GetScene () const`

Get the scene this camera is in.

Returns

Pointer to scene containing this camera

10.15.3.37 `Ogre::SceneNode* gazebo::rendering::Camera::GetSceneNode () const`

Get the camera's scene node.

Returns

The scene node the camera is attached to

10.15.3.38 `std::string gazebo::rendering::Camera::GetScreenshotPath () const`

Get the path to saved screenshots.

Returns

Path to saved screenshots.

10.15.3.39 `unsigned int gazebo::rendering::Camera::GetTextureHeight () const`

Get the height of the off-screen render texture.

Returns

Render texture height

10.15.3.40 `unsigned int gazebo::rendering::Camera::GetTextureWidth () const`

Get the width of the off-screen render texture.

Returns

Render texture width

10.15.3.41 `virtual unsigned int gazebo::rendering::Camera::GetTriangleCount () [inline],[virtual]`

Get the triangle count.

Returns

The current triangle count

10.15.3.42 `math::Vector3 gazebo::rendering::Camera::GetUp ()`

Get the viewport up vector.

Returns

The viewport up vector

10.15.3.43 `math::Angle gazebo::rendering::Camera::GetVFOV () const`

Get the camera FOV (vertical)

Returns

The vertical field of view

10.15.3.44 `Ogre::Viewport* gazebo::rendering::Camera::GetViewport () const`

Get a pointer to the `Ogre::Viewport`.

Returns

Pointer to the `Ogre::Viewport`

10.15.3.45 `unsigned int gazebo::rendering::Camera::GetViewportHeight () const`

Get the viewport height in pixels.

Returns

The viewport height

10.15.3.46 `unsigned int gazebo::rendering::Camera::GetViewportWidth () const`

Get the viewport width in pixels.

Returns

The viewport width

10.15.3.47 `unsigned int gazebo::rendering::Camera::GetWindowId () const`

Get the ID of the window this camera is rendering into.

Returns

The ID of the window.

10.15.3.48 `bool gazebo::rendering::Camera::GetWorldPointOnPlane (int _x, int _y, const math::Plane & _plane, math::Vector3 & _result)`

Get point on a plane.

Parameters

in	<code>_x</code>	X coordinate in camera's viewport, in pixels
in	<code>_y</code>	Y coordinate in camera's viewport, in pixels
in	<code>_plane</code>	Plane on which to find the intersecting point
out	<code>_result</code>	Point on the plane

Returns

True if a valid point was found

10.15.3.49 `math::Pose gazebo::rendering::Camera::GetWorldPose ()`

Get the global pose of the camera.

Returns

Pose of the camera in the world coordinate frame

10.15.3.50 `math::Vector3 gazebo::rendering::Camera::GetWorldPosition () const`

Get the camera position in the world.

Returns

The world position of the camera

10.15.3.51 `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. 675)

10.15.3.52 `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.

10.15.3.53 `virtual void gazebo::rendering::Camera::Init () [virtual]`

Initialize the camera.

Reimplemented in `gazebo::rendering::GpuLaser` (p. 350), `gazebo::rendering::DepthCamera` (p. 275), and `gazebo::rendering::UserCamera` (p. 984).

10.15.3.54 `bool gazebo::rendering::Camera::IsAnimating () const`

Return true if the camera is moving due to an animation.

10.15.3.55 `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

10.15.3.56 `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

10.15.3.57 virtual void gazebo::rendering::Camera::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the camera with a set of parameters.

Parameters

in	<i>_sdf</i>	The SDF camera info
----	-------------	---------------------

Reimplemented in **gazebo::rendering::UserCamera** (p. 984).

10.15.3.58 virtual void gazebo::rendering::Camera::Load () [virtual]

Load the camera with default parameters.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 350), **gazebo::rendering::DepthCamera** (p. 276), and **gazebo::rendering::UserCamera** (p. 985).

10.15.3.59 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. 197)

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. 985).

10.15.3.60 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. 197)

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

10.15.3.61 virtual void gazebo::rendering::Camera::PostRender () [virtual]

Post render.

Called after the render signal.

Reimplemented in `gazebo::rendering::GpuLaser` (p. 350), `gazebo::rendering::DepthCamera` (p. 276), and `gazebo::rendering::UserCamera` (p. 985).

10.15.3.62 `void gazebo::rendering::Camera::ReadPixelBuffer ()` [protected]

Read image data from pixel buffer.

10.15.3.63 `void gazebo::rendering::Camera::Render ()`

Render the camera.

Called after the pre-render signal. This function will generate camera images

10.15.3.64 `virtual void gazebo::rendering::Camera::RenderImpl ()` [protected],[virtual]

Implementation of the render call.

10.15.3.65 `void gazebo::rendering::Camera::RotatePitch (math::Angle _angle)`

Rotate the camera around the pitch axis.

Parameters

<code>in</code>	<code>_angle</code>	Pitch amount
-----------------	---------------------	--------------

10.15.3.66 `void gazebo::rendering::Camera::RotateYaw (math::Angle _angle)`

Rotate the camera around the yaw axis.

Parameters

<code>in</code>	<code>_angle</code>	Rotation amount
-----------------	---------------------	-----------------

10.15.3.67 `bool gazebo::rendering::Camera::SaveFrame (const std::string & _filename)`

Save the last frame to disk.

Parameters

<code>in</code>	<code>_filename</code>	File in which to save a single frame
-----------------	------------------------	--------------------------------------

Returns

True if saving was successful

10.15.3.68 `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

10.15.3.69 `void gazebo::rendering::Camera::SetAspectRatio (float _ratio)`

Set the aspect ratio.

Parameters

in	<i>_ratio</i>	The aspect ratio (width / height) in pixels
----	---------------	---

10.15.3.70 `void gazebo::rendering::Camera::SetCaptureData (bool _value)`

Set whether to capture data.

Parameters

in	<i>_value</i>	Set to true to capture data into a memory buffer.
----	---------------	---

10.15.3.71 `void gazebo::rendering::Camera::SetCaptureDataOnce ()`

Capture data once and save to disk.

10.15.3.72 `void gazebo::rendering::Camera::SetClipDist (float _near, float _far)`

Set the clip distances.

Parameters

in	<code>_near</code>	Near clip distance in meters
in	<code>_far</code>	Far clip distance in meters

10.15.3.73 `void gazebo::rendering::Camera::SetHFOV (math::Angle _angle)`

Set the camera FOV (horizontal)

Parameters

in	<code>_radians</code>	Horizontal field of view
----	-----------------------	--------------------------

10.15.3.74 `void gazebo::rendering::Camera::SetImageHeight (unsigned int _h)`

Set the image height.

Parameters

in	<code>_h</code>	Image height
----	-----------------	--------------

10.15.3.75 `void gazebo::rendering::Camera::SetImageSize (unsigned int _w, unsigned int _h)`

Set the image size.

Parameters

in	<code>_w</code>	Image width
in	<code>_h</code>	Image height

10.15.3.76 `void gazebo::rendering::Camera::SetImageWidth (unsigned int _w)`

Set the image height.

Parameters

in	<code>_w</code>	Image width
----	-----------------	-------------

10.15.3.77 `void gazebo::rendering::Camera::SetName (const std::string & _name)`

Set the camera's name.

Parameters

in	<code>_name</code>	New name for the camera
----	--------------------	-------------------------

10.15.3.78 void gazebo::rendering::Camera::SetRenderRate (double *_hz*)

Set the render Hz rate.

Parameters

in	<i>_hz</i>	The Hz rate
----	------------	-------------

10.15.3.79 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. 986).

10.15.3.80 void gazebo::rendering::Camera::SetSaveFramePathname (const std::string & *_pathname*)

Set the save frame pathname.

Parameters

in	<i>_pathname</i>	Directory in which to store saved image frames
----	------------------	--

10.15.3.81 void gazebo::rendering::Camera::SetScene (ScenePtr *_scene*)

Set the scene this camera is viewing.

Parameters

in	<i>_scene</i>	Pointer to the scene
----	---------------	----------------------

10.15.3.82 void gazebo::rendering::Camera::SetSceneNode (Ogre::SceneNode * *_node*)

Set the camera's scene node.

Parameters

in	<i>_node</i>	The scene nodes to attach the camera to
----	--------------	---

10.15.3.83 void gazebo::rendering::Camera::SetWindowId (unsigned int *_windowId*)

10.15.3.84 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. 648) of the camera
----	--------------------	--

Reimplemented in **gazebo::rendering::UserCamera** (p. 987).

10.15.3.85 void **gazebo::rendering::Camera::SetWorldPosition** (const **math::Vector3** & `_pos`)

Set the world position.

Parameters

in	<code>_pos</code>	The new position of the camera
----	-------------------	--------------------------------

10.15.3.86 void **gazebo::rendering::Camera::SetWorldRotation** (const **math::Quaternion** & `_quat`)

Set the world orientation.

Parameters

in	<code>_quat</code>	The new orientation of the camera
----	--------------------	-----------------------------------

10.15.3.87 void **gazebo::rendering::Camera::ShowWireframe** (bool `_s`)

Set whether to view the world in wireframe.

Parameters

in	<code>_s</code>	Set to True to render objects as wireframe
----	-----------------	--

10.15.3.88 void **gazebo::rendering::Camera::ToggleShowWireframe** ()

Toggle whether to view the world in wireframe.

10.15.3.89 void **gazebo::rendering::Camera::TrackVisual** (const std::string & `_visualName`)

Set the camera to track a scene node.

Parameters

in	<code>_visualName</code>	Name of the visual to track
----	--------------------------	-----------------------------

10.15.3.90 bool **gazebo::rendering::Camera::TrackVisualImpl** (const std::string & `_visualName`) [protected]

Implementation of the **Camera::TrackVisual** (p. 202) call.

Parameters

in	<code>_visualName</code>	Name of the visual to track
----	--------------------------	-----------------------------

Returns

True if able to track the visual

10.15.3.91 `virtual bool gazebo::rendering::Camera::TrackVisualImpl (VisualPtr _visual)` [protected],[virtual]

Set the camera to track a scene node.

Parameters

in	<code>_visual</code>	The visual to track
----	----------------------	---------------------

Returns

True if able to track the visual

Reimplemented in **gazebo::rendering::UserCamera** (p. 987).

10.15.3.92 `void gazebo::rendering::Camera::Translate (const math::Vector3 & _direction)`

Translate the camera.

Parameters

in	<code>_direction</code>	The translation vector
----	-------------------------	------------------------

10.15.3.93 `virtual void gazebo::rendering::Camera::Update ()` [virtual]

Reimplemented in **gazebo::rendering::UserCamera** (p. 987).

10.15.4 Member Data Documentation

10.15.4.1 `Ogre::AnimationState* gazebo::rendering::Camera::animState` [protected]

Animation state, used to animate the camera.

10.15.4.2 `unsigned char* gazebo::rendering::Camera::bayerFrameBuffer` [protected]

Buffer for a bayer image frame.

10.15.4.3 `Ogre::Camera* gazebo::rendering::Camera::camera` [protected]

The OGRE camera.

10.15.4.4 `bool gazebo::rendering::Camera::captureData` [protected]

True to capture frames into an image buffer.

10.15.4.5 `bool gazebo::rendering::Camera::captureDataOnce` [protected]

True to capture a frame once and save to disk.

10.15.4.6 `std::vector<event::ConnectionPtr> gazebo::rendering::Camera::connections` [protected]

The camera's event connections.

10.15.4.7 `int gazebo::rendering::Camera::imageFormat` [protected]

Format for saving images.

10.15.4.8 `int gazebo::rendering::Camera::imageHeight` [protected]

Save image height.

10.15.4.9 `int gazebo::rendering::Camera::imageWidth` [protected]

Save image width.

10.15.4.10 `bool gazebo::rendering::Camera::initialized` [protected]

True if initialized.

10.15.4.11 `common::Time gazebo::rendering::Camera::lastRenderWallTime` [protected]

Time the last frame was rendered.

10.15.4.12 `std::string gazebo::rendering::Camera::name` [protected]

Name of the camera.

10.15.4.13 `bool gazebo::rendering::Camera::newData` [protected]

True if new data is available.

10.15.4.14 `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.

Referenced by `ConnectNewImageFrame()`, and `DisconnectNewImageFrame()`.

10.15.4.15 `boost::function<void()> gazebo::rendering::Camera::onAnimationComplete` [protected]

User callback for when an animation completes.

10.15.4.16 `Ogre::SceneNode*` `gazebo::rendering::Camera::pitchNode` [protected]

Scene (p. 728) nod that controls camera pitch.

10.15.4.17 `common::Time` `gazebo::rendering::Camera::prevAnimTime` [protected]

Previous time the camera animation was updated.

10.15.4.18 `Ogre::RenderTarget*` `gazebo::rendering::Camera::renderTarget` [protected]

Target that renders frames.

10.15.4.19 `Ogre::Texture*` `gazebo::rendering::Camera::renderTexture` [protected]

Texture that receives results from rendering.

10.15.4.20 `std::list<msgs::Request>` `gazebo::rendering::Camera::requests` [protected]

List of requests.

10.15.4.21 `unsigned int` `gazebo::rendering::Camera::saveCount` [protected]

Number of saved frames.

10.15.4.22 `unsigned char*` `gazebo::rendering::Camera::saveFrameBuffer` [protected]

10.15.4.23 `ScenePtr` `gazebo::rendering::Camera::scene` [protected]

Pointer to the scene.

10.15.4.24 `Ogre::SceneNode*` `gazebo::rendering::Camera::sceneNode` [protected]

Scene (p. 728) node that controls camera position.

10.15.4.25 `std::string` `gazebo::rendering::Camera::screenshotPath` [protected]

Path to saved screenshots.

10.15.4.26 `sdf::ElementPtr` `gazebo::rendering::Camera::sdf` [protected]

Camera (p. 179)'s SDF values.

10.15.4.27 `unsigned int` `gazebo::rendering::Camera::textureHeight` [protected]

Height of the render texture.

10.15.4.28 `unsigned int gazebo::rendering::Camera::textureWidth` `[protected]`

Width of the render texture.

10.15.4.29 `Ogre::Viewport* gazebo::rendering::Camera::viewport` `[protected]`

Viewport the ogre camera uses.

10.15.4.30 `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:

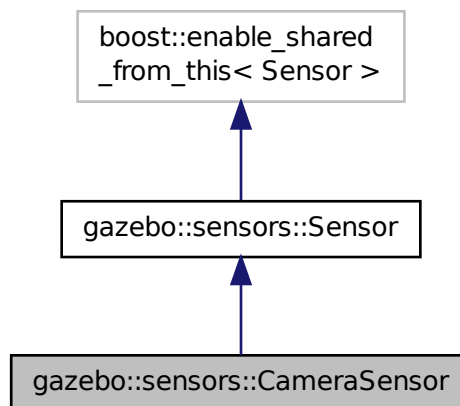
- **Camera.hh**

10.16 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. 179).*
- 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 void **UpdateImpl** (bool _force)
Update the sensor information.

Additional Inherited Members

10.16.1 Detailed Description

Basic camera sensor.

This sensor is used for simulating standard monocular cameras

10.16.2 Constructor & Destructor Documentation

10.16.2.1 gazebo::sensors::CameraSensor::CameraSensor ()

Constructor.

10.16.2.2 virtual gazebo::sensors::CameraSensor::~CameraSensor () [virtual]

Destructor.

10.16.3 Member Function Documentation

10.16.3.1 `virtual void gazebo::sensors::CameraSensor::Fini () [protected],[virtual]`

Finalize the camera.

Reimplemented from `gazebo::sensors::Sensor` (p. 755).

10.16.3.2 `rendering::CameraPtr gazebo::sensors::CameraSensor::GetCamera () const [inline]`

Returns a pointer to the `rendering::Camera` (p. 179).

Returns

The Pointer to the camera sensor.

10.16.3.3 `const unsigned char* gazebo::sensors::CameraSensor::GetImageData ()`

Gets the raw image data from the sensor.

Returns

The pointer to the image data array.

10.16.3.4 `unsigned int gazebo::sensors::CameraSensor::GetImageHeight () const`

Gets the height of the image in pixels.

Returns

The image height in pixels.

10.16.3.5 `unsigned int gazebo::sensors::CameraSensor::GetImageWidth () const`

Gets the width of the image in pixels.

Returns

The image width in pixels.

10.16.3.6 `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. 757).

10.16.3.7 `virtual void gazebo::sensors::CameraSensor::Init() [virtual]`

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 758).

10.16.3.8 `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. 758).

10.16.3.9 `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. 751) parameters
<code>in</code>	<code>_worldName</code>	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 759).

10.16.3.10 `virtual void gazebo::sensors::CameraSensor::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. 759).

10.16.3.11 `bool gazebo::sensors::CameraSensor::SaveFrame(const std::string & _filename)`

Saves the image to the disk.

Parameters

<code>in</code>	<code>_filename</code>	The name of the file to be saved.
-----------------	------------------------	-----------------------------------

Returns

True if successful, false if unsuccessful.

10.16.3.12 virtual void gazebo::sensors::CameraSensor::UpdateImpl (bool *_force*) [protected],[virtual]

Update the sensor information.

Parameters

in	<i>_force</i>	True if update is forced, false if not
----	---------------	--

Reimplemented from **gazebo::sensors::Sensor** (p. 760).

The documentation for this class was generated from the following file:

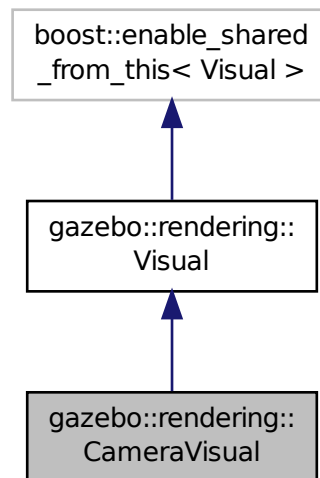
- **CameraSensor.hh**

10.17 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. 1034).*

Additional Inherited Members

10.17.1 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.

10.17.2 Constructor & Destructor Documentation

10.17.2.1 gazebo::rendering::CameraVisual::CameraVisual (const std::string & *_name*, VisualPtr *_vis*)

Constructor.

Parameters

in	<i>_name</i>	Name of the Visual (p. 1034)
in	<i>_vis</i>	Pointer to the parent Visual (p. 1034)

10.17.2.2 virtual gazebo::rendering::CameraVisual::~CameraVisual () [virtual]

Destructor.

10.17.3 Member Function Documentation

10.17.3.1 void gazebo::rendering::CameraVisual::Load (unsigned int *_width*, unsigned int *_height*)

Load the **Visual** (p. 1034).

Parameters

in	<i>_width</i>	Width of the Camera (p. 179) image
in	<i>_height</i>	Height of the Camera (p. 179) image

The documentation for this class was generated from the following file:

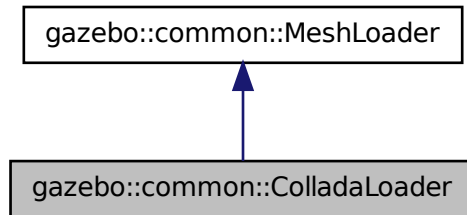
- **CameraVisual.hh**

10.18 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.

10.18.1 Detailed Description

Class used to load Collada mesh files.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 gazebo::common::ColladaLoader::ColladaLoader ()

Constructor.

10.18.2.2 virtual gazebo::common::ColladaLoader::~~ColladaLoader () [virtual]

Destructor.

10.18.3 Member Function Documentation

10.18.3.1 virtual Mesh* gazebo::common::ColladaLoader::Load (const std::string &_filename) [virtual]

Load a mesh.

Parameters

in	_filename	Collada file to load
----	-----------	----------------------

Returns

Pointer to a new **Mesh** (p. 519)

Implements **gazebo::common::MeshLoader** (p. 528).

The documentation for this class was generated from the following file:

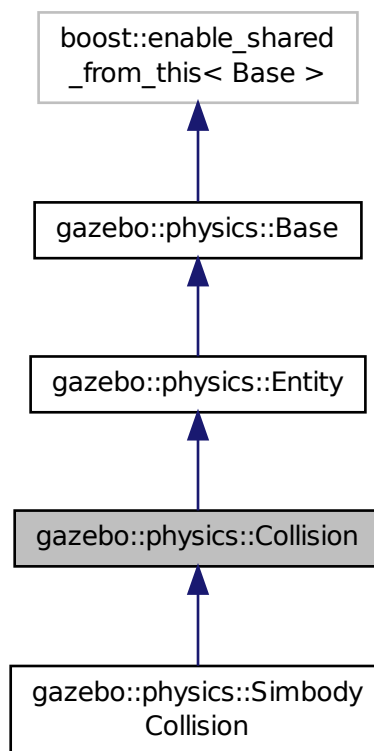
- **ColladaLoader.hh**

10.19 gazebo::physics::Collision Class Reference

Base (p. 153) 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**(1.10)
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**(1.10)
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 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**(1.10)
Turn contact recording on or off.
- void **SetLaserRetro** (float _retro)
Set the laser retro reflectiveness.
- virtual void **SetMaxContacts** (double _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. 775).*

Additional Inherited Members

10.19.1 Detailed Description

Base (p. 153) class for all collision entities.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 gazebo::physics::Collision::Collision (LinkPtr *_link*) [explicit]

Constructor.

Parameters

in	<i>_link</i>	Link (p. 455) that contains this collision object.
----	--------------	--

10.19.2.2 virtual gazebo::physics::Collision::~~Collision () [virtual]

Destructor.

10.19.3 Member Function Documentation

10.19.3.1 void gazebo::physics::Collision::AddContact (const Contact & *_contact*)

Add an occurrence of a contact to this collision.

Parameters

in	<i>_contact</i>	The contact which was detected by a collision engine.
----	-----------------	---

10.19.3.2 void gazebo::physics::Collision::FillMsg (msgs::Collision & *_msg*)

Fill a collision message.

Parameters

out	<i>_msg</i>	The message to fill with this collision's data.
-----	-------------	---

10.19.3.3 virtual void gazebo::physics::Collision::Fini () [virtual]

Finalize the collision.

Reimplemented from **gazebo::physics::Entity** (p. 297).

10.19.3.4 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. 297).

Implemented in **gazebo::physics::SimbodyCollision** (p. 787).

10.19.3.5 `bool gazebo::physics::Collision::GetContactsEnabled () const`

Return true of contacts are on.

Returns

True of contact are on.

10.19.3.6 `float gazebo::physics::Collision::GetLaserRetro () const`

Get the laser retro reflectiveness.

Returns

The laser retro value.

10.19.3.7 `LinkPtr gazebo::physics::Collision::GetLink () const`

Get the link this collision belongs to.

Returns

The parent **Link** (p. 455).

10.19.3.8 `virtual int gazebo::physics::Collision::GetMaxContacts () [virtual]`

returns number of contacts allowed for this collision.

This overrides global value (in **PhysicsEngine** (p. 620)) if specified.

Returns

max num contacts allowed for this collision.

10.19.3.9 `ModelPtr gazebo::physics::Collision::GetModel () const`

Get the model this collision belongs to.

Returns

The parent model.

10.19.3.10 `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. 298).

10.19.3.11 `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. 299).

10.19.3.12 `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. 299).

10.19.3.13 `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. 299).

10.19.3.14 `ShapePtr gazebo::physics::Collision::GetShape () const`

Get the collision shape.

Returns

The collision shape.

10.19.3.15 `unsigned int gazebo::physics::Collision::GetShapeType ()`

Get the shape type.

Returns

The shape type.

See Also

`EntityType` (p. 156)

10.19.3.16 CollisionState gazebo::physics::Collision::GetState ()

Get the collision state.

Returns

The collision state.

10.19.3.17 SurfaceParamsPtr gazebo::physics::Collision::GetSurface () const [inline]

Get the surface parameters.

Returns

The surface parameters.

10.19.3.18 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. 299).

10.19.3.19 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. 300).

10.19.3.20 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. 300).

10.19.3.21 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. 300).

10.19.3.22 `virtual void gazebo::physics::Collision::Init () [virtual]`

Initialize the collision.

Reimplemented from **gazebo::physics::Base** (p. 160).

10.19.3.23 `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.

10.19.3.24 `virtual void gazebo::physics::Collision::Load (sdf::ElementPtr _sdf) [virtual]`

Load the collision.

Parameters

<code>in</code>	<code>_sdf</code>	SDF to load from.
-----------------	-------------------	-------------------

Reimplemented from **gazebo::physics::Entity** (p. 301).

Reimplemented in **gazebo::physics::SimbodyCollision** (p. 787).

10.19.3.25 `void gazebo::physics::Collision::ProcessMsg (const msgs::Collision & _msg)`

Update parameters from a message.

Parameters

<code>in</code>	<code>_msg</code>	Message to update from.
-----------------	-------------------	-------------------------

10.19.3.26 `virtual void gazebo::physics::Collision::SetCategoryBits (unsigned int _bits) [pure virtual]`

Set the category bits, used during collision detection.

Parameters

<code>in</code>	<code>_bits</code>	The bits to set.
-----------------	--------------------	------------------

Implemented in **gazebo::physics::SimbodyCollision** (p. 788).

10.19.3.27 `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::SimbodyCollision** (p. 788).

10.19.3.28 void gazebo::physics::Collision::SetCollision (bool *_placeable*)

Set the encapsulated collision object.

Parameters

in	<i>_placeable</i>	True to make the object movable.
----	-------------------	----------------------------------

10.19.3.29 void gazebo::physics::Collision::SetContactsEnabled (bool *_enable*)

Turn contact recording on or off.

Parameters

in	<i>_enable</i>	True to enable collision contacts.
----	----------------	------------------------------------

10.19.3.30 void gazebo::physics::Collision::SetLaserRetro (float *_retro*)

Set the laser retro reflectiveness.

Parameters

in	<i>_retro</i>	The laser retro value.
----	---------------	------------------------

10.19.3.31 virtual void gazebo::physics::Collision::SetMaxContacts (double *_maxContacts*) [virtual]

Number of contacts allowed for this collision.

This overrides global value (in **PhysicsEngine** (p. 620)) if specified.

Parameters

in	<i>_maxContacts</i>	max num contacts allowed for this collision.
----	---------------------	--

10.19.3.32 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.
----	---------------	--------------------------------

10.19.3.33 void gazebo::physics::Collision::SetShape (ShapePtr *_shape*)

Set the shape for this collision.

Parameters

in	<i>_shape</i>	The shape for this collision object.
----	---------------	--------------------------------------

10.19.3.34 void gazebo::physics::Collision::SetState (const CollisionState & *_state*)

Set the current collision state.

Parameters

in	<i>The</i>	collision state.
----	------------	------------------

10.19.3.35 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. 303).

10.19.4 Member Data Documentation

10.19.4.1 LinkPtr gazebo::physics::Collision::link [protected]

The link this collision belongs to.

10.19.4.2 bool gazebo::physics::Collision::placeable [protected]

Flag for placeable.

10.19.4.3 ShapePtr gazebo::physics::Collision::shape [protected]

Pointer to **physics::Shape** (p. 775).

The documentation for this class was generated from the following file:

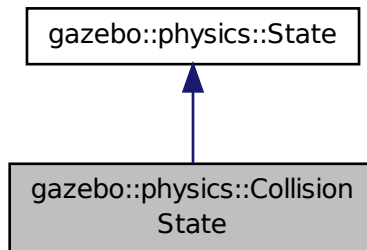
- **Collision.hh**

10.20 gazebo::physics::CollisionState Class Reference

Store state information of a **physics::Collision** (p. 213) 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. 213) 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

10.20.1 Detailed Description

Store state information of a **physics::Collision** (p. 213) object.

This class captures the entire state of a **Collision** (p. 213) at one specific time during a simulation run.

State (p. 910) of a **Collision** (p. 213) is its Pose.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 gazebo::physics::CollisionState::CollisionState ()

Default constructor.

10.20.2.2 gazebo::physics::CollisionState::CollisionState (const CollisionPtr *collision*) [explicit]

Constructor.

Build a **CollisionState** (p. 222) from an existing **Collision** (p. 213).

Parameters

in	<i>_model</i>	Pointer to the Link (p. 455) from which to gather state info.
----	---------------	--

10.20.2.3 gazebo::physics::CollisionState::CollisionState (const sdf::ElementPtr *sdf*) [explicit]

Constructor.

Build a **CollisionState** (p. 222) from SDF data

Parameters

in	<i>_sdf</i>	SDF data to load a collision state from.
----	-------------	--

10.20.2.4 virtual gazebo::physics::CollisionState::~~CollisionState () [virtual]

Destructor.

10.20.3 Member Function Documentation

10.20.3.1 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.
-----	-------------	--------------------------

10.20.3.2 `const math::Pose& gazebo::physics::CollisionState::GetPose () const`

Get the **Collision** (p. 213) pose.

Returns

The pose of the **CollisionState** (p. 222)

10.20.3.3 `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.

10.20.3.4 `virtual void gazebo::physics::CollisionState::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Load **CollisionState** (p. 222) 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. 913).

10.20.3.5 `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.

10.20.3.6 `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.

10.20.3.7 CollisionState& gazebo::physics::CollisionState::operator= (const CollisionState & _state)

Assignment operator.

Parameters

<i>in</i>	<i>_state</i>	State (p. 910) value
-----------	---------------	-----------------------------

Returns

Reference to this

10.20.4 Friends And Related Function Documentation**10.20.4.1 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. 213) state to output

Returns

the stream

The documentation for this class was generated from the following file:

- **CollisionState.hh**

10.21 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.

10.21.1 Detailed Description

Defines a color.

10.21.2 Member Typedef Documentation

10.21.2.1 typedef unsigned int gazebo::common::Color::ABGR

10.21.2.2 typedef unsigned int gazebo::common::Color::ARGB

10.21.2.3 typedef unsigned int gazebo::common::Color::BGRA

10.21.2.4 typedef unsigned int gazebo::common::Color::RGBA

10.21.3 Constructor & Destructor Documentation

10.21.3.1 gazebo::common::Color::Color ()

Constructor.

10.21.3.2 gazebo::common::Color::Color (float *_r*, float *_g*, float *_b*, float *_a* = 1.0)

Constructor.

Parameters

<i>in</i>	<i>_r</i>	Red value (range 0 to 1)
<i>in</i>	<i>_g</i>	Green value (range 0 to 1)
<i>in</i>	<i>_b</i>	Blue value (range 0 to 1)
<i>in</i>	<i>_a</i>	Alpha value (0=transparent, 1=opaque)

10.21.3.3 gazebo::common::Color::Color (const **Color** & *_clr*)

Copy Constructor.

Parameters

<i>in</i>	<i>_clr</i>	Color (p. 226) to copy
-----------	-------------	-------------------------------

10.21.3.4 `virtual gazebo::common::Color::~~Color () [virtual]`

Destructor.

10.21.4 Member Function Documentation

10.21.4.1 `ABGR gazebo::common::Color::GetAsABGR () const`

Get as uint32 ABGR packed value.

Returns

the color

10.21.4.2 `ARGB gazebo::common::Color::GetAsARGB () const`

Get as uint32 ARGB packed value.

Returns

the color

10.21.4.3 `BGRA gazebo::common::Color::GetAsBGRA () const`

Get as uint32 BGRA packed value.

Returns

the color

10.21.4.4 `math::Vector3 gazebo::common::Color::GetAsHSV () const`

Get the color in HSV colorspace.

Returns

HSV values in a `math::Vector3` (p. 1004) format

10.21.4.5 `RGBA gazebo::common::Color::GetAsRGBA () const`

Get as uint32 RGBA packed value.

Returns

the color

10.21.4.6 `math::Vector3 gazebo::common::Color::GetAsYUV () const`

Get the color in YUV colorspace.

Returns

the YUV color

10.21.4.7 `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`

10.21.4.8 `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

10.21.4.9 `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

10.21.4.10 `const Color& gazebo::common::Color::operator*=(const Color & _pt)`

Multiplication equal operator.

Parameters

in	_pt	The color to multiply by
----	-----	--------------------------

Returns

The resulting color

10.21.4.11 **Color** gazebo::common::Color::operator+ (const **Color** & _pt) const

Addition operator (this + _pt)

Parameters

in	_pt	Color (p. 226) to add
----	-----	------------------------------

Returns

The resulting color

10.21.4.12 **Color** gazebo::common::Color::operator+ (const float & _v) const

Add _v to all color components.

Parameters

in	_v	Value to add to each color component
----	----	--------------------------------------

Returns

The resulting color

10.21.4.13 const **Color**& gazebo::common::Color::operator+= (const **Color** & _pt)

Addition equal operator.

Parameters

in	_pt	Color (p. 226) to add
----	-----	------------------------------

Returns

The resulting color

10.21.4.14 **Color** gazebo::common::Color::operator- (const **Color** & _pt) const

Subtraction operator.

Parameters

in	_pt	The color to subtract
----	-----	-----------------------

Returns

The resulting color

10.21.4.15 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

10.21.4.16 const Color& gazebo::common::Color::operator-= (const Color & _pt)

Subtraction equal operator.

Parameters

in	_pt	Color (p. 226) to subtract
----	-----	-----------------------------------

Returns

The resulting color

10.21.4.17 const Color gazebo::common::Color::operator/ (const Color & _pt) const

Division operator.

Parameters

in	_pt	Color (p. 226) to divide by
----	-----	------------------------------------

Returns

The resulting color

10.21.4.18 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

10.21.4.19 `const Color& gazebo::common::Color::operator/= (const Color & _pt)`

Division equal operator.

Parameters

in	_pt	Color (p. 226) to divide by
----	-----	------------------------------------

Returns

The resulting color

10.21.4.20 `Color& gazebo::common::Color::operator= (const Color & _pt)`

Equal operator.

Parameters

in	_pt	Color (p. 226) to copy
----	-----	-------------------------------

Returns

Reference to this color

10.21.4.21 `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

10.21.4.22 `float gazebo::common::Color::operator[] (unsigned int _index)`

Array index operator.

Parameters

in	_index	Color (p. 226) component index(0=red, 1=green, 2=blue)
----	--------	---

Returns

r, g, b, or a when _index is 0, 1, 2 or 3

10.21.4.23 void gazebo::common::Color::Reset ()

Reset the color to default values.

10.21.4.24 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	_r	Red value (range 0 to 1)
in	_g	Green value (range 0 to 1)
in	_b	Blue value (range 0 to 1)
in	_a	Alpha value (0=transparent, 1=opaque)

10.21.4.25 void gazebo::common::Color::SetFromABGR (const ABGR _v)

Set from uint32 ABGR packed value.

Parameters

in	_v	the new color
----	----	---------------

10.21.4.26 void gazebo::common::Color::SetFromARGB (const ARGB _v)

Set from uint32 ARGB packed value.

Parameters

in	_v	the new color
----	----	---------------

10.21.4.27 void gazebo::common::Color::SetFromBGRA (const BGRA _v)

Set from uint32 BGRA packed value.

Parameters

in	_v	the new color
----	----	---------------

10.21.4.28 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)

10.21.4.29 void gazebo::common::Color::SetFromRGBA (const RGBA *_v*)

Set from uint32 RGBA packed value.

Parameters

in	<i>_v</i>	the new color
----	-----------	---------------

10.21.4.30 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

10.21.5 Friends And Related Function Documentation

10.21.5.1 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

10.21.5.2 std::istream& operator>> (std::istream & *_in*, Color & *_pt*) [friend]

Stream insertion operator.

Parameters

in	<i>_in</i>	the input stream
in	<i>_pt</i>	

10.21.6 Member Data Documentation

10.21.6.1 float gazebo::common::Color::a

10.21.6.2 float gazebo::common::Color::b

10.21.6.3 const Color gazebo::common::Color::Black [static]

(0, 0, 0)

10.21.6.4 const Color gazebo::common::Color::Blue [static]

(0, 0, 1)

10.21.6.5 float gazebo::common::Color::g

10.21.6.6 const Color gazebo::common::Color::Green [static]

(0, 1, 0)

10.21.6.7 const Color gazebo::common::Color::Purple [static]

(1, 0, 1)

10.21.6.8 float gazebo::common::Color::r

10.21.6.9 const Color gazebo::common::Color::Red [static]

(1, 0, 0)

10.21.6.10 const Color gazebo::common::Color::White [static]

(1, 1, 1)

10.21.6.11 const Color gazebo::common::Color::Yellow [static]

(1, 1, 0)

The documentation for this class was generated from the following file:

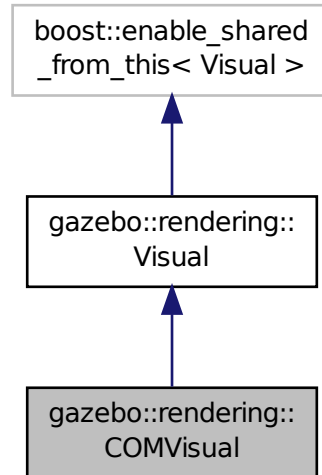
- **Color.hh**

10.22 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. 1034) from an SDF pointer.*
- virtual void **Load** (ConstLinkPtr &_msg)
Load from a message.

Additional Inherited Members

10.22.1 Detailed Description

Basic Center of Mass visualization.

10.22.2 Constructor & Destructor Documentation

10.22.2.1 gazebo::rendering::COMVisual::COMVisual (const std::string & _name, VisualPtr _vis)

Constructor.

Parameters

in	<code>_name</code>	Name of the Visual (p. 1034)
in	<code>_vis</code>	Parent Visual (p. 1034)

10.22.2.2 virtual gazebo::rendering::COMVisual::~~COMVisual () [virtual]

Destructor.

10.22.3 Member Function Documentation

10.22.3.1 virtual void gazebo::rendering::COMVisual::Load (sdf::ElementPtr *elem*) [virtual]

Load the **Visual** (p. 1034) from an SDF pointer.

Parameters

in	<code>_elem</code>	SDF Element pointer
----	--------------------	---------------------

10.22.3.2 virtual void gazebo::rendering::COMVisual::Load (ConstLinkPtr & *msg*) [virtual]

Load from a message.

Parameters

in	<code>_msg</code>	Pointer to the message
----	-------------------	------------------------

The documentation for this class was generated from the following file:

- **COMVisual.hh**

10.23 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.

10.23.1 Detailed Description

A class that encapsulates a connection.

10.23.2 Constructor & Destructor Documentation

10.23.2.1 gazebo::event::Connection::Connection () [inline]

Constructor.

10.23.2.2 gazebo::event::Connection::Connection (Event * _e, int _i)

Constructor.

Parameters

in	<code>_e</code>	Event (p. 305) pointer to connect with
in	<code>_i</code>	Unique id

10.23.2.3 gazebo::event::Connection::~~Connection ()

Destructor.

10.23.3 Member Function Documentation

10.23.3.1 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:

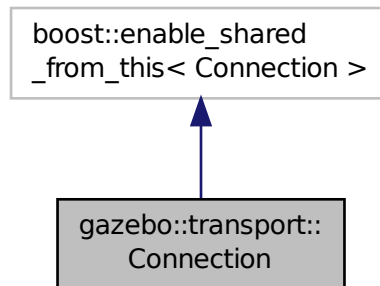
- **Event.hh**

10.24 gazebo::transport::Connection Class Reference

Single TCP/IP connection manager.

```
#include <transport/transport.hh>
```


Inheritance diagram for gazebo::transport::Connection:



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.*

10.24.1 Detailed Description

Single TCP/IP connection manager.

10.24.2 Member Typedef Documentation

10.24.2.1 typedef boost::function<void(const ConnectionPtr&)> gazebo::transport::Connection::AcceptCallback

The signature of a connection accept callback.

10.24.2.2 typedef boost::function<void(const std::string &_data)> gazebo::transport::Connection::ReadCallback

The signature of a connection read callback.

10.24.3 Constructor & Destructor Documentation

10.24.3.1 gazebo::transport::Connection::Connection ()

Constructor.

10.24.3.2 virtual gazebo::transport::Connection::~~Connection () [virtual]

Destructor.

10.24.4 Member Function Documentation

10.24.4.1 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, HEADER_LENGTH, and IsOpen().

10.24.4.2 void gazebo::transport::Connection::Cancel ()

Cancel all async operations on an open socket.

10.24.4.3 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

10.24.4.4 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::event::EventT< T >::Connect().

10.24.4.5 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. 243)
----	--------------------	---

References gazebo::event::EventT< T >::Disconnect().

10.24.4.6 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
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.

10.24.4.7 void gazebo::transport::Connection::EnqueueMsg (const std::string & *_buffer*, 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

10.24.4.8 unsigned int gazebo::transport::Connection::GetId () const

Get the ID of the connection.

Returns

The connection's unique ID.

10.24.4.9 `std::string gazebo::transport::Connection::GetIPWhiteList () const`

Get the IP white list, from GAZEBO_IP_WHITE_LIST environment variable.

Returns

GAZEBO_IP_WHITE_LIST

10.24.4.10 `std::string gazebo::transport::Connection::GetLocalAddress () const`

Get the local address of this connection.

Returns

The local address

10.24.4.11 `static std::string gazebo::transport::Connection::GetLocalHostname () [static]`

Get the local hostname.

Returns

The local hostname

10.24.4.12 `unsigned int gazebo::transport::Connection::GetLocalPort () const`

Get the port of this connection.

Returns

The local port

10.24.4.13 `std::string gazebo::transport::Connection::GetLocalURI () const`

Get the local URI.

Returns

The local URI

10.24.4.14 `std::string gazebo::transport::Connection::GetRemoteAddress () const`

Get the remote address.

Returns

The remote address

10.24.4.15 `std::string gazebo::transport::Connection::GetRemoteHostname () const`

Get the remote hostname.

Returns

The remote hostname

10.24.4.16 `unsigned int gazebo::transport::Connection::GetRemotePort () const`

Get the remote port number.

Returns

The remote port

10.24.4.17 `std::string gazebo::transport::Connection::GetRemoteURI () const`

Get the remote URI.

Returns

The remote URI

10.24.4.18 `bool gazebo::transport::Connection::IsOpen () const`

Is the connection open?

Returns

true if the connection is open; false otherwise

Referenced by AsyncRead().

10.24.4.19 `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

10.24.4.20 `void gazebo::transport::Connection::ProcessWriteQueue (bool _blocking = false)`

Handle on-write callbacks.

10.24.4.21 `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

10.24.4.22 `void gazebo::transport::Connection::Shutdown ()`

Shutdown the socket.

10.24.4.23 `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
-----------------	-------------------------	---

10.24.4.24 `void gazebo::transport::Connection::StopRead ()`

Stop the read loop.

10.24.4.25 `static bool gazebo::transport::Connection::ValidateIP (const std::string & _ip) [static]`

Return true if the `_ip` is a valid.

Parameters

<code>in</code>	<code><i>_ip</i></code>	Dotted quad to validate.
-----------------	-------------------------	--------------------------

Returns

True if the `_ip` is a valid.

The documentation for this class was generated from the following file:

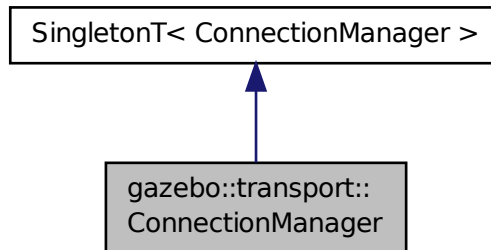
- **Connection.hh**

10.25 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)
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

10.25.1 Detailed Description

Manager of connections.

10.25.2 Member Function Documentation

10.25.2.1 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

10.25.2.2 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)

10.25.2.3 void gazebo::transport::ConnectionManager::Fini ()

Finalize the connection manager.

10.25.2.4 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
-----	--------------------	--

10.25.2.5 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
-----	--------------------	---------------------------------------

10.25.2.6 bool gazebo::transport::ConnectionManager::Init (const std::string & *_masterHost*, unsigned int *_masterPort*)

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

Returns

true if initialization succeeded, false otherwise

10.25.2.7 bool gazebo::transport::ConnectionManager::IsRunning () const

Is the manager running?

Returns

true if running, false otherwise

10.25.2.8 void gazebo::transport::ConnectionManager::RegisterTopicNamespace (const std::string & *_name*)

Register a new topic namespace.

Parameters

in	<i>_name</i>	The name of the topic namespace to be registered
----	--------------	--

10.25.2.9 void gazebo::transport::ConnectionManager::RemoveConnection (ConnectionPtr & *_conn*)

Remove a connection from the manager.

Parameters

in	<i>_conn</i>	The connection to be removed
----	--------------	------------------------------

10.25.2.10 void gazebo::transport::ConnectionManager::Run ()

Run the connection manager loop.

Does not return until stopped.

10.25.2.11 void gazebo::transport::ConnectionManager::Stop ()

Stop the connection manager.

10.25.2.12 void gazebo::transport::ConnectionManager::Subscribe (const std::string & *_topic*, const std::string & *_msgType*, bool *_latching*)

Subscribe to a topic.

Parameters

in	<i>_topic</i>	The topic to subscribe to
in	<i>_msgType</i>	The type of the topic
in	<i>_latching</i>	If true, latch the latest incoming message; otherwise don't

10.25.2.13 void gazebo::transport::ConnectionManager::TriggerUpdate ()

Inform the connection manager that it needs an update.

10.25.2.14 void gazebo::transport::ConnectionManager::Unadvertise (const std::string & *_topic*)

Unadvertise a topic.

Parameters

in	<i>_topic</i>	The topic to unadvertise
----	---------------	--------------------------

10.25.2.15 void gazebo::transport::ConnectionManager::Unsubscribe (const msgs::Subscribe & *_sub*)

Unsubscribe from a topic.

Parameters

in	<i>_sub</i>	A subscription object
----	-------------	-----------------------

10.25.2.16 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

10.25.3 Member Data Documentation

10.25.3.1 `std::vector<event::ConnectionPtr> gazebo::transport::ConnectionManager::eventConnections` [protected]

The documentation for this class was generated from the following file:

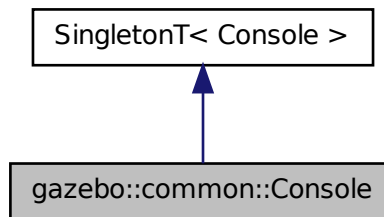
- **ConnectionManager.hh**

10.26 gazebo::common::Console Class Reference

Message, error, warning functionality.

```
#include <common/commom.hh>
```

Inheritance diagram for gazebo::common::Console:



Public Member Functions

- `std::ostream & ColorErr (const std::string &_lbl, const std::string &_file, unsigned int _line, int _color)`
Use this to output an error to the terminal.
- `std::ostream & ColorMsg (const std::string &_lbl, int _color)`
Use this to output a colored message to the terminal.
- `bool GetQuiet () const`
Get whether quiet output is set.
- `void Init (const std::string &_logFilename)`
Load the message parameters.
- `bool IsInitialized () const`
Return true if Init has been called.
- `std::ofstream & Log ()`
Use this to output a colored message to the terminal.
- `void SetQuiet (bool _q)`
Set quiet output.

Additional Inherited Members

10.26.1 Detailed Description

Message, error, warning functionality.

The documentation for this class was generated from the following file:

- **Console.hh**

10.27 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. 1070) in which the contact occurred.

- **JointWrench wrench [32]**

Array of forces for the contact.

10.27.1 Detailed Description

A contact between two collisions.

Each contact can consist of a number of contact points

10.27.2 Constructor & Destructor Documentation

10.27.2.1 gazebo::physics::Contact::Contact ()

Constructor.

10.27.2.2 gazebo::physics::Contact::Contact (const Contact & _contact)

Copy constructor.

Parameters

in	_contact	Contact (p. 253) to copy.
----	----------	----------------------------------

10.27.2.3 virtual gazebo::physics::Contact::~~Contact () [virtual]

Destructor.

10.27.3 Member Function Documentation

10.27.3.1 std::string gazebo::physics::Contact::DebugString () const

Produce a debug string.

Returns

A string that contains the values of the contact.

10.27.3.2 void gazebo::physics::Contact::FillMsg (msgs::Contact & _msg) const

Populate a msgs::Contact with data from this.

Parameters

out	<code>_msg</code>	Contact (p. 253) message the will hold the data.
-----	-------------------	---

10.27.3.3 **Contact&** gazebo::physics::Contact::operator= (const **Contact** & *.contact*)

Operator =.

Parameters

in	<code>_contact</code>	Contact (p. 253) to copy.
----	-----------------------	----------------------------------

Returns

Reference to this contact

10.27.3.4 **Contact&** gazebo::physics::Contact::operator= (const msgs::Contact & *.contact*)

Operator =.

Parameters

in	<code>_contact</code>	msgs::Contact to copy.
----	-----------------------	------------------------

Returns

Reference to this contact

10.27.3.5 void gazebo::physics::Contact::Reset ()

Reset to default values.

10.27.4 Member Data Documentation

10.27.4.1 **Collision*** gazebo::physics::Contact::collision1

Pointer to the first collision object.

10.27.4.2 **Collision*** gazebo::physics::Contact::collision2

Pointer to the second collision object.

10.27.4.3 int gazebo::physics::Contact::count

Length of all the arrays.

10.27.4.4 `double gazebo::physics::Contact::depths[32]`

Array of contact depths.

10.27.4.5 `math::Vector3 gazebo::physics::Contact::normals[32]`

Array of force normals.

10.27.4.6 `math::Vector3 gazebo::physics::Contact::positions[32]`

Array of force positions.

10.27.4.7 `common::Time gazebo::physics::Contact::time`

Time at which the contact occurred.

10.27.4.8 `WorldPtr gazebo::physics::Contact::world`

World (p. 1070) in which the contact occurred.

10.27.4.9 `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**

10.28 `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.
- void **Init** (`WorldPtr` _world)
*Initialize the **ContactManager** (p. 256).*
- `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 **ResetCount** ()
Set the contact count to zero.

10.28.1 Detailed Description

Aggregates all the contact information generated by the collision detection engine.

10.28.2 Constructor & Destructor Documentation

10.28.2.1 gazebo::physics::ContactManager::ContactManager ()

Constructor.

10.28.2.2 virtual gazebo::physics::ContactManager::~~ContactManager () [virtual]

Destructor.

10.28.3 Member Function Documentation

10.28.3.1 void gazebo::physics::ContactManager::Clear ()

Clear all stored contacts.

10.28.3.2 `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.

10.28.3.3 `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.

10.28.3.4 `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.

10.28.3.5 `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. 258).

Parameters

<i>in</i>	<i>_index</i>	Index of the Contact (p. 253) to return.
-----------	---------------	---

Returns

Pointer to a contact, NULL If index is invalid.

10.28.3.6 `unsigned int gazebo::physics::ContactManager::GetContactCount () const`

Return the number of valid contacts.

10.28.3.7 `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. 258)

Returns

Vector of contact pointers.

10.28.3.8 void gazebo::physics::ContactManager::Init (WorldPtr *_world*)

Initialize the **ContactManager** (p. 256).

This is required in order to publish contact messages via the **ContactManager::PublishContacts** (p. 259) method.

Parameters

<i>in</i>	<i>_world</i>	Pointer to the world that is initializing the contact manager.
-----------	---------------	--

10.28.3.9 **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.

10.28.3.10 void gazebo::physics::ContactManager::PublishContacts ()

Publish all contacts in a msgs::Contacts message.

10.28.3.11 void gazebo::physics::ContactManager::ResetCount ()

Set the contact count to zero.

The documentation for this class was generated from the following file:

- **ContactManager.hh**

10.29 gazebo::physics::ContactPublisher Class Reference

A custom contact publisher created for each contact filter in the **Contact** (p. 253) 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. 253) message publisher.

10.29.1 Detailed Description

A custom contact publisher created for each contact filter in the **Contact** (p. 253) Manager.

10.29.2 Member Data Documentation

10.29.2.1 `std::vector<std::string> gazebo::physics::ContactPublisher::collisionNames`

10.29.2.2 `boost::unordered_set<Collision *> gazebo::physics::ContactPublisher::collisions`

Pointers of collisions monitored by contact manager for contacts.

10.29.2.3 `std::vector<Contact *> gazebo::physics::ContactPublisher::contacts`

A list of contacts associated to the collisions.

10.29.2.4 `transport::PublisherPtr gazebo::physics::ContactPublisher::publisher`

Contact (p. 253) message publisher.

The documentation for this class was generated from the following file:

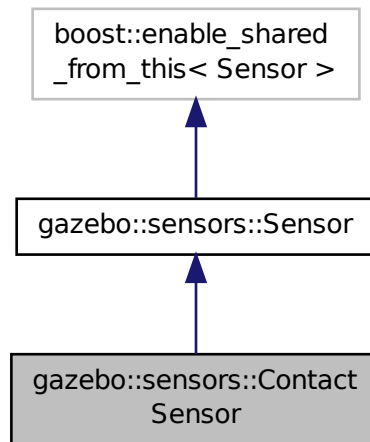
- **ContactManager.hh**

10.30 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. 260).*
- 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 void **UpdateImpl** (bool _force)
Update the sensor information.

Additional Inherited Members

10.30.1 Detailed Description

Contact sensor.

This sensor detects and reports contacts between objects

10.30.2 Constructor & Destructor Documentation

10.30.2.1 gazebo::sensors::ContactSensor::ContactSensor ()

Constructor.

10.30.2.2 virtual gazebo::sensors::ContactSensor::~~ContactSensor () [virtual]

Destructor.

10.30.3 Member Function Documentation

10.30.3.1 virtual void gazebo::sensors::ContactSensor::Fini () [protected],[virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 755).

10.30.3.2 unsigned int gazebo::sensors::ContactSensor::GetCollisionContactCount (const std::string & _collisionName) const

Return the number of contacts for an observed collision.

Parameters

in	<code>_collisionName</code>	The name of the observed collision.
----	-----------------------------	-------------------------------------

Returns

The collision contact count.

10.30.3.3 unsigned int gazebo::sensors::ContactSensor::GetCollisionCount () const

Get the number of collisions that the sensor is observing.

Returns

Number of collisions.

10.30.3.4 std::string gazebo::sensors::ContactSensor::GetCollisionName (unsigned int *_index*) const

Get a collision name at index *_index*.

Parameters

<i>in</i>	<i>_index</i>	Index of collision in collection of collisions.
-----------	---------------	---

Returns

name of collision.

10.30.3.5 msgs::Contacts gazebo::sensors::ContactSensor::GetContacts () const

Get all the contacts for the **ContactSensor** (p. 260).

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. 260) 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. 260) SDF. All collision pairs between **ContactSensor** (p. 260) <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.

10.30.3.6 `std::map<std::string, physics::Contact> gazebo::sensors::ContactSensor::GetContacts (const std::string & _collisionName)`

Gets contacts of a collision.

Parameters

in	<code>_collisionName</code>	Name of collision
----	-----------------------------	-------------------

Returns

Container of contacts

10.30.3.7 `virtual void gazebo::sensors::ContactSensor::Init () [virtual]`

Initialize the sensor.

Reimplemented from `gazebo::sensors::Sensor` (p. 758).

10.30.3.8 `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. 758).

10.30.3.9 `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 <code>Sensor</code> (p. 751) parameters
in	<code>_worldName</code>	Name of world to load from

Reimplemented from `gazebo::sensors::Sensor` (p. 759).

10.30.3.10 `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. 759).

10.30.3.11 virtual void gazebo::sensors::ContactSensor::UpdateImpl (bool *_force*) [protected], [virtual]

Update the sensor information.

Parameters

in	<i>_force</i>	True if update is forced, false if not.
----	---------------	---

Reimplemented from **gazebo::sensors::Sensor** (p. 760).

The documentation for this class was generated from the following file:

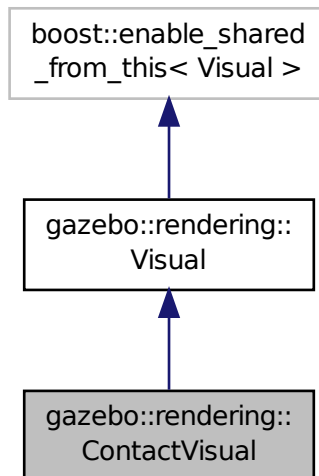
- **ContactSensor.hh**

10.31 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

10.31.1 Detailed Description

Contact visualization.

This class visualizes contact points by drawing arrows in the 3D environment.

10.31.2 Constructor & Destructor Documentation

10.31.2.1 `gazebo::rendering::ContactVisual::ContactVisual (const std::string & _name, VisualPtr _vis, const std::string & _topicName)`

Constructor.

Parameters

in	<code><i>_name</i></code>	Name of the ContactVisual (p. 265)
in	<code><i>_vis</i></code>	Pointer the parent Visual (p. 1034)
in	<code><i>_topicName</i></code>	Name of the topic which publishes the contact information.

10.31.2.2 `virtual gazebo::rendering::ContactVisual::~ContactVisual () [virtual]`

Destructor.

10.31.3 Member Function Documentation

10.31.3.1 `void gazebo::rendering::ContactVisual::SetEnabled (bool _enabled)`

Set to true to enable contact visualization.

Parameters

in	<code><i>_enabled</i></code>	True to show contacts, false to hide.
----	------------------------------	---------------------------------------

The documentation for this class was generated from the following file:

- **ContactVisual.hh**

10.32 gazebo::rendering::Conversions Class Reference

Conversions (p. 266) **Conversions.hh** (p. 1132) **rendering/Conversions.hh** (p. 1132).

```
#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. 123) 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. 123) quaternion.*
- static **math::Quaternion Convert** (const Ogre::Quaternion &_v)
***Ogre** (p. 123) quaternion to Gazebo quaternion.*

10.32.1 Detailed Description

Conversions (p. 266) **Conversions.hh** (p. 1132) **rendering/Conversions.hh** (p. 1132).

A set of utility function to convert between Gazebo and **Ogre** (p. 123) data types

10.32.2 Member Function Documentation

10.32.2.1 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. 123) color value

10.32.2.2 static **common::Color** gazebo::rendering::Conversions::Convert (const Ogre::ColourValue &_clr) [static]

Return the equivalent gazebo color.

Parameters

in	_clr	Ogre (p. 123) color to convert
----	------	---------------------------------------

Returns

Gazebo color value

10.32.2.3 static Ogre::Vector3 gazebo::rendering::Conversions::Convert (const **math::Vector3** &_v) [static]

return **Ogre** (p. 123) Vector from Gazebo Vector3

Parameters

in	_v	Gazebo vector
----	----	---------------

Returns

Ogre (p. 123) vector

10.32.2.4 `static math::Vector3 gazebo::rendering::Conversions::Convert (const Ogre::Vector3 & _v) [static]`

return gazebo Vector from ogre Vector3

Parameters

in	_v	Ogre (p. 123) vector
----	----	-----------------------------

Returns

Gazebo vector

10.32.2.5 `static Ogre::Quaternion gazebo::rendering::Conversions::Convert (const math::Quaternion & _v) [static]`

Gazebo quaternion to **Ogre** (p. 123) quaternion.

Parameters

in	_v	Gazebo quaternion
----	----	-------------------

Returns

Ogre (p. 123) quaternion

10.32.2.6 `static math::Quaternion gazebo::rendering::Conversions::Convert (const Ogre::Quaternion & _v) [static]`

Ogre (p. 123) quaternion to Gazebo quaternion.

Parameters

in	_v	Ogre (p. 123) quaternion return Gazebo quaternion
----	----	--

The documentation for this class was generated from the following file:

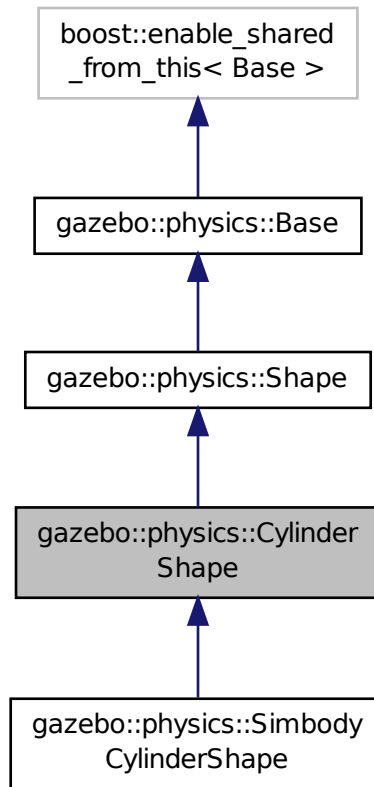
- **Conversions.hh**

10.33 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

10.33.1 Detailed Description

Cylinder collision.

10.33.2 Constructor & Destructor Documentation

10.33.2.1 `gazebo::physics::CylinderShape::CylinderShape (CollisionPtr _parent)` `[explicit]`

Constructor.

Parameters

<code>in</code>	<code>_parent</code>	Parent of the shape.
-----------------	----------------------	----------------------

10.33.2.2 `virtual gazebo::physics::CylinderShape::~~CylinderShape ()` `[virtual]`

Destructor.

10.33.3 Member Function Documentation

10.33.3.1 `void gazebo::physics::CylinderShape::FillMsg (msgs::Geometry & _msg)` `[virtual]`

Fill in the values for a geometry message.

Parameters

<code>out</code>	<code>_msg</code>	The geometry message to fill.
------------------	-------------------	-------------------------------

Implements `gazebo::physics::Shape` (p. 777).

10.33.3.2 `double gazebo::physics::CylinderShape::GetLength ()` `const`

Get length.

Returns

The cylinder length.

10.33.3.3 `double gazebo::physics::CylinderShape::GetRadius () const`

Get radius.

Returns

The cylinder radius.

10.33.3.4 `void gazebo::physics::CylinderShape::Init () [virtual]`

Initialize the cylinder.

Implements **`gazebo::physics::Shape`** (p. 777).

10.33.3.5 `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. 778).

10.33.3.6 `void gazebo::physics::CylinderShape::SetLength (double _length)`

Set length.

Parameters

<code>in</code>	<code>_length</code>	New length of the cylinder.
-----------------	----------------------	-----------------------------

10.33.3.7 `void gazebo::physics::CylinderShape::SetRadius (double _radius)`

Set radius.

Parameters

<code>in</code>	<code>_radius</code>	New radius of the cylinder.
-----------------	----------------------	-----------------------------

10.33.3.8 `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. 778).

10.33.3.9 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. 790).

Referenced by [gazebo::physics::SimbodyCylinderShape::SetSize\(\)](#).

The documentation for this class was generated from the following file:

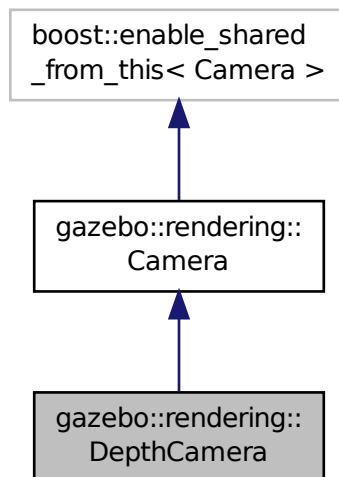
- [CylinderShape.hh](#)

10.34 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

10.34.1 Detailed Description

Depth camera used to render depth data into an image buffer.

10.34.2 Constructor & Destructor Documentation

10.34.2.1 `gazebo::rendering::DepthCamera::DepthCamera (const std::string & _namePrefix, ScenePtr _scene, bool _autoRender = true)`

Constructor.

Parameters

in	<code>_namePrefix</code>	Unique prefix name for the camera.
in	<code>_scene</code>	Scene (p. 728) that will contain the camera
in	<code>_autoRender</code>	Almost everyone should leave this as true.

10.34.2.2 `virtual gazebo::rendering::DepthCamera::~DepthCamera () [virtual]`

Destructor.

10.34.3 Member Function Documentation

10.34.3.1 `template<typename T > event::ConnectionPtr gazebo::rendering::DepthCamera::ConnectNewDepthFrame (T _subscriber) [inline]`

Connect a to the new depth image signal.

Parameters

in	<code>_subscriber</code>	Subscriber callback function
----	--------------------------	------------------------------

Returns

Pointer to the new Connection. This must be kept in scope

References `gazebo::event::EventT< T >::Connect()`.

10.34.3.2 `template<typename T > event::ConnectionPtr gazebo::rendering::DepthCamera::ConnectNewRGBPointCloud (T _subscriber) [inline]`

Connect a to the new rgb point cloud signal.

Parameters

in	<code>_subscriber</code>	Subscriber callback function
----	--------------------------	------------------------------

Returns

Pointer to the new Connection. This must be kept in scope

References `gazebo::event::EventT< T >::Connect()`.

10.34.3.3 void gazebo::rendering::DepthCamera::CreateDepthTexture (const std::string & *_textureName*)

Create a texture which will hold the depth data.

Parameters

in	<i>_textureName</i>	Name of the texture to create
----	---------------------	-------------------------------

10.34.3.4 void gazebo::rendering::DepthCamera::DisconnectNewDepthFrame (event::ConnectionPtr & *_c*) [inline]

Disconnect from an depth image singal.

Parameters

in	<i>_c</i>	The connection to disconnect
----	-----------	------------------------------

References gazebo::event::EventT< T >::Disconnect().

10.34.3.5 void gazebo::rendering::DepthCamera::DisconnectNewRGBPointCloud (event::ConnectionPtr & *c*) [inline]

Disconnect from an rgb point cloud singal.

Parameters

in	<i>_c</i>	The connection to disconnect
----	-----------	------------------------------

References gazebo::event::EventT< T >::Disconnect().

10.34.3.6 void gazebo::rendering::DepthCamera::Fini () [virtual]

Finalize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 188).

10.34.3.7 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

10.34.3.8 void gazebo::rendering::DepthCamera::Init () [virtual]

Initialize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 196).

10.34.3.9 void gazebo::rendering::DepthCamera::Load (sdf::ElementPtr & *_sdf*)

Load the camera with a set of parmeters.

Parameters

in	<code>_sdf</code>	The SDF camera info
----	-------------------	---------------------

10.34.3.10 `void gazebo::rendering::DepthCamera::Load() [virtual]`

Load the camera with default parameters.

Reimplemented from `gazebo::rendering::Camera` (p. 197).

10.34.3.11 `virtual void gazebo::rendering::DepthCamera::PostRender() [virtual]`

Render the camera.

Reimplemented from `gazebo::rendering::Camera` (p. 197).

10.34.3.12 `virtual void gazebo::rendering::DepthCamera::SetDepthTarget(Ogre::RenderTarget * _target) [virtual]`

Set the render target, which renders the depth data.

Parameters

in	<code>_target</code>	Pointer to the render target
----	----------------------	------------------------------

10.34.4 Member Data Documentation

10.34.4.1 `Ogre::RenderTarget* gazebo::rendering::DepthCamera::depthTarget [protected]`

Pointer to the depth target.

10.34.4.2 `Ogre::Texture* gazebo::rendering::DepthCamera::depthTexture [protected]`

Pointer to the depth texture.

10.34.4.3 `Ogre::Viewport* gazebo::rendering::DepthCamera::depthViewport [protected]`

Pointer to the depth viewport.

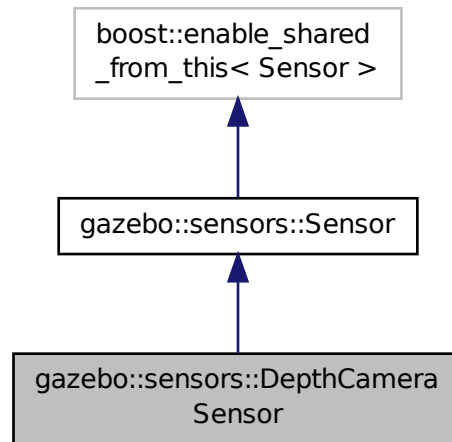
The documentation for this class was generated from the following file:

- `DepthCamera.hh`

10.35 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. 272).*
- 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 void **UpdateImpl** (bool _force)
Update the sensor information.

Additional Inherited Members

10.35.1 Constructor & Destructor Documentation

10.35.1.1 gazebo::sensors::DepthCameraSensor::DepthCameraSensor ()

Constructor.

10.35.1.2 virtual gazebo::sensors::DepthCameraSensor::~~DepthCameraSensor () [virtual]

Destructor.

10.35.2 Member Function Documentation

10.35.2.1 virtual void gazebo::sensors::DepthCameraSensor::Fini () [protected], [virtual]

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 755).

10.35.2.2 rendering::DepthCameraPtr gazebo::sensors::DepthCameraSensor::GetDepthCamera () const [inline]

Returns a pointer to the **rendering::DepthCamera** (p. 272).

Returns

Depth Camera pointer

10.35.2.3 virtual void gazebo::sensors::DepthCameraSensor::Init () [protected], [virtual]

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 758).

10.35.2.4 virtual void gazebo::sensors::DepthCameraSensor::Load (const std::string & *_worldName*, sdf::ElementPtr & *_sdf*) [protected], [virtual]

Load the sensor with SDF parameters.

Parameters

in	<i>_sdf</i>	SDF Sensor (p. 751) parameters
in	<i>_worldName</i>	Name of world to load from

10.35.2.5 virtual void gazebo::sensors::DepthCameraSensor::Load (const std::string & *_worldName*) [protected], [virtual]

Load the sensor with default parameters.

Parameters

in	<i>_worldName</i>	Name of world to load from
----	-------------------	----------------------------

Reimplemented from **gazebo::sensors::Sensor** (p. 759).

10.35.2.6 bool gazebo::sensors::DepthCameraSensor::SaveFrame (const std::string & *filename*)

Saves an image frame of depth camera sensor to file.

Parameters

in	<i>Name</i>	of file to save as
----	-------------	--------------------

Returns

True if saved, false if not

10.35.2.7 virtual void gazebo::sensors::DepthCameraSensor::SetActive (bool *value*) [virtual]

Set whether the sensor is active or not.

Parameters

in	<i>_value</i>	True if active, false if not
----	---------------	------------------------------

Reimplemented from **gazebo::sensors::Sensor** (p. 759).

10.35.2.8 virtual void gazebo::sensors::DepthCameraSensor::UpdateImpl (bool *force*) [protected],[virtual]

Update the sensor information.

Parameters

in	<i>_force</i>	True if update is forced, false if not
----	---------------	--

Reimplemented from **gazebo::sensors::Sensor** (p. 760).

The documentation for this class was generated from the following file:

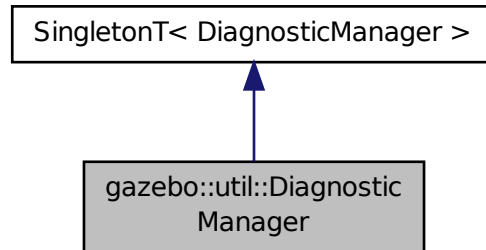
- **DepthCameraSensor.hh**

10.36 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

10.36.1 Detailed Description

A diagnostic manager class.

10.36.2 Member Function Documentation

10.36.2.1 `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

10.36.2.2 `boost::filesystem::path gazebo::util::DiagnosticManager::GetLogPath () const`

Get the path in which logs are stored.

Returns

The path in which logs are stored.

10.36.2.3 `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

10.36.2.4 `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

10.36.2.5 `int gazebo::util::DiagnosticManager::GetTimerCount () const`

Get the number of timers.

Returns

The number of timers

10.36.2.6 void gazebo::util::DiagnosticManager::Init (const std::string & *_worldName*)

Initialize to report diagnostics about a world.

Parameters

in	<i>_worldName</i>	Name of the world.
----	-------------------	--------------------

10.36.2.7 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

in	<i>_name</i>	Name of the timer to access.
in	<i>_prefix</i>	Informational string that is output with the elapsed time.

10.36.2.8 void gazebo::util::DiagnosticManager::StartTimer (const std::string & *_name*)

Start a new timer instance.

Parameters

in	<i>_name</i>	Name of the timer.
----	--------------	--------------------

Returns

A pointer to the new diagnostic timer

10.36.2.9 void gazebo::util::DiagnosticManager::StopTimer (const std::string & *_name*)

Stop a currently running timer.

Parameters

in	<i>_name</i>	Name of the timer to stop.
----	--------------	----------------------------

The documentation for this class was generated from the following file:

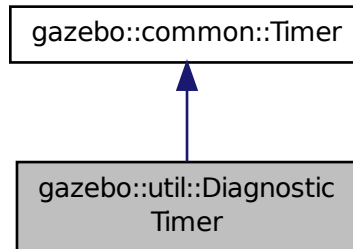
- **Diagnostics.hh**

10.37 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.

10.37.1 Detailed Description

A timer designed for diagnostics.

10.37.2 Constructor & Destructor Documentation

10.37.2.1 gazebo::util::DiagnosticTimer::DiagnosticTimer (const std::string & _name)

Constructor.

Parameters

in	<i>_name</i>	Name of the timer
----	--------------	-------------------

10.37.2.2 `virtual gazebo::util::DiagnosticTimer::~~DiagnosticTimer () [virtual]`

Destructor.

10.37.3 Member Function Documentation

10.37.3.1 `const std::string gazebo::util::DiagnosticTimer::GetName () const [inline]`

Get the name of the timer.

Returns

The name of timer

10.37.3.2 `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.
----	----------------	---

10.37.3.3 `virtual void gazebo::util::DiagnosticTimer::Start () [virtual]`

Start the timer.

Reimplemented from `gazebo::common::Timer` (p. 967).

10.37.3.4 `virtual void gazebo::util::DiagnosticTimer::Stop () [virtual]`

Stop the timer.

Reimplemented from `gazebo::common::Timer` (p. 967).

The documentation for this class was generated from the following file:

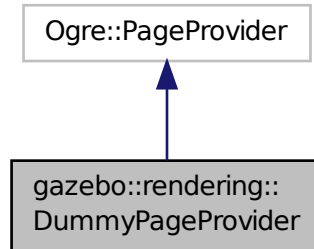
- **Diagnostics.hh**

10.38 gazebo::rendering::DummyPageProvider Class Reference

Pretends to provide procedural page content to avoid page loading.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::DummyPageProvider:



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.

10.38.1 Detailed Description

Pretends to provide procedural page content to avoid page loading.

10.38.2 Member Function Documentation

10.38.2.1 **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.

10.38.2.2 **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.

10.38.2.3 `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.

10.38.2.4 `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:

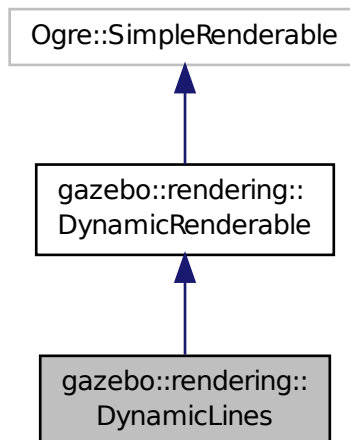
- **Heightmap.hh**

10.39 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. 123)'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

10.39.1 Detailed Description

Class for drawing lines that can change.

10.39.2 Constructor & Destructor Documentation

10.39.2.1 gazebo::rendering::DynamicLines::DynamicLines (RenderOpType _opType = RENDERING_LINE_STRIP)

Constructor.

Parameters

in	_opType	The type of Line
----	---------	------------------

10.39.2.2 virtual gazebo::rendering::DynamicLines::~DynamicLines () [virtual]

Destructor.

10.39.3 Member Function Documentation

10.39.3.1 `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	<code>_pt</code>	math::Vector3 (p. 1004) point
in	<code>_color</code>	common::Color (p. 226) Point color

10.39.3.2 `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	<code>_x</code>	X position
in	<code>_y</code>	Y position
in	<code>_z</code>	Z position
in	<code>_color</code>	common::Color (p. 226) Point color

10.39.3.3 `void gazebo::rendering::DynamicLines::Clear ()`

Remove all points from the point list.

10.39.3.4 `static std::string gazebo::rendering::DynamicLines::GetMovableType () [static]`

Get type of movable.

Returns

This returns "gazebo::dynamiclines"

10.39.3.5 `virtual const Ogre::String& gazebo::rendering::DynamicLines::getMovableType () const [virtual]`

Overridden function from **Ogre** (p. 123)'s base class.

Returns

Returns "gazebo::ogredynamicslines"

10.39.3.6 `const math::Vector3& gazebo::rendering::DynamicLines::GetPoint (unsigned int _index) const`

Return the location of an existing point in the point list.

Parameters

in	<code>_index</code>	Number of the point to return
----	---------------------	-------------------------------

Returns

math::Vector3 (p. 1004) value of the point

10.39.3.7 unsigned int gazebo::rendering::DynamicLines::GetPointCount () const

Return the total number of points in the point list.

Returns

Number of points

10.39.3.8 void gazebo::rendering::DynamicLines::SetColor (unsigned int *_index*, const common::Color & *_color*)

Change the color of an existing point in the point list.

Parameters

in	<i>_index</i>	Index of the point to set
in	<i>_color</i>	common::Color (p. 226) Pixelcolor color to set the point to

10.39.3.9 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. 1004) value to set the point to

10.39.3.10 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:

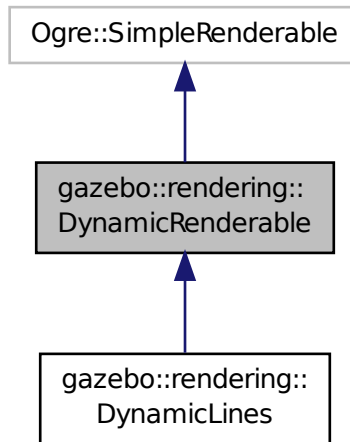
- **DynamicLines.hh**

10.40 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.

10.40.1 Detailed Description

Abstract base class providing mechanisms for dynamically growing hardware buffers.

10.40.2 Constructor & Destructor Documentation

10.40.2.1 gazebo::rendering::DynamicRenderable::DynamicRenderable ()

Constructor.

10.40.2.2 virtual gazebo::rendering::DynamicRenderable::~~DynamicRenderable () [virtual]

Virtual destructor.

10.40.3 Member Function Documentation

10.40.3.1 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.

10.40.3.2 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.

10.40.3.3 `virtual Ogre::Real gazebo::rendering::DynamicRenderable::getBoundingRadius () const` [virtual]

Implementation of `Ogre::SimpleRenderable`.

Returns

The bounding radius

10.40.3.4 `std::string gazebo::rendering::DynamicRenderable::GetMovableType () const`

Get type of movable.

Returns

This returns "gazebo::DynamicRenderable"

10.40.3.5 `RenderOpType gazebo::rendering::DynamicRenderable::GetOperationType () const`

Get the render operation type.

Returns

The render operation type.

10.40.3.6 `virtual Ogre::Real gazebo::rendering::DynamicRenderable::getSquaredViewDepth (const Ogre::Camera * _cam) const`
[virtual]

Implementation of `Ogre::SimpleRenderable`.

Parameters

in	_cam	Pointer to the Ogre (p. 123) camera that views the renderable.
----	------	---

Returns

The squared depth in the **Camera** (p. 179)'s view

10.40.3.7 `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. 291).

Parameters

in	_opType	The type of render operation to perform.
in	_useIndices	Specifies whether to use indices to determine the vertices to use as input.

10.40.3.8 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

in	<i>_vertexCount</i>	The number of vertices the buffer must hold.
in	<i>_indexCount</i>	The number of indices the buffer must hold. This parameter is ignored if not using indices.

10.40.3.9 void gazebo::rendering::DynamicRenderable::SetOperationType (RenderOpType *_opType*)

Set the render operation type.

Parameters

in	<i>_opType</i>	The type of render operation to perform.
----	----------------	--

10.40.4 Member Data Documentation

10.40.4.1 size.t gazebo::rendering::DynamicRenderable::indexBufferCapacity [protected]

Maximum capacity of the currently allocated index buffer.

10.40.4.2 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:

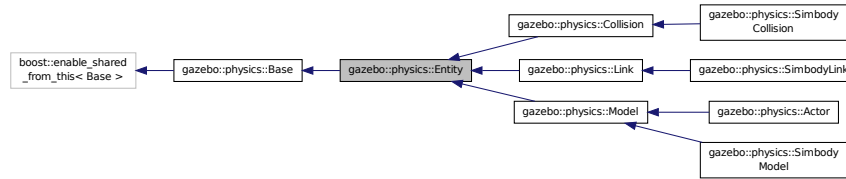
- **DynamicRenderable.hh**

10.41 gazebo::physics::Entity Class Reference

Base (p. 153) 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 \sim **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. 304).*
- **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 _updateChildren=true)
*Set angular and linear rates of an **physics::Entity** (p. 293).*
- 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

10.41.1 Detailed Description

Base (p. 153) class for all physics objects in Gazebo.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 `gazebo::physics::Entity::Entity (BasePtr _parent) [explicit]`

Constructor.

Parameters

in	_parent	Parent of the entity.
----	---------	-----------------------

10.41.2.2 `virtual gazebo::physics::Entity::~~Entity () [virtual]`

Destructor.

10.41.3 Member Function Documentation

10.41.3.1 virtual void gazebo::physics::Entity::Fini () [virtual]

Finalize the entity.

Reimplemented from **gazebo::physics::Base** (p. 158).

Reimplemented in **gazebo::physics::Actor** (p. 128), **gazebo::physics::Link** (p. 463), **gazebo::physics::Model** (p. 542), **gazebo::physics::Collision** (p. 216), and **gazebo::physics::SimbodyLink** (p. 818).

10.41.3.2 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. 463), **gazebo::physics::Model** (p. 542), **gazebo::physics::Collision** (p. 216), and **gazebo::physics::SimbodyCollision** (p. 787).

10.41.3.3 CollisionPtr gazebo::physics::Entity::GetChildCollision (const std::string & _name)

Get a child collision entity, if one exists.

Parameters

in	<i>_name</i>	Name of the child collision object.
----	--------------	-------------------------------------

Returns

Pointer to the **Collision** (p. 213) object, or NULL if not found.

10.41.3.4 LinkPtr gazebo::physics::Entity::GetChildLink (const std::string & _name)

Get a child linke entity, if one exists.

Parameters

in	<i>_name</i>	Name of the child Link (p. 455) object.
----	--------------	--

Returns

Pointer to the **Link** (p. 455) object, or NULL if not found.

10.41.3.5 math::Box gazebo::physics::Entity::GetCollisionBoundingBox () const

Returns collision bounding box.

Returns

Collision bounding box.

10.41.3.6 `const math::Pose& gazebo::physics::Entity::GetDirtyPose () const`

Returns **Entity::dirtyPose** (p. 304).

The dirty pose is the pose set by the physics engine before its value is propagated to the rest of the simulator.

Returns

The dirty pose of the entity.

10.41.3.7 `math::Pose gazebo::physics::Entity::GetInitialRelativePose () const`

Get the initial relative pose.

Returns

The initial relative pose.

10.41.3.8 `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.

10.41.3.9 `ModelPtr gazebo::physics::Entity::GetParentModel ()`

Get the parent model, if one exists.

Returns

Pointer to a model, or NULL if no parent model exists.

10.41.3.10 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularAccel () const` `[inline],[virtual]`

Get the angular acceleration of the entity.

Returns

A **math::Vector3** (p. 1004) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 466), **gazebo::physics::Collision** (p. 217), and **gazebo::physics::Model** (p. 544).

10.41.3.11 virtual **math::Vector3** gazebo::physics::Entity::GetRelativeAngularVel () const [inline],[virtual]

Get the angular velocity of the entity.

Returns

A **math::Vector3** (p. 1004) for the velocity.

Reimplemented in **gazebo::physics::Link** (p. 466), **gazebo::physics::Collision** (p. 218), and **gazebo::physics::Model** (p. 545).

10.41.3.12 virtual **math::Vector3** gazebo::physics::Entity::GetRelativeLinearAccel () const [inline],[virtual]

Get the linear acceleration of the entity.

Returns

A **math::Vector3** (p. 1004) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 466), **gazebo::physics::Collision** (p. 218), and **gazebo::physics::Model** (p. 545).

10.41.3.13 virtual **math::Vector3** gazebo::physics::Entity::GetRelativeLinearVel () const [inline],[virtual]

Get the linear velocity of the entity.

Returns

A **math::Vector3** (p. 1004) for the linear velocity.

Reimplemented in **gazebo::physics::Link** (p. 466), **gazebo::physics::Collision** (p. 218), and **gazebo::physics::Model** (p. 545).

10.41.3.14 **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.

10.41.3.15 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. 1004) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 467), **gazebo::physics::Collision** (p. 219), and **gazebo::physics::Model** (p. 546).

10.41.3.16 `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. 1004) for the velocity.

Reimplemented in **gazebo::physics::Collision** (p. 219), **gazebo::physics::Model** (p. 546), and **gazebo::physics::SimbodyLink** (p. 818).

10.41.3.17 `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. 1004) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 468), **gazebo::physics::Collision** (p. 219), and **gazebo::physics::Model** (p. 546).

10.41.3.18 `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. 1004) for the linear velocity.

Reimplemented in **gazebo::physics::Collision** (p. 219), and **gazebo::physics::Model** (p. 546).

10.41.3.19 `const math::Pose& gazebo::physics::Entity::GetWorldPose () const [inline]`

Get the absolute pose of the entity.

Returns

The absolute pose of the entity.

10.41.3.20 `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.

10.41.3.21 `bool gazebo::physics::Entity::IsStatic () const`

Return whether this entity is static.

Returns

True if static.

10.41.3.22 `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. 161).

Reimplemented in **gazebo::physics::Link** (p. 469), **gazebo::physics::Actor** (p. 128), **gazebo::physics::Model** (p. 546), **gazebo::physics::Collision** (p. 220), **gazebo::physics::SimbodyCollision** (p. 787), **gazebo::physics::SimbodyLink** (p. 820), and **gazebo::physics::SimbodyModel** (p. 826).

10.41.3.23 `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. 470), **gazebo::physics::Model** (p. 547), **gazebo::physics::SimbodyLink** (p. 820), and **gazebo::physics::SimbodyCollision** (p. 788).

10.41.3.24 `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. 293) this Entity (p. 293) should be ontop of.
----	-------------	--

10.41.3.25 `void gazebo::physics::Entity::PlaceOnNearestEntityBelow ()`

Move this entity to be ontop of the nearest entity below.

10.41.3.26 `virtual void gazebo::physics::Entity::Reset () [virtual]`

Reset the entity.

Reimplemented from **gazebo::physics::Base** (p. 162).

Reimplemented in **gazebo::physics::Model** (p. 547), and **gazebo::physics::Link** (p. 470).

10.41.3.27 `void gazebo::physics::Entity::SetAnimation (const common::PoseAnimationPtr & _anim, boost::function< void()> _onComplete)`

Set an animation for this entity.

Parameters

in	<code>_anim</code>	Pose animation.
in	<code>_onComplete</code>	Callback for when the animation completes.

10.41.3.28 `void gazebo::physics::Entity::SetAnimation (common::PoseAnimationPtr _anim)`

Set an animation for this entity.

Parameters

in	<code>_anim</code>	Pose animation.
----	--------------------	-----------------

10.41.3.29 `void gazebo::physics::Entity::SetCanonicalLink (bool _value)`

Set to true if this entity is a canonical link for a model.

Parameters

in	<code>_value</code>	True if the link is canonical.
----	---------------------	--------------------------------

10.41.3.30 `void gazebo::physics::Entity::SetInitialRelativePose (const math::Pose & _pose)`

Set the initial pose.

Parameters

in	<code>_pose</code>	The initial pose.
----	--------------------	-------------------

10.41.3.31 `virtual void gazebo::physics::Entity::SetName (const std::string & _name)` [virtual]

Set the name of the entity.

Parameters

in	<code>_name</code>	The new name.
----	--------------------	---------------

Reimplemented from `gazebo::physics::Base` (p. 162).

10.41.3.32 `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	<code>_pose</code>	The new pose.
in	<code>_notify</code>	True = tell children of the pose change.
in	<code>_publish</code>	True to publish the pose.

10.41.3.33 `void gazebo::physics::Entity::SetStatic (const bool & _static)`

Set whether this entity is static: immovable.

Parameters

in	<code>_static</code>	True = static.
----	----------------------	----------------

10.41.3.34 `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.

10.41.3.35 `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. 293).

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.

10.41.3.36 `virtual void gazebo::physics::Entity::StopAnimation () [virtual]`

Stop the current animation, if any.

Reimplemented in **gazebo::physics::Model** (p. 551).

10.41.3.37 `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. 163).

Reimplemented in **gazebo::physics::Actor** (p. 129), **gazebo::physics::Link** (p. 475), **gazebo::physics::Model** (p. 551), and **gazebo::physics::Collision** (p. 222).

10.41.4 Member Data Documentation

10.41.4.1 **common::PoseAnimationPtr** gazebo::physics::Entity::animation [protected]

Current pose animation.

10.41.4.2 **event::ConnectionPtr** gazebo::physics::Entity::animationConnection [protected]

Connection used to update an animation.

10.41.4.3 **math::Pose** gazebo::physics::Entity::animationStartPose [protected]

Start pose of an animation.

10.41.4.4 **std::vector<event::ConnectionPtr>** gazebo::physics::Entity::connections [protected]

All our event connections.

10.41.4.5 **math::Pose** gazebo::physics::Entity::dirtyPose [protected]

The pose set by a physics engine.

10.41.4.6 **transport::NodePtr** gazebo::physics::Entity::node [protected]

Communication node.

10.41.4.7 **EntityPtr** gazebo::physics::Entity::parentEntity [protected]

A helper that prevents numerous dynamic_casts.

10.41.4.8 **common::Time** gazebo::physics::Entity::prevAnimationTime [protected]

Previous time an animation was updated.

10.41.4.9 **transport::PublisherPtr** gazebo::physics::Entity::requestPub [protected]

Request publisher.

10.41.4.10 **math::Vector3** gazebo::physics::Entity::scale [protected]

Scale of the entity.

10.41.4.11 `transport::PublisherPtr gazebo::physics::Entity::visPub` [protected]

Visual publisher.

10.41.4.12 `msgs::Visual* gazebo::physics::Entity::visualMsg` [protected]

Visual message container.

The documentation for this class was generated from the following file:

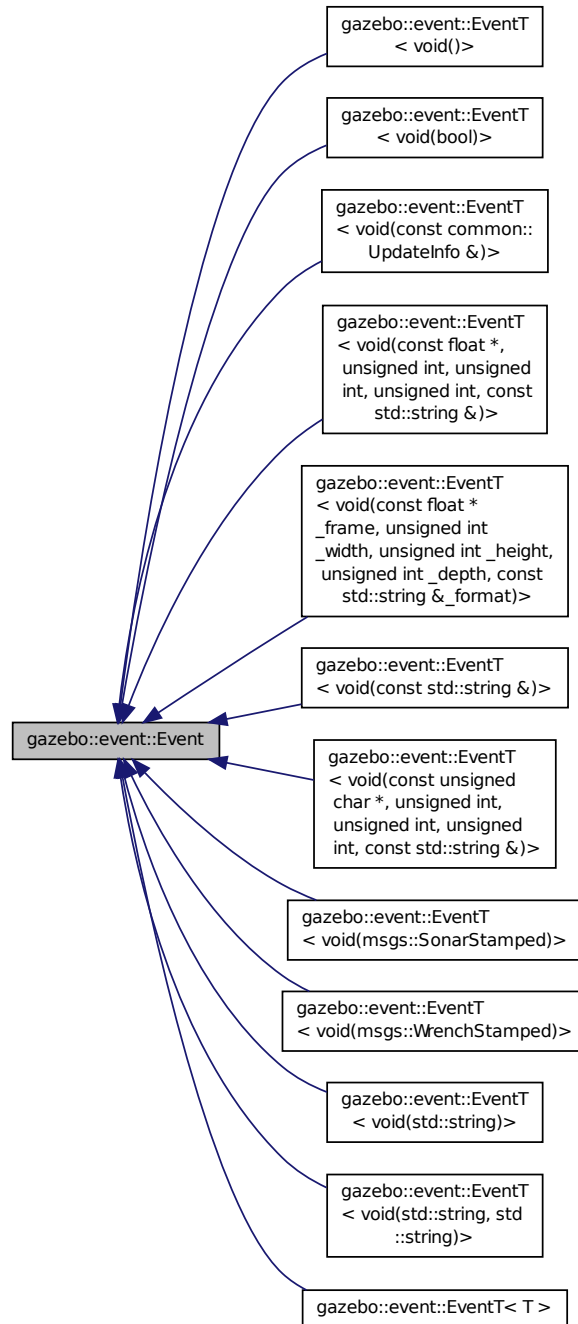
- **Entity.hh**

10.42 gazebo::event::Event Class Reference

Base class for all events.

```
#include <common/common.hh>
```

Inheritance diagram for gazebo::event::Event:



Public Member Functions

- virtual `~Event()`

Constructor.

- virtual void **Disconnect** (**ConnectionPtr** _c)=0

Disconnect.

- virtual void **Disconnect** (int _id)=0

Disconnect.

10.42.1 Detailed Description

Base class for all events.

10.42.2 Constructor & Destructor Documentation

10.42.2.1 virtual gazebo::event::Event::~Event () [inline],[virtual]

Constructor.

10.42.3 Member Function Documentation

10.42.3.1 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. 44), **gazebo::event::EventT< void(msgs::SonarStamped)>** (p. 44), **gazebo::event::EventT< void(std::string)>** (p. 44), **gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 44), **gazebo::event::EventT< void(msgs::WrenchStamped)>** (p. 44), **gazebo::event::EventT< void(const float *_frame, unsigned int _width, unsigned int _height, unsigned int _depth, const std::string &_format)>** (p. 44), **gazebo::event::EventT< void(const std::string &)>** (p. 44), **gazebo::event::EventT< void()>** (p. 44), **gazebo::event::EventT< void(const common::UpdateInfo &)>** (p. 44), **gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 44), **gazebo::event::EventT< void(std::string, std::string)>** (p. 44), and **gazebo::event::EventT< void(bool)>** (p. 44).

10.42.3.2 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. 45), **gazebo::event::EventT< void(msgs::SonarStamped)>** (p. 45), **gazebo::event::EventT< void(std::string)>** (p. 45), **gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 45), **gazebo::event::EventT< void(msgs::WrenchStamped)>** (p. 45), **gazebo::event::EventT< void(const float *_frame, unsigned int _width, unsigned int _height, unsigned int _depth, const std::string &_format)>** (p. 45), **gazebo::event::EventT< void(const std::string &)>** (p. 45), **gazebo::event::EventT< void()>** (p. 45), **gazebo::event::EventT< void(const common::**

UpdateInfo &> (p. 45), **gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &>** (p. 45), **gazebo::event::EventT< void(std::string, std::string)>** (p. 45), and **gazebo::event::EventT< void(bool)>** (p. 45).

The documentation for this class was generated from the following file:

- **Event.hh**

10.43 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.

10.43.1 Detailed Description

Base class for rendering events.

10.43.2 Member Function Documentation

10.43.2.1 `template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectCreateScene (T _subscriber) [inline],[static]`

Connect to a scene created event.

Parameters

in	<i>_subscriber</i>	Callback to trigger when event occurs.
----	--------------------	--

Returns

Pointer the connection. This must stay in scope.

References gazebo::event::EventT< T >::Connect(), and createScene.

10.43.2.2 `template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectRemoveScene (T _subscriber) [inline],[static]`

Connect to a scene removed event.

Parameters

in	<i>_subscriber</i>	Callback to trigger when event occurs.
----	--------------------	--

Returns

Pointer the connection. This must stay in scope.

References gazebo::event::EventT< T >::Connect(), and removeScene.

10.43.2.3 `static void gazebo::rendering::Events::DisconnectCreateScene (event::ConnectionPtr _connection) [inline],[static]`

Disconnect from a scene created event.

Parameters

in	<i>_connection</i>	The connection to disconnect.
----	--------------------	-------------------------------

References createScene, and gazebo::event::EventT< T >::Disconnect().

10.43.2.4 `static void gazebo::rendering::Events::DisconnectRemoveScene (event::ConnectionPtr _connection) [inline],[static]`

Disconnect from a scene removed event.

Parameters

in	<i>_connection</i>	The connection to disconnect.
----	--------------------	-------------------------------

References gazebo::event::EventT< T >::Disconnect(), and removeScene.

10.43.3 Member Data Documentation

10.43.3.1 `event::EventT<void (const std::string &)> gazebo::rendering::Events::createScene [static]`

The event used to trigger a create scene event.

Referenced by `ConnectCreateScene()`, and `DisconnectCreateScene()`.

10.43.3.2 `event::EventT<void (const std::string &)> gazebo::rendering::Events::removeScene` [static]

The event used to trigger a remove scene event.

Referenced by `ConnectRemoveScene()`, and `DisconnectRemoveScene()`.

The documentation for this class was generated from the following file:

- **RenderEvents.hh**

10.44 `gazebo::event::Events` Class Reference

An **Event** (p. 305) 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.

10.44.1 Detailed Description

An **Event** (p. 305) class to get notifications for simulator events.

10.44.2 Member Function Documentation

10.44.2.1 `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

References `addEntity`, and `gazebo::event::EventT< T >::Connect()`.

10.44.2.2 `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

References `gazebo::event::EventT< T >::Connect()`, and `entityCreated`.

10.44.2.3 `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

References `gazebo::event::EventT< T >::Connect()`, and `deleteEntity`.

10.44.2.4 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStart (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the diagnostic timer start signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

References gazebo::event::EventT< T >::Connect(), and diagTimerStart.

10.44.2.5 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStop (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the diagnostic timer stop signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

References gazebo::event::EventT< T >::Connect(), and diagTimerStop.

10.44.2.6 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPause (T _subscriber)` `[inline],`
`[static]`

Connect a boost::slot the the pause signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

References gazebo::event::EventT< T >::Connect(), and pause.

10.44.2.7 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPostRender (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the post render update signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

References gazebo::event::EventT< T >::Connect(), and postRender.

10.44.2.8 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPreRender (T _subscriber)`
`[inline], [static]`

Render start signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

References gazebo::event::EventT< T >::Connect(), and preRender.

10.44.2.9 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectRender (T _subscriber)`
`[inline], [static]`

Connect a boost::slot the render update signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

References gazebo::event::EventT< T >::Connect(), and render.

10.44.2.10 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectSetSelectedEntity (T _subscriber)`
`[inline], [static]`

Connect a boost::slot the set selected entity.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

References gazebo::event::EventT< T >::Connect(), and setSelectedEntity.

10.44.2.11 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectSigInt (T _subscriber)`
`[inline], [static]`

Connect a boost::slot to the sigint event.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

References gazebo::event::EventT< T >::Connect(), and sigInt.

10.44.2.12 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStep (T _subscriber) [inline], [static]`

Connect a boost::slot the the step signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

References gazebo::event::EventT< T >::Connect(), and step.

10.44.2.13 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStop (T _subscriber) [inline], [static]`

Connect a boost::slot the the stop signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

References gazebo::event::EventT< T >::Connect(), and stop.

10.44.2.14 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldCreated (T _subscriber) [inline], [static]`

Connect a boost::slot the the world created signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

References gazebo::event::EventT< T >::Connect(), and worldCreated.

10.44.2.15 `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

References gazebo::event::EventT< T >::Connect(), and worldUpdateBegin.

10.44.2.16 `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

References gazebo::event::EventT< T >::Connect(), and worldUpdateEnd.

10.44.2.17 `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
-----------------	---------------------------------	------------------------------

References addEntity, and gazebo::event::EventT< T >::Disconnect().

10.44.2.18 `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
-----------------	---------------------------------	------------------------------

References gazebo::event::EventT< T >::Disconnect(), and entityCreated.

10.44.2.19 `static void gazebo::event::Events::DisconnectDeleteEntity (ConnectionPtr _subscriber) [inline], [static]`

Disconnect a boost::slot the delete entity.

Parameters

in	_subscriber	the subscriber to this event
----	-------------	------------------------------

References deleteEntity, and gazebo::event::EventT< T >::Disconnect().

10.44.2.20 `static void gazebo::event::Events::DisconnectDiagTimerStart (ConnectionPtr _subscriber) [inline], [static]`

Disconnect a boost::slot the diagnostic timer start signal.

Parameters

in	_subscriber	the subscriber to this event
----	-------------	------------------------------

References diagTimerStart, and gazebo::event::EventT< T >::Disconnect().

10.44.2.21 `static void gazebo::event::Events::DisconnectDiagTimerStop (ConnectionPtr _subscriber) [inline], [static]`

Disconnect a boost::slot the diagnostic timer stop signal.

Parameters

in	_subscriber	the subscriber to this event
----	-------------	------------------------------

References diagTimerStop, and gazebo::event::EventT< T >::Disconnect().

10.44.2.22 `static void gazebo::event::Events::DisconnectPause (ConnectionPtr _subscriber) [inline], [static]`

Disconnect a boost::slot the the pause signal.

Parameters

in	_subscriber	the subscriber to this event
----	-------------	------------------------------

References gazebo::event::EventT< T >::Disconnect(), and pause.

10.44.2.23 `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
----	--------------------	------------------------------

References gazebo::event::EventT< T >::Disconnect(), and postRender.

10.44.2.24 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
----	--------------------	------------------------------

References gazebo::event::EventT< T >::Disconnect(), and preRender.

10.44.2.25 static void gazebo::event::Events::DisconnectRender (**ConnectionPtr** *_subscriber*) [inline],[static]

Disconnect a boost::slot the render update signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

References gazebo::event::EventT< T >::Disconnect(), and render.

10.44.2.26 static void gazebo::event::Events::DisconnectSetSelectedEntity (**ConnectionPtr** *_subscriber*) [inline],[static]

Disconnect a boost::slot the set selected entity.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

References gazebo::event::EventT< T >::Disconnect(), and setSelectedEntity.

10.44.2.27 static void gazebo::event::Events::DisconnectSigInt (**ConnectionPtr** *_subscriber*) [inline],[static]

Disconnect a boost::slot to the sigint event.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

References gazebo::event::EventT< T >::Disconnect(), and sigInt.

10.44.2.28 static void gazebo::event::Events::DisconnectStep (**ConnectionPtr** *_subscriber*) [inline],[static]

Disconnect a boost::slot the the step signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

References gazebo::event::EventT< T >::Disconnect(), and step.

10.44.2.29 `static void gazebo::event::Events::DisconnectStop (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the the stop signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

References gazebo::event::EventT< T >::Disconnect(), and stop.

10.44.2.30 `static void gazebo::event::Events::DisconnectWorldCreated (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the the world created signal.

References gazebo::event::EventT< T >::Disconnect(), and worldCreated.

10.44.2.31 `static void gazebo::event::Events::DisconnectWorldUpdateBegin (ConnectionPtr _subscriber) [static]`

Disconnect a boost::slot the the world update start signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

10.44.2.32 `static void gazebo::event::Events::DisconnectWorldUpdateEnd (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the the world update end signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

References gazebo::event::EventT< T >::Disconnect(), and worldUpdateEnd.

10.44.3 Member Data Documentation

10.44.3.1 `EventT<void (std::string)> gazebo::event::Events::addEntity [static]`

An entity has been added.

Referenced by ConnectAddEntity(), and DisconnectAddEntity().

10.44.3.2 `EventT<void (std::string)> gazebo::event::Events::deleteEntity` [static]

An entity has been deleted.

Referenced by `ConnectDeleteEntity()`, and `DisconnectDeleteEntity()`.

10.44.3.3 `EventT<void (std::string)> gazebo::event::Events::diagTimerStart` [static]

Diagnostic timer start.

Referenced by `ConnectDiagTimerStart()`, and `DisconnectDiagTimerStart()`.

10.44.3.4 `EventT<void (std::string)> gazebo::event::Events::diagTimerStop` [static]

Diagnostic timer stop.

Referenced by `ConnectDiagTimerStop()`, and `DisconnectDiagTimerStop()`.

10.44.3.5 `EventT<void (std::string)> gazebo::event::Events::entityCreated` [static]

An entity has been created.

Referenced by `ConnectCreateEntity()`, and `DisconnectCreateEntity()`.

10.44.3.6 `EventT<void (bool)> gazebo::event::Events::pause` [static]

Pause signal.

Referenced by `ConnectPause()`, and `DisconnectPause()`.

10.44.3.7 `EventT<void ()> gazebo::event::Events::postRender` [static]

Post-Render.

Referenced by `ConnectPostRender()`, and `DisconnectPostRender()`.

10.44.3.8 `EventT<void ()> gazebo::event::Events::preRender` [static]

Pre-render.

Referenced by `ConnectPreRender()`, and `DisconnectPreRender()`.

10.44.3.9 `EventT<void ()> gazebo::event::Events::render` [static]

Render.

Referenced by `ConnectRender()`, and `DisconnectRender()`.

10.44.3.10 `EventT<void (std::string, std::string)> gazebo::event::Events::setSelectedEntity` [static]

An entity has been selected.

Referenced by `ConnectSetSelectedEntity()`, and `DisconnectSetSelectedEntity()`.

10.44.3.11 **EventT<void ()> gazebo::event::Events::sigInt** [static]

Simulation stop signal.

Referenced by `ConnectSigInt()`, and `DisconnectSigInt()`.

10.44.3.12 **EventT<void ()> gazebo::event::Events::step** [static]

Step the simulation once signal.

Referenced by `ConnectStep()`, and `DisconnectStep()`.

10.44.3.13 **EventT<void ()> gazebo::event::Events::stop** [static]

Simulation stop signal.

Referenced by `ConnectStop()`, and `DisconnectStop()`.

10.44.3.14 **EventT<void (std::string)> gazebo::event::Events::worldCreated** [static]

A world has been created.

Referenced by `ConnectWorldCreated()`, and `DisconnectWorldCreated()`.

10.44.3.15 **EventT<void (const common::UpdateInfo &)> gazebo::event::Events::worldUpdateBegin** [static]

World update has started.

Referenced by `ConnectWorldUpdateBegin()`.

10.44.3.16 **EventT<void ()> gazebo::event::Events::worldUpdateEnd** [static]

World update has ended.

Referenced by `ConnectWorldUpdateEnd()`, and `DisconnectWorldUpdateEnd()`.

The documentation for this class was generated from the following file:

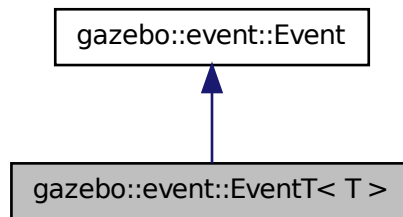
- **Events.hh**

10.45 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

- 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.

10.45.1 Detailed Description

```
template<typename T>class gazebo::event::EventT< T >
```

A class for event processing.

10.45.2 Member Function Documentation

10.45.2.1 `template<typename T> void gazebo::event::EventT< T >::operator()() [inline]`

Access the signal.

10.45.2.2 `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
----	----	---------------

10.45.2.3 `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

10.45.2.4 `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	_p1	the first parameter
in	_p2	the second parameter
in	_p3	the second parameter

10.45.2.5 `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

10.45.2.6 `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

10.45.2.7 `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

10.45.2.8 `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

10.45.2.9 `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

10.45.2.10 `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

10.45.2.11 `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

10.45.2.12 `template<typename T> void gazebo::event::EventT< T >::Signal () [inline]`

Signal the event for all subscribers.

Referenced by `gazebo::event::EventT< void(bool)>::operator()()`.

10.45.2.13 `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
----	-----------------	-----------

10.45.2.14 `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

10.45.2.15 `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

10.45.2.16 `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

10.45.2.17 `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

10.45.2.18 `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

10.45.2.19 `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

10.45.2.20 `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

10.45.2.21 `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

10.45.2.22 `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:

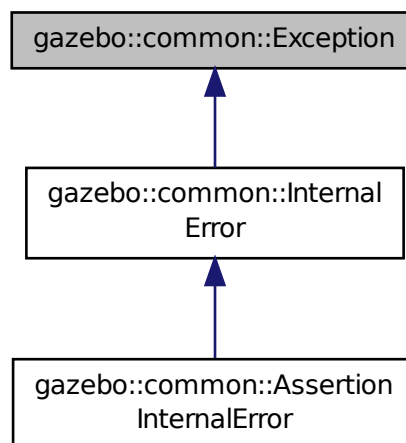
- **Event.hh**

10.46 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

10.46.1 Detailed Description

Class for generating exceptions.

10.46.2 Constructor & Destructor Documentation

10.46.2.1 gazebo::common::Exception::Exception ()

Constructor.

10.46.2.2 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

10.46.2.3 virtual gazebo::common::Exception::~~Exception () [virtual]

Destructor.

10.46.3 Member Function Documentation

10.46.3.1 `std::string gazebo::common::Exception::GetErrorFile () const`

Return the error function.

Returns

The error function name

10.46.3.2 `std::string gazebo::common::Exception::GetErrorStr () const`

Return the error string.

Returns

The error string

10.46.3.3 `void gazebo::common::Exception::Print () const`

Print the exception to std out.

10.46.4 Friends And Related Function Documentation

10.46.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::common::Exception & _err) [friend]`

stream insertion operator for Gazebo Error

Parameters

<i>in</i>	<i>_out</i>	the output stream
<i>in</i>	<i>_err</i>	the exception

The documentation for this class was generated from the following file:

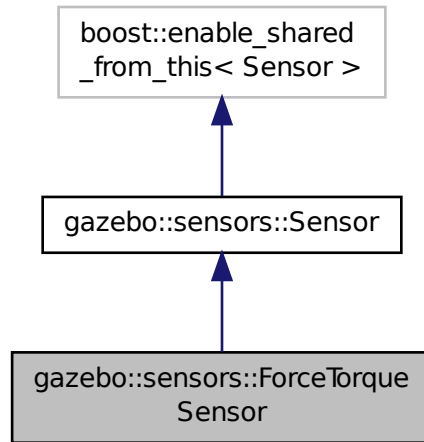
- **Exception.hh**

10.47 gazebo::sensors::ForceTorqueSensor Class Reference

Sensor (p. 751) 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.
- 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 void **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Protected Attributes

- **event::EventT**< void(msgs::WrenchStamped)> **update**
Update event.

10.47.1 Detailed Description

Sensor (p. 751) for measure force and torque on a joint.

10.47.2 Constructor & Destructor Documentation

10.47.2.1 gazebo::sensors::ForceTorqueSensor::ForceTorqueSensor ()

Constructor.

10.47.2.2 virtual gazebo::sensors::ForceTorqueSensor::~~ForceTorqueSensor () [virtual]

Destructor.

10.47.3 Member Function Documentation

10.47.3.1 template<typename T > event::ConnectionPtr gazebo::sensors::ForceTorqueSensor::ConnectUpdate (T _subscriber) [inline]

Connect a to the update signal.

Parameters

in	<code>_subscriber</code>	Callback function.
----	--------------------------	--------------------

Returns

The connection, which must be kept in scope.

References gazebo::event::EventT< T >::Connect(), and update.

10.47.3.2 void gazebo::sensors::ForceTorqueSensor::DisconnectUpdate (event::ConnectionPtr & _conn) [inline]

Disconnect from the update signal.

Parameters

in	<code>_conn</code>	Connection to remove.
----	--------------------	-----------------------

References gazebo::event::EventT< T >::Disconnect(), and update.

10.47.3.3 `virtual void gazebo::sensors::ForceTorqueSensor::Fini () [protected],[virtual]`

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 755).

10.47.3.4 `math::Vector3 gazebo::sensors::ForceTorqueSensor::GetForce () const`

Get the current joint force.

Returns

The latest measured force.

10.47.3.5 `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. 757).

10.47.3.6 `math::Vector3 gazebo::sensors::ForceTorqueSensor::GetTorque () const`

Get the current joint torque.

Returns

The latest measured torque.

10.47.3.7 `virtual void gazebo::sensors::ForceTorqueSensor::Init () [virtual]`

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 758).

10.47.3.8 `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. 758).

10.47.3.9 virtual void gazebo::sensors::ForceTorqueSensor::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. 759).

10.47.3.10 virtual void 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	<i>_force</i>	True if update is forced, false if not
----	---------------	--

Reimplemented from **gazebo::sensors::Sensor** (p. 760).

10.47.4 Member Data Documentation

10.47.4.1 event::EventT<void(msgs::WrenchStamped)> gazebo::sensors::ForceTorqueSensor::update [protected]

Update event.

Referenced by `ConnectUpdate()`, and `DisconnectUpdate()`.

The documentation for this class was generated from the following file:

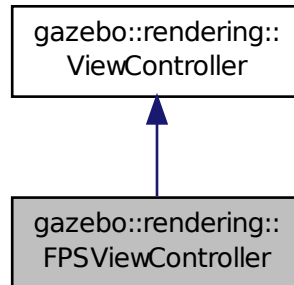
- **ForceTorqueSensor.hh**

10.48 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

10.48.1 Detailed Description

First Person Shooter style view controller.

10.48.2 Constructor & Destructor Documentation

10.48.2.1 gazebo::rendering::FPSViewController::FPSViewController (UserCameraPtr *_camera*)

Constructor.

Parameters

in	Camera (p. 179)	to controll
----	------------------------	-------------

10.48.2.2 virtual gazebo::rendering::FPSViewController::~~FPSViewController () [virtual]

Destructor.

10.48.3 Member Function Documentation

10.48.3.1 static std::string gazebo::rendering::FPSViewController::GetTypeString () [static]

Get the type name of this view controller.

Returns

The name of the controller type: "fps"

10.48.3.2 void gazebo::rendering::FPSViewController::HandleKeyPressEvent (const std::string & *_key*) [virtual]

Handle a key press event.

Parameters

in	<i>_key</i>	The key that was pressed.
----	-------------	---------------------------

Implements **gazebo::rendering::ViewController** (p. 1032).

10.48.3.3 void gazebo::rendering::FPSViewController::HandleKeyReleaseEvent (const std::string & *_key*) [virtual]

Handle a key release event.

Parameters

in	<i>_key</i>	The key that was released.
----	-------------	----------------------------

Implements **gazebo::rendering::ViewController** (p. 1033).

10.48.3.4 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. 1033).

10.48.3.5 `virtual void gazebo::rendering::FPSViewController::Init () [virtual]`

Initialize the controller.

Implements `gazebo::rendering::ViewController` (p. 1033).

10.48.3.6 `virtual void gazebo::rendering::FPSViewController::Update () [virtual]`

Update the camera position.

Implements `gazebo::rendering::ViewController` (p. 1034).

The documentation for this class was generated from the following file:

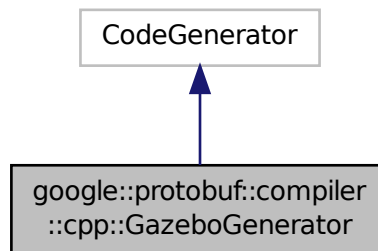
- `FPSViewController.hh`

10.49 google::protobuf::compiler::cpp::GazeboGenerator Class Reference

Google protobuf message generator for `gazebo::msgs` (p. 102).

```
#include <GazeboGenerator.hh>
```

Inheritance diagram for `google::protobuf::compiler::cpp::GazeboGenerator`:



Public Member Functions

- `GazeboGenerator` (const std::string &_name)
- virtual `~GazeboGenerator` ()
- virtual bool `Generate` (const FileDescriptor *file, const string ¶meter, OutputDirectory *directory, string *error) const

10.49.1 Detailed Description

Google protobuf message generator for `gazebo::msgs` (p. 102).

10.49.2 Constructor & Destructor Documentation

10.49.2.1 `google::protobuf::compiler::cpp::GazeboGenerator::GazeboGenerator (const std::string & _name)`

10.49.2.2 `virtual google::protobuf::compiler::cpp::GazeboGenerator::~GazeboGenerator () [virtual]`

10.49.3 Member Function Documentation

10.49.3.1 `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:

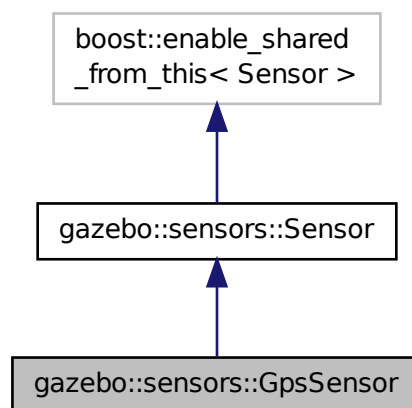
- `GazeboGenerator.hh`

10.50 gazebo::sensors::GpsSensor Class Reference

`GpsSensor` (p. 341) 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)
- virtual void **Load** (const std::string &_worldName)

Load the sensor with default parameters.

Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

This gets overwritten by derived sensor types.

Additional Inherited Members

10.50.1 Detailed Description

GpsSensor (p. 341) to provide position measurement.

10.50.2 Constructor & Destructor Documentation

10.50.2.1 gazebo::sensors::GpsSensor::GpsSensor ()

Constructor.

10.50.2.2 virtual gazebo::sensors::GpsSensor::~~GpsSensor () [virtual]

Destructor.

10.50.3 Member Function Documentation

10.50.3.1 virtual void gazebo::sensors::GpsSensor::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 755).

10.50.3.2 `double gazebo::sensors::GpsSensor::GetAltitude () const`

Accessor for current altitude.

Returns

Current altitude above sea level.

10.50.3.3 `math::Angle gazebo::sensors::GpsSensor::GetLatitude () const`

Accessor for current latitude angle.

Returns

Current latitude angle.

10.50.3.4 `math::Angle gazebo::sensors::GpsSensor::GetLongitude () const`

Accessor for current longitude angle.

Returns

Current longitude angle.

10.50.3.5 `virtual void gazebo::sensors::GpsSensor::Init () [virtual]`

Initialize the sensor.

Reimplemented from **`gazebo::sensors::Sensor`** (p. 758).

10.50.3.6 `virtual void gazebo::sensors::GpsSensor::Load (const std::string & _worldName, sdf::ElementPtr & _sdf) [virtual]`

10.50.3.7 `virtual void gazebo::sensors::GpsSensor::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. 759).

10.50.3.8 `virtual void 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
----	---------------------	--

Reimplemented from `gazebo::sensors::Sensor` (p. 760).

The documentation for this class was generated from the following file:

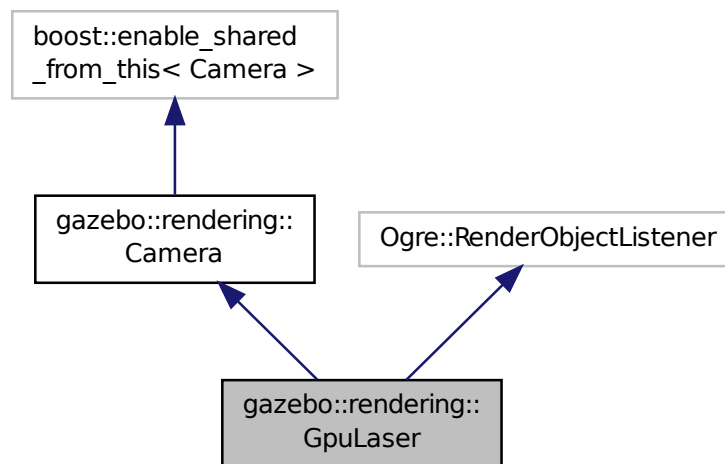
- `GpsSensor.hh`

10.51 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)
- 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 (equivalent 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

10.51.1 Detailed Description

GPU based laser distance sensor.

10.51.2 Constructor & Destructor Documentation

10.51.2.1 `gazebo::rendering::GpuLaser::GpuLaser (const std::string & _namePrefix, ScenePtr _scene, bool _autoRender = true)`

Constructor.

Parameters

in	<code>_namePrefix</code>	Unique prefix name for the camera.
in	<code>_scene</code>	Scene (p. 728) that will contain the camera
in	<code>_autoRender</code>	Almost everyone should leave this as true.

10.51.2.2 `virtual gazebo::rendering::GpuLaser::~GpuLaser () [virtual]`

Destructor.

10.51.3 Member Function Documentation

10.51.3.1 `template<typename T > event::ConnectionPtr gazebo::rendering::GpuLaser::ConnectNewLaserFrame (T _subscriber) [inline]`

Connect to a laser frame signal.

Parameters

in	<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.

References `gazebo::event::EventT< T >::Connect()`.

10.51.3.2 `void gazebo::rendering::GpuLaser::CreateLaserTexture (const std::string & _textureName)`

Create the texture which is used to render laser data.

Parameters

in	<code>_textureName</code>	Name of the new texture.
----	---------------------------	--------------------------

10.51.3.3 `void gazebo::rendering::GpuLaser::DisconnectNewLaserFrame (event::ConnectionPtr & _c) [inline]`

Disconnect from a laser frame signal.

Parameters

in	<code>_c</code>	The connection to disconnect
----	-----------------	------------------------------

References gazebo::event::EventT< T >::Disconnect().

10.51.3.4 `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. 188).

10.51.3.5 `double gazebo::rendering::GpuLaser::GetCameraCount () const`

Get the number of cameras required.

Returns

Number of cameras needed to generate the rays

10.51.3.6 `double gazebo::rendering::GpuLaser::GetCosHorzFOV () const`

Get Cos Horz field-of-view.

Returns

$2 * \text{atan}(\tan(\text{this->hfov}/2) / \cos(\text{this->vfov}/2))$

10.51.3.7 `double gazebo::rendering::GpuLaser::GetCosVertFOV () const`

Get Cos Vert field-of-view.

Returns

$2 * \text{atan}(\tan(\text{this->vfov}/2) / \cos(\text{this->hfov}/2))$

10.51.3.8 `double gazebo::rendering::GpuLaser::GetFarClip () const`

Get far clip.

Returns

far clip distance

10.51.3.9 `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.

10.51.3.10 `double gazebo::rendering::GpuLaser::GetHorzHalfAngle () const`

Get $(\text{horizontal_max_angle} + \text{horizontal_min_angle}) * 0.5$.

Returns

$(\text{horizontal_max_angle} + \text{horizontal_min_angle}) * 0.5$

10.51.3.11 `const float* gazebo::rendering::GpuLaser::GetLaserData ()`

All things needed to get back z buffer for laser data.

Returns

Array of laser data.

10.51.3.12 `double gazebo::rendering::GpuLaser::GetNearClip () const`

Get near clip.

Returns

near clip distance

10.51.3.13 `double gazebo::rendering::GpuLaser::GetRayCountRatio () const`

Get the ray count ratio (equivalent to aspect ratio)

Returns

The ray count ratio (equivalent to aspect ratio)

10.51.3.14 `double gazebo::rendering::GpuLaser::GetVertFOV () const`

Get the vertical field-of-view.

Returns

The vertical field of view of the laser sensor.

10.51.3.15 `double gazebo::rendering::GpuLaser::GetVertHalfAngle () const`

Get $(\text{vertical_max_angle} + \text{vertical_min_angle}) * 0.5$.

Returns

$(\text{vertical_max_angle} + \text{vertical_min_angle}) * 0.5$

10.51.3.16 virtual void gazebo::rendering::GpuLaser::Init () [virtual]

Initialize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 196).

10.51.3.17 bool gazebo::rendering::GpuLaser::IsHorizontal () const

Gets if sensor is horizontal.

Returns

True if horizontal, false if not

10.51.3.18 virtual void gazebo::rendering::GpuLaser::Load (sdf::ElementPtr & _sdf) [virtual]

10.51.3.19 virtual void gazebo::rendering::GpuLaser::Load () [virtual]

Load the camera with default parameters.

Reimplemented from **gazebo::rendering::Camera** (p. 197).

10.51.3.20 virtual void gazebo::rendering::GpuLaser::notifyRenderSingleObject (Ogre::Renderable * _rend, const Ogre::Pass * _p, const Ogre::AutoParamDataSource * _s, const Ogre::LightList * _ll, bool _supp) [virtual]

10.51.3.21 virtual void gazebo::rendering::GpuLaser::PostRender () [virtual]

Post render.

Called after the render signal.

Reimplemented from **gazebo::rendering::Camera** (p. 197).

10.51.3.22 void gazebo::rendering::GpuLaser::SetCameraCount (double _cameraCount)

Set the number of cameras required.

Parameters

in	<code>_cameraCount</code>	The number of cameras required to generate the rays
----	---------------------------	---

10.51.3.23 void gazebo::rendering::GpuLaser::SetCosHorzFOV (double _chfov)

Set the Cos Horz FOV.

Parameters

in	<code>_chfov</code>	Cos Horz FOV
----	---------------------	--------------

10.51.3.24 void gazebo::rendering::GpuLaser::SetCosVertFOV (double *_cvfov*)

Set the Cos Horz FOV.

Parameters

in	<i>_cvfov</i>	Cos Horz FOV
----	---------------	--------------

10.51.3.25 void gazebo::rendering::GpuLaser::SetFarClip (double *_far*)

Set the far clip distance.

Parameters

in	<i>_far</i>	far clip distance
----	-------------	-------------------

10.51.3.26 void gazebo::rendering::GpuLaser::SetHorzFOV (double *_hfov*)

Set the horizontal fov.

Parameters

in	<i>_hfov</i>	horizontal fov
----	--------------	----------------

10.51.3.27 void gazebo::rendering::GpuLaser::SetHorzHalfAngle (double *_angle*)

Set the horizontal half angle.

Parameters

in	<i>_angle</i>	horizontal half angle
----	---------------	-----------------------

10.51.3.28 void gazebo::rendering::GpuLaser::SetIsHorizontal (bool *_horizontal*)

Set sensor horizontal or vertical.

Parameters

in	<i>_horizontal</i>	True if horizontal, false if not
----	--------------------	----------------------------------

10.51.3.29 void gazebo::rendering::GpuLaser::SetNearClip (double *_near*)

Set the near clip distance.

Parameters

in	<i>_near</i>	near clip distance
----	--------------	--------------------

10.51.3.30 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

10.51.3.31 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)
----	-----------------------	--

10.51.3.32 void gazebo::rendering::GpuLaser::SetVertFOV (double *_vfov*)

Set the vertical fov.

Parameters

in	<i>_vfov</i>	vertical fov
----	--------------	--------------

10.51.3.33 void gazebo::rendering::GpuLaser::SetVertHalfAngle (double *_angle*)

Set the vertical half angle.

Parameters

in	<i>_angle</i>	vertical half angle
----	---------------	---------------------

10.51.4 Member Data Documentation

10.51.4.1 unsigned int gazebo::rendering::GpuLaser::cameraCount [protected]

Number of cameras needed to generate the rays.

10.51.4.2 double gazebo::rendering::GpuLaser::chfov [protected]

Cos horizontal field-of-view.

10.51.4.3 double gazebo::rendering::GpuLaser::cvfov [protected]

Cos vertical field-of-view.

10.51.4.4 `double gazebo::rendering::GpuLaser::far` [protected]

Far clip plane.

10.51.4.5 `double gazebo::rendering::GpuLaser::hfov` [protected]

Horizontal field-of-view.

10.51.4.6 `double gazebo::rendering::GpuLaser::horzHalfAngle` [protected]

Horizontal half angle.

10.51.4.7 `bool gazebo::rendering::GpuLaser::isHorizontal` [protected]

True if the sensor is horizontal only.

10.51.4.8 `double gazebo::rendering::GpuLaser::near` [protected]

Near clip plane.

10.51.4.9 `double gazebo::rendering::GpuLaser::rayCountRatio` [protected]

Ray count ratio.

10.51.4.10 `double gazebo::rendering::GpuLaser::vertHalfAngle` [protected]

Vertical half angle.

10.51.4.11 `double gazebo::rendering::GpuLaser::vfov` [protected]

Vertical field-of-view.

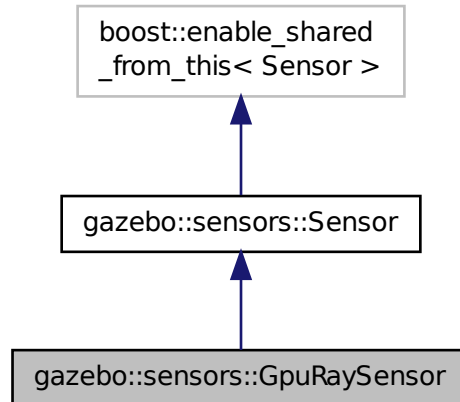
The documentation for this class was generated from the following file:

- **GpuLaser.hh**

10.52 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. 344).*
- 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.
- 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 void **UpdateImpl** (bool _force)

Update the sensor information.

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 elementz.
- sdf::ElementPtr **vertElem**

Vertical SDF element.
- unsigned int **vertRangeCount**

Vertical range count.
- unsigned int **vertRayCount**

Vertical ray count.

10.52.1 Constructor & Destructor Documentation

10.52.1.1 gazebo::sensors::GpuRaySensor::GpuRaySensor ()

Constructor.

10.52.1.2 `virtual gazebo::sensors::GpuRaySensor::~~GpuRaySensor () [virtual]`

Destructor.

10.52.2 Member Function Documentation

10.52.2.1 `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.
-----------------	--------------------------	-----------------

10.52.2.2 `void gazebo::sensors::GpuRaySensor::DisconnectNewLaserFrame (event::ConnectionPtr & _conn)`

Disconnect Laser Frame.

Parameters

<code>in, out</code>	<code>_conn</code>	Connection pointer to disconnect.
----------------------	--------------------	-----------------------------------

10.52.2.3 `virtual void gazebo::sensors::GpuRaySensor::Fini () [protected],[virtual]`

Finalize the ray.

Reimplemented from `gazebo::sensors::Sensor` (p. 755).

10.52.2.4 `math::Angle gazebo::sensors::GpuRaySensor::GetAngleMax () const`

Get the maximum angle.

Returns

the maximum angle

10.52.2.5 `math::Angle gazebo::sensors::GpuRaySensor::GetAngleMin () const`

Get the minimum angle.

Returns

The minimum angle

10.52.2.6 `double gazebo::sensors::GpuRaySensor::GetAngleResolution () const`

Get radians between each range.

10.52.2.7 unsigned int gazebo::sensors::GpuRaySensor::GetCameraCount () const

Gets the camera count.

Returns

Number of cameras

10.52.2.8 double gazebo::sensors::GpuRaySensor::GetCosHorzFOV () const

Get Cos Horz field-of-view.

Returns

$2 * \text{atan}(\tan(\text{this->hfov}/2) / \cos(\text{this->vfov}/2))$

10.52.2.9 double gazebo::sensors::GpuRaySensor::GetCosVertFOV () const

Get Cos Vert field-of-view.

Returns

$2 * \text{atan}(\tan(\text{this->vfov}/2) / \cos(\text{this->hfov}/2))$

10.52.2.10 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

<code>in</code>	<code>_index</code>	Index of specific ray
-----------------	---------------------	-----------------------

Returns

Fiducial value of ray

10.52.2.11 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.

10.52.2.12 `double gazebo::sensors::GpuRaySensor::GetHorzHalfAngle () const`

Get $(\text{horizontal_max_angle} + \text{horizontal_min_angle}) * 0.5$.

Returns

$(\text{horizontal_max_angle} + \text{horizontal_min_angle}) * 0.5$

10.52.2.13 `rendering::GpuLaserPtr gazebo::sensors::GpuRaySensor::GetLaserCamera () const` `[inline]`

Returns a pointer to the internally kept `rendering::GpuLaser` (p. 344).

Returns

Pointer to GpuLaser

10.52.2.14 `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.

10.52.2.15 `int gazebo::sensors::GpuRaySensor::GetRangeCount () const`

Get the range count.

Returns

The number of ranges

10.52.2.16 `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.

10.52.2.17 `double gazebo::sensors::GpuRaySensor::GetRangeMax () const`

Get the maximum range.

Returns

The maximum range

10.52.2.18 `double gazebo::sensors::GpuRaySensor::GetRangeMin () const`

Get the minimum range.

Returns

The minimum range

10.52.2.19 `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

10.52.2.20 `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
-----	--------	---

10.52.2.21 `int gazebo::sensors::GpuRaySensor::GetRayCount () const`

Get the ray count.

Returns

The number of rays

10.52.2.22 `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.

10.52.2.23 `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

10.52.2.24 `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. 757).

10.52.2.25 `double gazebo::sensors::GpuRaySensor::GetVertFOV () const`

Get the vertical field-of-view.

10.52.2.26 `double gazebo::sensors::GpuRaySensor::GetVertHalfAngle () const`

Get $(\text{vertical_max_angle} + \text{vertical_min_angle}) * 0.5$.

Returns

$(\text{vertical_max_angle} + \text{vertical_min_angle}) * 0.5$

10.52.2.27 `math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMax () const`

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.52.2.28 `math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMin () const`

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.52.2.29 `int gazebo::sensors::GpuRaySensor::GetVerticalRangeCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.52.2.30 `int gazebo::sensors::GpuRaySensor::GetVerticalRayCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.52.2.31 `virtual void gazebo::sensors::GpuRaySensor::Init () [virtual]`

Initialize the ray.

Reimplemented from `gazebo::sensors::Sensor` (p. 758).

10.52.2.32 `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. 758).

10.52.2.33 `bool gazebo::sensors::GpuRaySensor::IsHorizontal () const`

Gets if sensor is horizontal.

Returns

True if horizontal, false if not

10.52.2.34 virtual void gazebo::sensors::GpuRaySensor::Load (const std::string & *_worldName*, sdf::ElementPtr & *_sdf*)
[virtual]

Load the sensor with SDF parameters.

Parameters

in	<i>_sdf</i>	SDF Sensor (p. 751) parameters
in	<i>_worldName</i>	Name of world to load from

10.52.2.35 virtual void gazebo::sensors::GpuRaySensor::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. 759).

10.52.2.36 void gazebo::sensors::GpuRaySensor::SetAngleMax (double *_angle*)

Set the scan maximum angle.

Parameters

in	<i>_angle</i>	The maximum angle
----	---------------	-------------------

10.52.2.37 void gazebo::sensors::GpuRaySensor::SetAngleMin (double *_angle*)

Set the scan minimum angle.

Parameters

in	<i>_angle</i>	The minimum angle
----	---------------	-------------------

10.52.2.38 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
----	---------------	-------------------------------------

10.52.2.39 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
----	---------------	-------------------------------------

10.52.2.40 virtual void gazebo::sensors::GpuRaySensor::UpdateImpl (bool *_force*) [protected],[virtual]

Update the sensor information.

Parameters

in	<i>_force</i>	True if update is forced, false if not
----	---------------	--

Reimplemented from **gazebo::sensors::Sensor** (p. 760).

10.52.3 Member Data Documentation

10.52.3.1 sdf::ElementPtr gazebo::sensors::GpuRaySensor::cameraElem [protected]

Camera SDF element.

10.52.3.2 sdf::ElementPtr gazebo::sensors::GpuRaySensor::horzElem [protected]

Horizontal SDF element.

10.52.3.3 unsigned int gazebo::sensors::GpuRaySensor::horzRangeCount [protected]

Horizontal range count.

10.52.3.4 unsigned int gazebo::sensors::GpuRaySensor::horzRayCount [protected]

Horizontal ray count.

10.52.3.5 double gazebo::sensors::GpuRaySensor::rangeCountRatio [protected]

Range count ratio.

10.52.3.6 sdf::ElementPtr gazebo::sensors::GpuRaySensor::rangeElem [protected]

Range SDF element.

10.52.3.7 sdf::ElementPtr gazebo::sensors::GpuRaySensor::scanElem [protected]

Scan SDF element.

10.52.3.8 sdf::ElementPtr gazebo::sensors::GpuRaySensor::vertElem [protected]

Vertical SDF element.

10.52.3.9 unsigned int gazebo::sensors::GpuRaySensor::vertRangeCount [protected]

Vertical range count.

10.52.3.10 unsigned int gazebo::sensors::GpuRaySensor::vertRayCount [protected]

Vertical ray count.

The documentation for this class was generated from the following file:

- **GpuRaySensor.hh**

10.53 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. 123) 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.

10.53.1 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.

10.53.2 Constructor & Destructor Documentation

10.53.2.1 gazebo::rendering::Grid::Grid (Scene * *_scene*, uint32_t *_cellCount*, float *_cellLength*, float *_lineWidth*, const common::Color & *_color*)

Constructor.

Parameters

in	<i>_scene</i>	The scene this object is part of
in	<i>_cellCount</i>	The number of cells to draw
in	<i>_cellLength</i>	The size of each cell
in	<i>_lineWidth</i>	The width of the lines to use
in	<i>_color</i>	The color of the grid

10.53.2.2 gazebo::rendering::Grid::~~Grid ()

Destructor.

10.53.3 Member Function Documentation

10.53.3.1 void gazebo::rendering::Grid::Enable (bool *_enable*)

Enable or disable the grid.

Parameters

in	<i>_enable</i>	Set to true to view the grid, false to make invisible.
----	----------------	--

10.53.3.2 uint32_t gazebo::rendering::Grid::GetCellCount () const [inline]

Get the number of cells.

10.53.3.3 float gazebo::rendering::Grid::GetCellLength () const [inline]

Get the cell length.

Returns

The cell length

10.53.3.4 `common::Color gazebo::rendering::Grid::GetColor () const` `[inline]`

Return the grid color.

Returns

The grid color

10.53.3.5 `uint32_t gazebo::rendering::Grid::GetHeight () const` `[inline]`

Get the height of the grid.

Returns

The height

10.53.3.6 `float gazebo::rendering::Grid::GetLineWidth () const` `[inline]`

Get the width of the grid line.

Returns

The line width

10.53.3.7 `Ogre::SceneNode* gazebo::rendering::Grid::GetSceneNode ()` `[inline]`

Get the **Ogre** (p. 123) scene node associated with this grid.

Returns

The **Ogre** (p. 123) scene node associated with this grid

10.53.3.8 `void gazebo::rendering::Grid::Init ()`

Initialize the grid.

10.53.3.9 `void gazebo::rendering::Grid::SetCellCount (uint32_t _count)`

Set the number of cells.

Parameters

<code>in</code>	<code>_count</code>	The number of cells
-----------------	---------------------	---------------------

10.53.3.10 void gazebo::rendering::Grid::SetCellLength (float *_len*)

Set the cell length.

Parameters

in	<i>_len</i>	The cell length
----	-------------	-----------------

10.53.3.11 void gazebo::rendering::Grid::SetColor (const common::Color & *_color*)

Sets the color of the grid.

Parameters

in	<i>_color</i>	The grid color
----	---------------	----------------

10.53.3.12 void gazebo::rendering::Grid::SetHeight (uint32_t *_count*)

Set the height of the grid.

Parameters

in	<i>_count</i>	Grid (p. 365) height
----	---------------	-----------------------------

10.53.3.13 void gazebo::rendering::Grid::SetLineWidth (float *_width*)

Set the line width.

Parameters

in	<i>_width</i>	The width of the grid
----	---------------	-----------------------

10.53.3.14 void gazebo::rendering::Grid::SetUserData (const Ogre::Any & *_data*)

Sets user data on all ogre objects we own.

Parameters

in	<i>_data</i>	The user data
----	--------------	---------------

The documentation for this class was generated from the following file:

- **Grid.hh**

10.54 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.

10.54.1 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.

10.54.2 Constructor & Destructor Documentation

10.54.2.1 gazebo::physics::Gripper::Gripper (**ModelPtr** _model) [explicit]

Constructor.

Parameters

in	_model	The model which contains the Gripper (p. 368).
----	--------	---

10.54.2.2 virtual gazebo::physics::Gripper::~~Gripper () [virtual]

Destructor.

10.54.3 Member Function Documentation

10.54.3.1 std::string gazebo::physics::Gripper::GetName () const

Return the name of the gripper.

10.54.3.2 `virtual void gazebo::physics::Gripper::Init () [virtual]`

Initialize.

10.54.3.3 `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.

10.54.3.4 `virtual void gazebo::physics::Gripper::Load (sdf::ElementPtr _sdf) [virtual]`

Load the gripper.

Parameters

in	_sdf	Shared point to an sdf element that contains the list of links in the gripper.
----	------	--

10.54.4 Member Data Documentation

10.54.4.1 `transport::NodePtr gazebo::physics::Gripper::node [protected]`

Node for communication.

The documentation for this class was generated from the following file:

- **Gripper.hh**

10.55 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. 179) to and overlay window.*
- bool **AttachCameraToImage** (**DepthCameraPtr** &_camera, const std::string &_windowName)
*Use this function to draw the output from a **rendering::DepthCamera** (p. 272) 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.

10.55.1 Detailed Description

A class that creates a CEGUI overlay on a render window.

10.55.2 Constructor & Destructor Documentation

10.55.2.1 gazebo::rendering::GUIOverlay::GUIOverlay ()

Constructor.

10.55.2.2 virtual gazebo::rendering::GUIOverlay::~GUIOverlay () [virtual]

Destructor.

10.55.3 Member Function Documentation

10.55.3.1 bool gazebo::rendering::GUIOverlay::AttachCameraToImage (CameraPtr &_camera, const std::string &_windowName)

Use this function to draw the output from a **rendering::Camera** (p. 179) to and overlay window.

Parameters

in	<code>_camera</code>	Pointer to the camera.
in	<code>_windowName</code>	Name of the window to receive the camera image

Returns

True if successful

10.55.3.2 `bool gazebo::rendering::GUIOverlay::AttachCameraToImage (DepthCameraPtr & _camera, const std::string & _windowName)`

Use this function to draw the output from a **rendering::DepthCamera** (p. 272) to and overlay window.

Parameters

in	<code>_camera</code>	Pointer to the camera.
in	<code>_windowName</code>	Name of the window to receive the camera image

Returns

True if successful

10.55.3.3 `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> .

10.55.3.4 `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.

10.55.3.5 bool gazebo::rendering::GUIOverlay::HandleKeyPressEvent (const std::string & *_key*)

Handle a key press event.

Parameters

<i>in</i>	<i>_key</i>	The key pressed.
-----------	-------------	------------------

Returns

True if the key press event was handled.

10.55.3.6 bool gazebo::rendering::GUIOverlay::HandleKeyReleaseEvent (const std::string & *_key*)

Handle a key release event.

Parameters

<i>in</i>	<i>_key</i>	The key released.
-----------	-------------	-------------------

Returns

True if the key release event was handled.

10.55.3.7 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.

10.55.3.8 void gazebo::rendering::GUIOverlay::Hide ()

Make the overlay invisible.

10.55.3.9 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.
-----------	----------------------	--

10.55.3.10 `bool gazebo::rendering::GUIOverlay::IsInitialized ()`

Return true if the overlay has been initialized.

Returns

True if initialized

10.55.3.11 `void gazebo::rendering::GUIOverlay::LoadLayout (const std::string & _filename)`

Load a CEGUI layout file.

Parameters

<code>in</code>	<code>_filename</code>	Name of the layout file.
-----------------	------------------------	--------------------------

10.55.3.12 `void gazebo::rendering::GUIOverlay::Resize (unsigned int _width, unsigned int _height)`

Resize the window.

10.55.3.13 `void gazebo::rendering::GUIOverlay::Show ()`

Make the overlay visible.

10.55.3.14 `void gazebo::rendering::GUIOverlay::Update ()`

Update the overlay's objects.

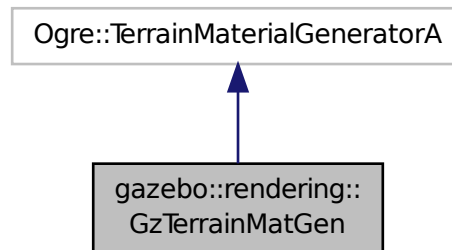
The documentation for this class was generated from the following file:

- **GUIOverlay.hh**

10.56 gazebo::rendering::GzTerrainMatGen Class Reference

```
#include <Heightmap.hh>
```

Inheritance diagram for gazebo::rendering::GzTerrainMatGen:



Classes

- class **SM2Profile**
Shader model 2 profile target.

Public Member Functions

- **GzTerrainMatGen** ()
Constructor.
- virtual **~GzTerrainMatGen** ()
Destructor.

10.56.1 Constructor & Destructor Documentation

10.56.1.1 gazebo::rendering::GzTerrainMatGen::GzTerrainMatGen ()

Constructor.

10.56.1.2 virtual gazebo::rendering::GzTerrainMatGen::~~GzTerrainMatGen () [virtual]

Destructor.

The documentation for this class was generated from the following file:

- **Heightmap.hh**

10.57 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.
- 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.

Static Public Attributes

- static const unsigned int **NumTerrainSubdivisions**
Number of pieces in which a terrain is subdivided for paging.

10.57.1 Detailed Description

Rendering a terrain using heightmap information.

10.57.2 Constructor & Destructor Documentation

10.57.2.1 gazebo::rendering::Heightmap::Heightmap (ScenePtr *_scene*)

Constructor.

Parameters

in	<i>_scene</i>	Pointer to the scene that will contain the heightmap
----	---------------	--

10.57.2.2 virtual gazebo::rendering::Heightmap::~Heightmap () [virtual]

Destructor.

10.57.3 Member Function Documentation

10.57.3.1 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. 179) 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

10.57.3.2 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.

10.57.3.3 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

10.57.3.4 `common::Image gazebo::rendering::Heightmap::GetImage () const`

Get the heightmap as an image.

Returns

An image that contains the terrain data.

10.57.3.5 `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. 179) 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.

10.57.3.6 `Ogre::TerrainGroup* gazebo::rendering::Heightmap::GetOgreTerrain () const`

Get a pointer to the OGRE terrain group object.

Returns

Pointer to the OGRE terrain.

10.57.3.7 `void gazebo::rendering::Heightmap::Load ()`

Load the heightmap.

10.57.3.8 `void gazebo::rendering::Heightmap::LoadFromMsg (ConstVisualPtr & _msg)`

Load the heightmap from a visual message.

Parameters

<code>in</code>	<code>_msg</code>	The visual message containing heightmap info
-----------------	-------------------	--

10.57.3.9 `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

<code>in</code>	<code>_camera</code>	Camera (p. 179) associated with the mouse press.
<code>in</code>	<code>_mousePos</code>	Position of the mouse in viewport coordinates.
<code>in</code>	<code>_outsideRadius</code>	Controls the radius of effect.
<code>in</code>	<code>_insideRadius</code>	Controls the size of the radius with the maximum effect (value between 0 and 1).
<code>in</code>	<code>_weight</code>	Controls modification magnitude.

Returns

True if the terrain was modified

10.57.3.10 `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

<code>in</code>	<code>_camera</code>	Camera (p. 179) associated with the mouse press.
<code>in</code>	<code>_mousePos</code>	Position of the mouse in viewport coordinates.
<code>in</code>	<code>_outsideRadius</code>	Controls the radius of effect.
<code>in</code>	<code>_insideRadius</code>	Controls the size of the radius with the maximum effect (value between 0 and 1).
<code>in</code>	<code>_weight</code>	Controls modification magnitude.

Returns

True if the terrain was modified

10.57.3.11 `void gazebo::rendering::Heightmap::SetWireframe (bool _show)`

Set the heightmap to render in wireframe mode.

Parameters

<code>in</code>	<code>_show</code>	True to render wireframe, false to render solid.
-----------------	--------------------	--

10.57.3.12 `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

<code>in</code>	<code>_camera</code>	Camera (p. 179) 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

10.57.3.13 `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.

10.57.4 Member Data Documentation

10.57.4.1 `const unsigned int gazebo::rendering::Heightmap::NumTerrainSubdivisions` `[static]`

Number of pieces in which a terrain is subdivided for paging.

The documentation for this class was generated from the following file:

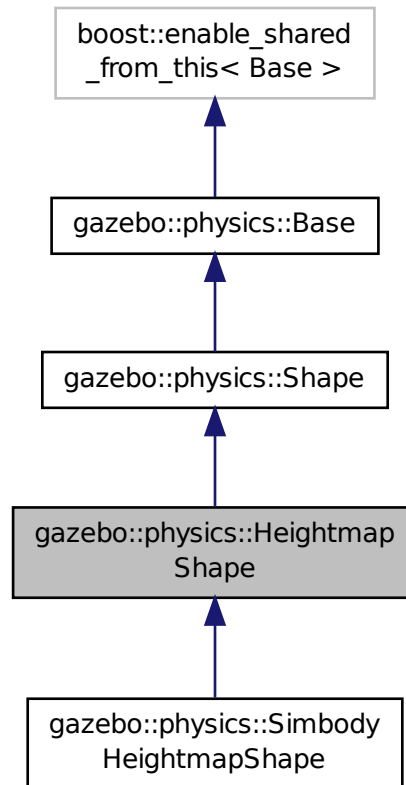
- **Heightmap.hh**

10.58 gazebo::physics::HeightmapShape Class Reference

HeightmapShape (p. 380) 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.
- **std::vector< float > heights**
Lookup table of heights.
- **common::Image 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

10.58.1 Detailed Description

HeightmapShape (p. 380) 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.

10.58.2 Constructor & Destructor Documentation

10.58.2.1 gazebo::physics::HeightmapShape::HeightmapShape (CollisionPtr _parent) [explicit]

Constructor.

Parameters

<code>in</code>	<code>_parent</code>	Parent Collision (p. 213) object.
-----------------	----------------------	--

10.58.2.2 virtual gazebo::physics::HeightmapShape::~~HeightmapShape () [virtual]

Destructor.

10.58.3 Member Function Documentation

10.58.3.1 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. 777).

10.58.3.2 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.

10.58.3.3 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.

10.58.3.4 float gazebo::physics::HeightmapShape::GetMaxHeight () const

Get the maximum height.

Returns

The maximum height.

10.58.3.5 float gazebo::physics::HeightmapShape::GetMinHeight () const

Get the minimum height.

Returns

The minimum height.

10.58.3.6 `math::Vector3 gazebo::physics::HeightmapShape::GetPos () const`

Get the origin in world coordinate frame.

Returns

The origin in world coordinate frame.

10.58.3.7 `math::Vector3 gazebo::physics::HeightmapShape::GetSize () const`

Get the size in meters.

Returns

The size in meters.

10.58.3.8 `int gazebo::physics::HeightmapShape::GetSubSampling () const`

Get the amount of subsampling.

Returns

Amount of subsampling.

10.58.3.9 `std::string gazebo::physics::HeightmapShape::GetURI () const`

Get the URI of the heightmap image.

Returns

The heightmap image URI.

10.58.3.10 `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. 996), result.x = width, result.y = length/height.

10.58.3.11 `virtual void gazebo::physics::HeightmapShape::Init () [virtual]`

Initialize the heightmap.

Implements **gazebo::physics::Shape** (p. 777).

Reimplemented in **gazebo::physics::SimbodyHeightmapShape** (p. 792).

10.58.3.12 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. 161).

10.58.3.13 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. 778).

10.58.3.14 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. 778).

10.58.4 Member Data Documentation

10.58.4.1 bool gazebo::physics::HeightmapShape::flipY [protected]

True to flip the heights along the y direction.

10.58.4.2 std::vector<float> gazebo::physics::HeightmapShape::heights [protected]

Lookup table of heights.

10.58.4.3 common::Image gazebo::physics::HeightmapShape::img [protected]

Image used to generate the heights.

10.58.4.4 int gazebo::physics::HeightmapShape::subSampling [protected]

The amount of subsampling. Default is 2.

10.58.4.5 unsigned int gazebo::physics::HeightmapShape::vertSize [protected]

Size of the height lookup table.

The documentation for this class was generated from the following file:

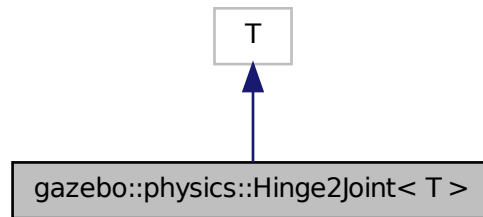
- **HeightmapShape.hh**

10.59 gazebo::physics::Hinge2Joint< T > Class Template Reference

A two axis hinge joint.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::Hinge2Joint< T >:



Public Member Functions

- **Hinge2Joint** (BasePtr _parent)
Constructor.
- virtual ~**Hinge2Joint** ()
Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the joint.

10.59.1 Detailed Description

```
template<class T>class gazebo::physics::Hinge2Joint< T >
```

A two axis hinge joint.

10.59.2 Constructor & Destructor Documentation

10.59.2.1 `template<class T> gazebo::physics::Hinge2Joint< T >::Hinge2Joint (BasePtr _parent) [inline], [explicit]`

Constructor.

Parameters

in	<i>_parent</i>	Parent link.
----	----------------	--------------

10.59.2.2 `template<class T> virtual gazebo::physics::Hinge2Joint< T >::~~Hinge2Joint () [inline], [virtual]`

Destructor.

10.59.3 Member Function Documentation

10.59.3.1 `template<class T> virtual unsigned int gazebo::physics::Hinge2Joint< T >::GetAngleCount () const [inline], [virtual]`

10.59.3.2 `template<class T> virtual void gazebo::physics::Hinge2Joint< T >::Load (sdf::ElementPtr _sdf) [inline], [virtual]`

Load the joint.

Parameters

in	<i>_sdf</i>	SDF values to load from.
----	-------------	--------------------------

Reimplemented in `gazebo::physics::SimbodyHinge2Joint` (p. 797).

The documentation for this class was generated from the following file:

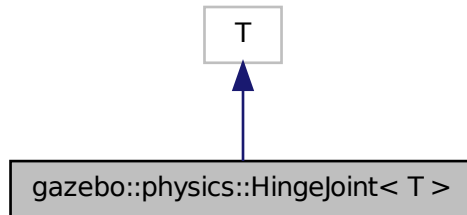
- `Hinge2Joint.hh`

10.60 gazebo::physics::HingeJoint< T > Class Template Reference

A single axis hinge joint.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::HingeJoint< T >:



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.

10.60.1 Detailed Description

```
template<class T>class gazebo::physics::HingeJoint< T >
```

A single axis hinge joint.

10.60.2 Constructor & Destructor Documentation

10.60.2.1 `template<class T> gazebo::physics::HingeJoint< T >::HingeJoint (BasePtr _parent) [inline]`

Constructor.

Parameters

in	_parent	Parent link
----	---------	-------------

10.60.2.2 `template<class T> virtual gazebo::physics::HingeJoint< T >::~~HingeJoint () [inline],[virtual]`

Destructor.

10.60.3 Member Function Documentation

10.60.3.1 `template<class T> virtual unsigned int gazebo::physics::HingeJoint< T >::GetAngleCount () const [inline],[virtual]`

10.60.3.2 `template<class T> virtual void gazebo::physics::HingeJoint< T >::Init () [inline],[protected],[virtual]`

Initialize joint.

10.60.3.3 `template<class T> virtual void gazebo::physics::HingeJoint< T >::Load (sdf::ElementPtr _sdf) [inline],[virtual]`

Load joint.

Parameters

in	_sdf	Pointer to SDF element
----	------	------------------------

Reimplemented in `gazebo::physics::SimbodyHingeJoint` (p. 802).

The documentation for this class was generated from the following file:

- `HingeJoint.hh`

10.61 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** ()
Get the max color.
- int **GetPitch** () const
- **Color GetPixel** (unsigned int _x, unsigned int _y)
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. 391).*

10.61.1 Detailed Description

Encapsulates an image.

10.61.2 Member Enumeration Documentation

10.61.2.1 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

10.61.3 Constructor & Destructor Documentation

10.61.3.1 gazebo::common::Image::Image (const std::string & *filename* = "") [explicit]

Constructor.

Parameters

in	<i>_filename</i>	the path to the image
----	------------------	-----------------------

10.61.3.2 virtual gazebo::common::Image::~Image () [virtual]

Destructor.

10.61.4 Member Function Documentation

10.61.4.1 `static Image::PixelFormat gazebo::common::Image::ConvertPixelFormat (const std::string & _format)`
`[static]`

Convert a string to a **Image::PixelFormat** (p. 391).

Parameters

in	_format	Pixel format string.
----	---------	----------------------

See Also

`Image::PixelFormatNames`

Returns

Image::PixelFormat (p. 391)

10.61.4.2 `Color gazebo::common::Image::GetAvgColor ()`

Get the average color.

Returns

The average color

10.61.4.3 `unsigned int gazebo::common::Image::GetBPP () const`

Get the size of one pixel in bits.

Returns

The BPP of the image

10.61.4.4 `void gazebo::common::Image::GetData (unsigned char ** _data, unsigned int & _count) const`

Get the image as a data array.

Parameters

out	_data	Pointer to a NULL array of char.
out	_count	The resulting data array size

10.61.4.5 `std::string gazebo::common::Image::GetFilename () const`

Get the full filename of the image.

Returns

The filename used to load the image

10.61.4.6 unsigned int gazebo::common::Image::GetHeight () const

Get the height.

Returns

The image height

10.61.4.7 Color gazebo::common::Image::GetMaxColor ()

Get the max color.

Returns

The max color

10.61.4.8 int gazebo::common::Image::GetPitch () const

Returns

The pitch of the image

10.61.4.9 Color gazebo::common::Image::GetPixel (unsigned int _x, unsigned int _y)

Get a pixel color value.

Parameters

in	_x	Column location in the image
in	_y	Row location in the image

10.61.4.10 PixelFormat gazebo::common::Image::GetPixelFormat () const

Get the pixel format.

Returns

PixelFormat

10.61.4.11 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	_data	Pointer to a NULL array of char.
out	_count	The resulting data array size

10.61.4.12 `unsigned int gazebo::common::Image::GetWidth () const`

Get the width.

Returns

The image width

10.61.4.13 `int gazebo::common::Image::Load (const std::string & _filename)`

Load an image.

Return 0 on success

Parameters

<code>in</code>	<code>_filename</code>	the path to the image file
-----------------	------------------------	----------------------------

10.61.4.14 `void gazebo::common::Image::Rescale (int _width, int _height)`

Rescale the image.

Parameters

<code>in</code>	<code>_width</code>	New image width
<code>in</code>	<code>_height</code>	New image height

10.61.4.15 `void gazebo::common::Image::SavePNG (const std::string & _filename)`

Save the image in PNG format.

Parameters

<code>in</code>	<code>_filename</code>	The name of the saved image
-----------------	------------------------	-----------------------------

10.61.4.16 `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

<code>in</code>	<code>_data</code>	Pointer to the raw image data
<code>in</code>	<code>_width</code>	Width in pixels
<code>in</code>	<code>_height</code>	Height in pixels
<code>in</code>	<code>_format</code>	Pixel format of the provided data

10.61.4.17 `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:

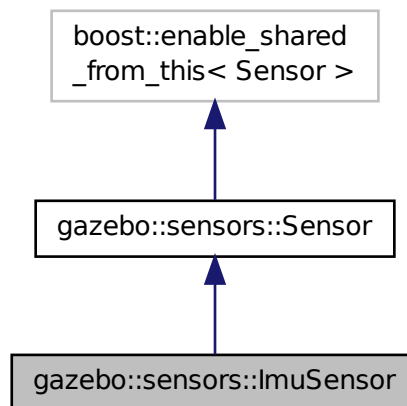
- **Image.hh**

10.62 gazebo::sensors::ImuSensor Class Reference

An IMU sensor.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::ImuSensor:



Public Member Functions

- **ImuSensor ()**
Constructor.
- virtual **~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 void **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.62.1 Detailed Description

An IMU sensor.

10.62.2 Constructor & Destructor Documentation

10.62.2.1 gazebo::sensors::ImuSensor::ImuSensor ()

Constructor.

10.62.2.2 virtual gazebo::sensors::ImuSensor::~~ImuSensor () [virtual]

Destructor.

10.62.3 Member Function Documentation

10.62.3.1 virtual void gazebo::sensors::ImuSensor::Fini () [protected],[virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 755).

10.62.3.2 `math::Vector3 gazebo::sensors::ImuSensor::GetAngularVelocity () const`

Returns the angular velocity.

Returns

Angular velocity.

10.62.3.3 `msgs::IMU gazebo::sensors::ImuSensor::GetImuMessage () const`

Returns the imu message.

Returns

Imu message.

10.62.3.4 `math::Vector3 gazebo::sensors::ImuSensor::GetLinearAcceleration () const`

Returns the imu linear acceleration.

Returns

Linear acceleration.

10.62.3.5 `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.

10.62.3.6 `virtual void gazebo::sensors::ImuSensor::Init () [virtual]`

Initialize the IMU.

Reimplemented from **`gazebo::sensors::Sensor`** (p. 758).

10.62.3.7 `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. 758).

10.62.3.8 `void gazebo::sensors::ImuSensor::Load (const std::string & _worldName, sdf::ElementPtr _sdf)` [protected], [virtual]

Load the sensor with SDF parameters.

Parameters

in	<code>_sdf</code>	SDF Sensor (p. 751) parameters.
in	<code>_worldName</code>	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 759).

10.62.3.9 `virtual void gazebo::sensors::ImuSensor::Load (const std::string & _worldName)` [protected], [virtual]

Load the sensor with default parameters.

Parameters

in	<code>_worldName</code>	Name of world to load from.
----	-------------------------	-----------------------------

Reimplemented from **gazebo::sensors::Sensor** (p. 759).

10.62.3.10 `void gazebo::sensors::ImuSensor::SetReferencePose ()`

Sets the current pose as the IMU reference pose.

10.62.3.11 `virtual void 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

in	<code>_force</code>	True if update is forced, false if not
----	---------------------	--

Reimplemented from **gazebo::sensors::Sensor** (p. 760).

The documentation for this class was generated from the following file:

- **ImuSensor.hh**

10.63 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. 455) 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. 455) frame If you specify GetMOI(this->**GetPose**()) (p. 403), 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. 455) 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. 455) 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.

10.63.1 Detailed Description

A class for inertial information about a link.

10.63.2 Constructor & Destructor Documentation

10.63.2.1 gazebo::physics::Inertial::Inertial ()

Default Constructor.

10.63.2.2 gazebo::physics::Inertial::Inertial (double *_mass*) [explicit]

Constructor.

Parameters

<i>in</i>	<i>_mass</i>	Mass value in kg if using metric.
-----------	--------------	-----------------------------------

10.63.2.3 gazebo::physics::Inertial::Inertial (const Inertial & *_inertial*)

Copy constructor.

Parameters

<i>in</i>	<i>_inertial</i>	Inertial (p. 398) element to copy
-----------	------------------	--

10.63.2.4 virtual gazebo::physics::Inertial::~~Inertial () [virtual]

Destructor.

10.63.3 Member Function Documentation

10.63.3.1 const math::Vector3& gazebo::physics::Inertial::GetCoG () const [inline]

Get the center of gravity.

Returns

The center of gravity.

References gazebo::math::Pose::pos.

10.63.3.2 Inertial gazebo::physics::Inertial::GetInertial (const math::Pose & *_frameOffset*) const

Get equivalent Inertia values with the **Link** (p. 455) frame offset, while holding the Pose of CoG constant in the world frame.

Parameters

<i>in</i>	<i>_frameOffset</i>	amount to offset the Link (p. 455) frame by, this is a transform defined in the Link (p. 455) frame.
-----------	---------------------	--

Returns

Inertial (p. 398) parameters with the shifted frame.

10.63.3.3 `double gazebo::physics::Inertial::GetIXX () const`

Get IXX.

Returns

IXX value

10.63.3.4 `double gazebo::physics::Inertial::GetIXY () const`

Get IXY.

Returns

IXY value

10.63.3.5 `double gazebo::physics::Inertial::GetIXZ () const`

Get IXZ.

Returns

IXZ value

10.63.3.6 `double gazebo::physics::Inertial::GetIYY () const`

Get IYY.

Returns

IYY value

10.63.3.7 `double gazebo::physics::Inertial::GetIYZ () const`

Get IYZ.

Returns

IYZ value

10.63.3.8 `double gazebo::physics::Inertial::GetIZZ () const`

Get IZZ.

Returns

IZZ value

10.63.3.9 `double gazebo::physics::Inertial::GetMass () const`

Get the mass.

10.63.3.10 `math::Matrix3 gazebo::physics::Inertial::GetMOI (const math::Pose & _pose) const`

Get the equivalent inertia from a point in local **Link** (p. 455) frame. If you specify `GetMOI(this->GetPose())` (p. 403), 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. 398) block, then the MOI is rotated accordingly, and contributions from changes in MOI location location due to point mass is added to the final MOI.

Parameters

<code>in</code>	<code>_pose</code>	location in Link (p. 455) local frame
-----------------	--------------------	--

Returns

equivalent inertia at `_pose`

10.63.3.11 `math::Matrix3 gazebo::physics::Inertial::GetMOI () const`

returns Moments of Inertia as a Matrix3

Returns

Moments of Inertia as a Matrix3

10.63.3.12 `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. 455) frame.

Returns

The inertial pose.

10.63.3.13 `math::Vector3 gazebo::physics::Inertial::GetPrincipalMoments () const`

Get the principal moments of inertia (Ixx, Iyy, Izz).

Returns

The principal moments.

10.63.3.14 `math::Vector3 gazebo::physics::Inertial::GetProductsofInertia () const`

Get the products of inertia (Ixy, Ixz, Iyz).

Returns

The products of inertia.

10.63.3.15 void gazebo::physics::Inertial::Load (sdf::ElementPtr *_sdf*)

Load from SDF values.

Parameters

in	<i>_sdf</i>	SDF value to load from.
----	-------------	-------------------------

10.63.3.16 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. 455) frame. New CG is computed from masses and perspective offsets, and both MOI contributions relocated to the new cog.

Parameters

in	<i>_inertial</i>	Inertial (p. 398) to add.
----	------------------	----------------------------------

Returns

The result of the addition.

10.63.3.17 const Inertial& gazebo::physics::Inertial::operator+= (const Inertial & *_inertial*)

Addition equal operator.

Parameters

in	<i>_inertial</i>	Inertial (p. 398) to add.
----	------------------	----------------------------------

Returns

Reference to this object.

10.63.3.18 Inertial& gazebo::physics::Inertial::operator= (const Inertial & *_inertial*)

Equal operator.

Parameters

in	<i>_inertial</i>	Inertial (p. 398) to copy.
----	------------------	-----------------------------------

Returns

Reference to this object.

10.63.3.19 void gazebo::physics::Inertial::ProcessMsg (const msgs::Inertial & *_msg*)

Update parameters from a message.

Parameters

in	<code>_msg</code>	Message to read
----	-------------------	-----------------

10.63.3.20 void gazebo::physics::Inertial::Reset ()

Reset all the mass properties.

10.63.3.21 void gazebo::physics::Inertial::Rotate (const math::Quaternion & `_rot`)

Rotate this mass.

Parameters

in	<code>_rot</code>	Rotation amount.
----	-------------------	------------------

10.63.3.22 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.

10.63.3.23 void gazebo::physics::Inertial::SetCoG (const math::Vector3 & `_center`)

Set the center of gravity.

Parameters

in	<code>_center</code>	Center of the gravity.
----	----------------------	------------------------

10.63.3.24 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. 455) frame.

Parameters

in	<code>_cx</code>	Center offset in x-direction in Link (p. 455) frame
in	<code>_cy</code>	Center offset in y-direction in Link (p. 455) frame
in	<code>_cz</code>	Center offset in z-direction in Link (p. 455) 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.

10.63.3.25 void gazebo::physics::Inertial::SetCoG (const math::Pose & _c)

Set the center of gravity.

Parameters

in	_c	Transform to center of gravity.
----	----	---------------------------------

10.63.3.26 void gazebo::physics::Inertial::SetInertiaMatrix (double _ixx, double _iyy, double _izz, double _ixy, double _ixz, double iyz)

Set the mass matrix.

Parameters

in	_ixx	X second moment of inertia (MOI) about x axis.
in	_iyy	Y second moment of inertia about y axis.
in	_izz	Z second moment of inertia about z axis.
in	_ixy	XY inertia.
in	_ixz	XZ inertia.
in	_iyz	YZ inertia.

10.63.3.27 void gazebo::physics::Inertial::SetIXX (double _v)

Set IXX.

Parameters

in	_v	IXX value
----	----	-----------

10.63.3.28 void gazebo::physics::Inertial::SetIXY (double _v)

Set IXY.

Parameters

in	_v	IXY value
----	----	-----------

10.63.3.29 void gazebo::physics::Inertial::SetIXZ (double _v)

Set IXZ.

Parameters

in	_v	IXZ value
----	----	-----------

10.63.3.30 void gazebo::physics::Inertial::SetIYY (double _v)

Set IYY.

Parameters

in	_v	IYY value
----	----	-----------

10.63.3.31 void gazebo::physics::Inertial::SetIYZ (double _v)

Set IYZ.

Parameters

in	_v	IXX value
----	----	-----------

10.63.3.32 void gazebo::physics::Inertial::SetIZZ (double _v)

Set IZZ.

Parameters

in	_v	IZZ value
----	----	-----------

10.63.3.33 void gazebo::physics::Inertial::SetMass (double m)

Set the mass.

10.63.3.34 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
----	----------------	-------------------------

10.63.3.35 void gazebo::physics::Inertial::UpdateParameters (sdf::ElementPtr _sdf)

update the parameters using new sdf values.

Parameters

in	_sdf	Update values from.
----	------	---------------------

10.63.4 Friends And Related Function Documentation

10.63.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::physics::Inertial & _inertial) [friend]

Output operator.

Parameters

in	<code>_out</code>	Output stream.
in	<code>_inertial</code>	Inertial (p. 398) object to output.

The documentation for this class was generated from the following file:

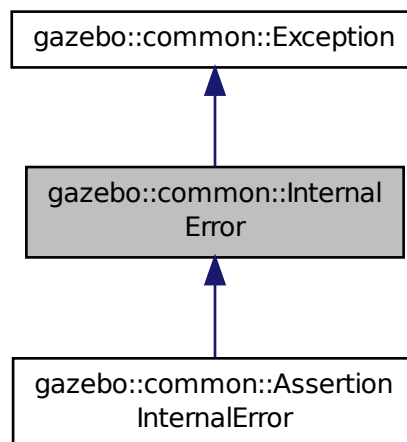
- **Inertial.hh**

10.64 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 **~InternalError** ()
Destructor.

10.64.1 Detailed Description

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

10.64.2 Constructor & Destructor Documentation

10.64.2.1 gazebo::common::InternalError::InternalError ()

Constructor.

10.64.2.2 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

10.64.2.3 virtual gazebo::common::InternalError::~~InternalError () [virtual]

Destructor.

The documentation for this class was generated from the following file:

- **Exception.hh**

10.65 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.

10.65.1 Detailed Description

Manages boost::asio IO.

10.65.2 Constructor & Destructor Documentation

10.65.2.1 gazebo::transport::IOManager::IOManager ()

Constructor.

10.65.2.2 gazebo::transport::IOManager::~~IOManager ()

Destructor.

10.65.3 Member Function Documentation

10.65.3.1 void gazebo::transport::IOManager::DecCount ()

Decrement the event count by 1.

10.65.3.2 unsigned int gazebo::transport::IOManager::GetCount () const

Get the event count.

Returns

The event count

10.65.3.3 boost::asio::io_service& gazebo::transport::IOManager::GetIO ()

Get handle to boost::asio IO service.

Returns

Handle to boost::asio IO service

10.65.3.4 void gazebo::transport::IOManager::IncCount ()

Increment the event count by 1.

10.65.3.5 void gazebo::transport::IOManager::Stop ()

Stop the IO service.

The documentation for this class was generated from the following file:

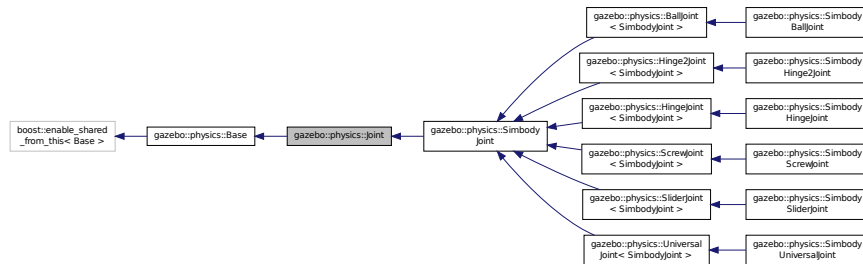
- **IOManager.hh**

10.66 gazebo::physics::Joint Class Reference

Base (p. 153) 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. 411) attribute types.

Public Member Functions

- **Joint (BasePtr _parent)**
Constructor.
- virtual **~Joint ()**
Destructor.
- virtual void **ApplyDamping ()**
Callback to apply damping force 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. 411) Force Torque Values if necessary for physics engine.

- double **CheckAndTruncateForce** (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 **math::Vector3 GetAnchor** (int *_index*) const =0
Get the anchor point.
- **math::Angle GetAngle** (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*)=0
Get a non-generic parameter for the joint.
- **LinkPtr GetChild** () const
Get the child link.
- double **GetDamping** (int *_index*)
Returns the current joint damping coefficient.
- double **GetDampingCoefficient** () const
Get damping coefficient of this joint.
- virtual double **GetEffortLimit** (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** (int *_index*) const =0
Get the axis of rotation in global coordnate frame.
- virtual **math::Angle GetHighStop** (int *_index*)=0
Get the high stop of an axis(index).
- double **GetInertiaRatio** (unsigned int *_index*) const
Accessor to inertia ratio across this joint.
- virtual **LinkPtr GetJointLink** (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. 455) due to the existence of this **Joint** (p. 411).*
- virtual **math::Vector3 GetLinkTorque** (unsigned int *_index*) const =0
*Get the torque applied to the center of mass of a **physics::Link** (p. 455) due to the existence of this **Joint** (p. 411).*
- **math::Vector3 GetLocalAxis** (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** (int *_index*)=0

- Get the low stop of an axis(index).*

 - virtual double **GetMaxForce** (int _index)=0

Get the max allowed force of an axis(index).
- **LinkPtr GetParent** () const
- Get the parent link.*
- **math::Angle GetUpperLimit** (unsigned int _index) const
- : get the joint lower limit (replaces GetLowStop and GetHighStop)*
- virtual double **GetVelocity** (int _index) const =0
- Get the rotation rate of an axis(index)*
- virtual double **GetVelocityLimit** (int _index)
- Get the velocity limit on axis(index).*
- 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. 411).*
- virtual void **Load** (sdf::ElementPtr _sdf)
- Load **physics::Joint** (p. 411) from a SDF sdf::Element.*
- virtual void **Reset** ()
- Reset the joint.*
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)=0
- Set the anchor point.*
- void **SetAngle** (int _index, **math::Angle** _angle)
- If the **Joint** (p. 411) is static, Gazebo stores the state of this **Joint** (p. 411) as a scalar inside the **Joint** (p. 411) class, so this call will NOT move the joint dynamically for a static **Model** (p. 537).*
- virtual void **SetAttribute** (const std::string &_key, int _index, const boost::any &_value)=0
- Set a non-generic parameter for the joint.*
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)=0
- Set the axis of rotation where axis is specified in local joint frame.*
- virtual void **SetDamping** (int _index, double _damping)=0
- Set the joint damping.*
- virtual void **SetForce** (int _index, double _effort)=0
- Set the force applied to this **physics::Joint** (p. 411).*
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)
- Set the high stop of an axis(index).*
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)
- Set the low stop of an axis(index).*
- virtual void **SetMaxForce** (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 void **SetProvideFeedback** (bool _enable)
- Set whether the joint should generate feedback.*
- void **SetState** (const **JointState** &_state)
- Set the joint state.*
- virtual void **SetVelocity** (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** (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
- **LinkPtr childLink**
The first link this joint connects to.
- double **dampingCoefficient**
joint dampingCoefficient
- double **effortLimit** [2]
*Store **Joint** (p. 411) effort limit as specified in SDF.*
- double **inertiaRatio** [2]
*Store **Joint** (p. 411) inertia ratio.*
- **math::Angle lowerLimit** [2]
*Store **Joint** (p. 411) position lower limit as specified in SDF.*
- **ModelPtr model**
Pointer to the parent model.
- **LinkPtr parentLink**
The second link this joint connects to.
- bool **provideFeedback**
Provide Feedback data for contact forces.
- **math::Angle upperLimit** [2]
*Store **Joint** (p. 411) position upper limit as specified in SDF.*
- bool **useCFMDamping**
option to use CFM damping
- double **velocityLimit** [2]
*Store **Joint** (p. 411) velocity limit as specified in SDF.*
- **JointWrench wrench**
*Cache **Joint** (p. 411) force torque values in case physics engine clears them at the end of update step.*

10.66.1 Detailed Description

Base (p. 153) class for all joints.

10.66.2 Member Enumeration Documentation

10.66.2.1 enum gazebo::physics::Joint::Attribute

Joint (p. 411) 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.

10.66.3 Constructor & Destructor Documentation

10.66.3.1 gazebo::physics::Joint::Joint (BasePtr *parent*) [explicit]

Constructor.

Parameters

<i>in</i>	Joint (p. 411)	parent
-----------	-----------------------	--------

10.66.3.2 virtual gazebo::physics::Joint::~~Joint () [virtual]

Destructor.

10.66.4 Member Function Documentation

10.66.4.1 virtual void gazebo::physics::Joint::ApplyDamping () [virtual]

Callback to apply damping force to joint.

10.66.4.2 virtual bool gazebo::physics::Joint::AreConnected (LinkPtr *one*, LinkPtr *two*) const [pure virtual]

Determines of 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.

Implemented in **gazebo::physics::SimbodyJoint** (p. 807).

10.66.4.3 `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.

10.66.4.4 `virtual void gazebo::physics::Joint::CacheForceTorque () [inline],[virtual]`

Cache **Joint** (p. 411) Force Torque Values if necessary for physics engine.

Reimplemented in **gazebo::physics::SimbodyJoint** (p. 807).

10.66.4.5 `double gazebo::physics::Joint::CheckAndTruncateForce (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

10.66.4.6 `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>_subscriber</code>	Callback for the connection.
-----------------	--------------------------	------------------------------

Returns

Connection pointer, which must be kept in scope.

References gazebo::event::EventT< T >::Connect().

10.66.4.7 virtual void gazebo::physics::Joint::Detach () [virtual]

Detach this joint from all links.

Reimplemented in **gazebo::physics::SimbodyJoint** (p. 807).

10.66.4.8 void gazebo::physics::Joint::DisconnectJointUpdate (event::ConnectionPtr & _conn) [inline]

Disconnect a boost::slot the the joint update signal.

Parameters

in	_conn	Connection to disconnect.
----	-------	---------------------------

References gazebo::event::EventT< T >::Disconnect().

10.66.4.9 void gazebo::physics::Joint::FillMsg (msgs::Joint & _msg)

Fill a joint message.

Parameters

out	_msg	Message to fill with this joint's properties.
-----	------	---

10.66.4.10 virtual math::Vector3 gazebo::physics::Joint::GetAnchor (int _index) const [pure virtual]

Get the anchor point.

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

Anchor value for the axis.

Implemented in **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 747), **gazebo::physics::SimbodyJoint** (p. 807), **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 887), **gazebo::physics::SimbodyHinge2Joint** (p. 795), **gazebo::physics::SimbodyUniversalJoint** (p. 860), and **gazebo::physics::SimbodyBallJoint** (p. 781).

10.66.4.11 math::Angle gazebo::physics::Joint::GetAngle (int _index) const

Get the angle of rotation of an axis(index)

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

Angle of the axis.

10.66.4.12 `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::BallJoint< SimbodyJoint >` (p. 152), `gazebo::physics::SliderJoint< SimbodyJoint >` (p. 888), `gazebo::physics::Hinge2Joint< SimbodyJoint >` (p. 387), `gazebo::physics::ScrewJoint< SimbodyJoint >` (p. 748), `gazebo::physics::UniversalJoint< SimbodyJoint >` (p. 978), and `gazebo::physics::HingeJoint< SimbodyJoint >` (p. 389).

10.66.4.13 `virtual math::Angle gazebo::physics::Joint::GetAngleImpl (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::SimbodyHinge2Joint` (p. 795), `gazebo::physics::SimbodyUniversalJoint` (p. 860), `gazebo::physics::SimbodyScrewJoint` (p. 845), `gazebo::physics::SimbodyHingeJoint` (p. 801), `gazebo::physics::SimbodySliderJoint` (p. 852), and `gazebo::physics::SimbodyBallJoint` (p. 781).

10.66.4.14 `virtual double gazebo::physics::Joint::GetAttribute (const std::string & _key, unsigned int _index)` [pure 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.
<code>in</code>	<code>_value</code>	Value of the attribute.

Implemented in `gazebo::physics::SimbodyJoint` (p. 808).

10.66.4.15 `LinkPtr gazebo::physics::Joint::GetChild () const`

Get the child link.

Returns

Pointer to the child link.

10.66.4.16 `double gazebo::physics::Joint::GetDamping (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. 411) viscous damping coefficient for this joint.

10.66.4.17 `double gazebo::physics::Joint::GetDampingCoefficient () const`

Get damping coefficient of this joint.

Returns

viscous joint damping coefficient

10.66.4.18 `virtual double gazebo::physics::Joint::GetEffortLimit (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

10.66.4.19 `virtual double gazebo::physics::Joint::GetForce (unsigned int _index) [virtual]`

Todo : not yet implemented. Get external forces applied at this **Joint** (p.411). 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::SimbodyJoint** (p. 808).

10.66.4.20 `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. 442) data structure. Where **JointWrench.body1Force** (p. 444) contains the force applied by the parent **Link** (p. 455) on the **Joint** (p. 411) specified in the parent **Link** (p. 455) frame, and **JointWrench.body2Force** (p. 444) contains the force applied by the child **Link** (p. 455) on the **Joint** (p. 411) specified in the child **Link** (p. 455) frame. Note that this sign convention is opposite of the reaction forces of the **Joint** (p. 411) 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. 455) 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. 808).

10.66.4.21 `virtual math::Vector3 gazebo::physics::Joint::GetGlobalAxis (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::SimbodyHinge2Joint** (p. 795), **gazebo::physics::SimbodyUniversalJoint** (p. 860), **gazebo::physics::SimbodyScrewJoint** (p. 845), **gazebo::physics::SimbodyBallJoint** (p. 781), **gazebo::physics::SimbodyHingeJoint** (p. 801), and **gazebo::physics::SimbodySliderJoint** (p. 852).

10.66.4.22 `virtual math::Angle gazebo::physics::Joint::GetHighStop (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::SimbodyHinge2Joint** (p. 795), **gazebo::physics::SimbodyUniversalJoint** (p. 861), **gazebo::physics::BallJoint** < **SimbodyJoint** > (p. 152), **gazebo::physics::SimbodyHingeJoint** (p. 801), **gazebo::physics::SimbodySliderJoint** (p. 852), and **gazebo::physics::SimbodyScrewJoint** (p. 845).

10.66.4.23 `double gazebo::physics::Joint::GetInertiaRatio (unsigned int _index) const`

Accessor to inertia ratio across this joint.

Parameters

<code>in</code>	<code><i>_index</i></code>	Joint (p. 411) axis index.
-----------------	----------------------------	-----------------------------------

Returns

returns the inertia ratio across specified joint axis.

10.66.4.24 `virtual LinkPtr gazebo::physics::Joint::GetJointLink (int _index) const` [pure virtual]

Get the link to which the joint is attached according the `_index`.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the link to retrieve.
-----------------	----------------------------	--------------------------------

Returns

Pointer to the request link. NULL if the index was invalid.

Implemented in **gazebo::physics::SimbodyJoint** (p. 809).

10.66.4.25 `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. 455) due to the existence of this **Joint** (p. 411).

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>	The index of the link(0 or 1).
-----------------	---------------------------	--------------------------------

Returns

Force applied to the link.

Implemented in **gazebo::physics::SimbodyJoint** (p. 809).

10.66.4.26 `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. 455) due to the existence of this **Joint** (p. 411).

Note that the unit of torque should be consistent with the rest of the simulation scales.

Parameters

<code>in</code>	<code><i>index</i></code>	The index of the link(0 or 1)
-----------------	---------------------------	-------------------------------

Returns

Torque applied to the link.

Implemented in **gazebo::physics::SimbodyJoint** (p. 809).

10.66.4.27 `math::Vector3 gazebo::physics::Joint::GetLocalAxis (int _index) const`

Get the axis of rotation.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis to get.
-----------------	----------------------------	---------------------------

Returns

Axis value for the provided index.

10.66.4.28 `math::Angle gazebo::physics::Joint::GetLowerLimit (unsigned int _index) const`

: get the joint upper limit (replaces GetLowStop and GetHighStop)

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Upper limit of the axis.

10.66.4.29 `virtual math::Angle gazebo::physics::Joint::GetLowStop (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><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Angle of the low stop value.

Implemented in **gazebo::physics::SimbodyHinge2Joint** (p.796), **gazebo::physics::SimbodyUniversalJoint** (p.861), **gazebo::physics::BallJoint**< **SimbodyJoint** > (p.152), **gazebo::physics::SimbodyHingeJoint** (p.802), **gazebo::physics::SimbodySliderJoint** (p.852), and **gazebo::physics::SimbodyScrewJoint** (p.846).

10.66.4.30 `virtual double gazebo::physics::Joint::GetMaxForce (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><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

The maximum force.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p.846), **gazebo::physics::SimbodyHinge2Joint** (p.796), **gazebo::physics::SimbodyUniversalJoint** (p.861), **gazebo::physics::SimbodyBallJoint** (p.781), **gazebo::physics::SimbodyHingeJoint** (p.802), and **gazebo::physics::SimbodySliderJoint** (p.853).

10.66.4.31 `LinkPtr gazebo::physics::Joint::GetParent () const`

Get the parent link.

Returns

Pointer to the parent link.

10.66.4.32 `math::Angle gazebo::physics::Joint::GetUpperLimit (unsigned int _index) const`

: get the joint lower limit (replacée GetLowStop and GetHighStop)

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Upper limit of the axis.

10.66.4.33 `virtual double gazebo::physics::Joint::GetVelocity (int _index) const [pure 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.

Implemented in `gazebo::physics::SimbodyScrewJoint` (p. 846), `gazebo::physics::SimbodyUniversalJoint` (p. 861), `gazebo::physics::SimbodyHinge2Joint` (p. 796), `gazebo::physics::SimbodyBallJoint` (p. 782), `gazebo::physics::SimbodyHingeJoint` (p. 802), and `gazebo::physics::SimbodySliderJoint` (p. 853).

10.66.4.34 `virtual double gazebo::physics::Joint::GetVelocityLimit (int _index) [virtual]`

Get the velocity limit on axis(index).

Parameters

in	<i>_index</i>	Index of axis, where 0=first axis and 1=second axis
----	---------------	---

Returns

Velocity limit specified in SDF

10.66.4.35 `virtual void gazebo::physics::Joint::Init () [virtual]`

Initialize a joint.

Reimplemented from `gazebo::physics::Base` (p. 160).

Reimplemented in `gazebo::physics::HingeJoint< SimbodyJoint >` (p. 389), `gazebo::physics::SimbodyHinge2Joint` (p. 797), `gazebo::physics::SimbodyScrewJoint` (p. 847), `gazebo::physics::SimbodyBallJoint` (p. 782), and `gazebo::physics::SimbodyUniversalJoint` (p. 862).

10.66.4.36 `void gazebo::physics::Joint::Load (LinkPtr _parent, LinkPtr _child, const math::Pose & _pose)`

Set pose, parent and child links of a `physics::Joint` (p. 411).

Parameters

in	<i>_parent</i>	Parent link.
in	<i>_child</i>	Child link.
in	<i>_pose</i>	Pose containing Joint (p. 411) Anchor offset from child link.

10.66.4.37 `virtual void gazebo::physics::Joint::Load (sdf::ElementPtr _sdf) [virtual]`

Load `physics::Joint` (p. 411) from a SDF `sdf::Element`.

Parameters

in	<i>_sdf</i>	SDF values to load from.
----	-------------	--------------------------

Reimplemented from `gazebo::physics::Base` (p. 161).

Reimplemented in `gazebo::physics::SimbodySliderJoint` (p. 853), `gazebo::physics::Hinge2Joint< SimbodyJoint >` (p. 387), `gazebo::physics::BallJoint< SimbodyJoint >` (p. 152), `gazebo::physics::ScrewJoint< SimbodyJoint >` (p. 748), `gazebo::physics::UniversalJoint< SimbodyJoint >` (p. 978), `gazebo::physics::HingeJoint< SimbodyJoint >` (p. 389), `gazebo::physics::SliderJoint< SimbodyJoint >` (p. 888), `gazebo::physics::SimbodyHingeJoint` (p. 802), `gazebo::physics::SimbodyHinge2Joint` (p. 797), `gazebo::physics::SimbodyUniversalJoint` (p. 862), `gazebo::physics::SimbodyJoint` (p. 810), `gazebo::physics::SimbodyScrewJoint` (p. 847), and `gazebo::physics::SimbodyBallJoint` (p. 782).

10.66.4.38 `virtual void gazebo::physics::Joint::Reset () [virtual]`

Reset the joint.

Reimplemented from `gazebo::physics::Base` (p. 162).

Reimplemented in `gazebo::physics::SimbodyJoint` (p. 810).

10.66.4.39 `virtual void gazebo::physics::Joint::SetAnchor (int _index, const math::Vector3 & _anchor) [pure virtual]`

Set the anchor point.

Parameters

<code>in</code>	<code>_index</code>	Indx of the axis.
<code>in</code>	<code>_anchor</code>	Anchor value.

Implemented in `gazebo::physics::ScrewJoint< SimbodyJoint >` (p. 748), `gazebo::physics::SimbodyJoint` (p. 810), and `gazebo::physics::SliderJoint< SimbodyJoint >` (p. 888).

10.66.4.40 `void gazebo::physics::Joint::SetAngle (int _index, math::Angle _angle)`

If the **Joint** (p. 411) is static, Gazebo stores the state of this **Joint** (p. 411) as a scalar inside the **Joint** (p. 411) class, so this call will NOT move the joint dynamically for a static **Model** (p. 537).

But if this **Model** (p. 537) is not static, then it is updated dynamically, all the conencted children **Link** (p. 455)'s are moved as a result of the **Joint** (p. 411) angle setting. Dynamic **Joint** (p. 411) angle update is accomplished by calling `JointController::SetJointPosition` (p. 435).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_angle</code>	Angle to set the joint to.

10.66.4.41 `virtual void gazebo::physics::Joint::SetAttribute (const std::string & _key, 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::SimbodyJoint` (p. 811).

10.66.4.42 `virtual void gazebo::physics::Joint::SetAxis (int _index, const math::Vector3 & _axis) [pure 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).

Implemented in `gazebo::physics::SimbodyJoint` (p. 811), `gazebo::physics::BallJoint< SimbodyJoint >` (p. 153), `gazebo::physics::SimbodyHinge2Joint` (p. 797), `gazebo::physics::SimbodyUniversalJoint` (p. 862), `gazebo::physics::SimbodyScrewJoint` (p. 847), `gazebo::physics::SimbodyHingeJoint` (p. 803), and `gazebo::physics::SimbodySliderJoint` (p. 854).

10.66.4.43 `virtual void gazebo::physics::Joint::SetDamping (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::SimbodyJoint` (p. 811), `gazebo::physics::SimbodySliderJoint` (p. 854), `gazebo::physics::SimbodyUniversalJoint` (p. 862), `gazebo::physics::SimbodyHinge2Joint` (p. 797), `gazebo::physics::SimbodyHingeJoint` (p. 803), `gazebo::physics::SimbodyScrewJoint` (p. 847), and `gazebo::physics::SimbodyBallJoint` (p. 782).

10.66.4.44 `virtual void gazebo::physics::Joint::SetForce (int _index, double _effort) [pure virtual]`

Set the force applied to this `physics::Joint` (p. 411).

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. 411)). 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::SimbodyJoint` (p. 811).

10.66.4.45 virtual void gazebo::physics::Joint::SetHighStop (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 in **gazebo::physics::SimbodyHinge2Joint** (p. 798), **gazebo::physics::SimbodyUniversalJoint** (p. 863), **gazebo::physics::SimbodyBallJoint** (p. 783), **gazebo::physics::SimbodyHingeJoint** (p. 804), **gazebo::physics::SimbodySliderJoint** (p. 854), and **gazebo::physics::SimbodyScrewJoint** (p. 848).

10.66.4.46 virtual void gazebo::physics::Joint::SetLowStop (int *_index*, const math::Angle & *_angle*) [virtual]

Set the low stop of an axis(index).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_angle</i>	Low stop angle.

Reimplemented in **gazebo::physics::SimbodyHinge2Joint** (p. 798), **gazebo::physics::SimbodyUniversalJoint** (p. 863), **gazebo::physics::SimbodyBallJoint** (p. 783), **gazebo::physics::SimbodyHingeJoint** (p. 804), **gazebo::physics::SimbodySliderJoint** (p. 855), and **gazebo::physics::SimbodyScrewJoint** (p. 848).

10.66.4.47 virtual void gazebo::physics::Joint::SetMaxForce (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

in	<i>_index</i>	Index of the axis.
in	<i>_force</i>	Maximum force that can be applied to the axis.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 848), **gazebo::physics::SimbodyHinge2Joint** (p. 798), **gazebo::physics::SimbodyUniversalJoint** (p. 863), **gazebo::physics::SimbodyBallJoint** (p. 783), **gazebo::physics::SimbodyHingeJoint** (p. 804), and **gazebo::physics::SimbodySliderJoint** (p. 855).

10.66.4.48 void gazebo::physics::Joint::SetModel (ModelIPtr *_model*)

Set the model this joint belongs too.

Parameters

in	<i>_model</i>	Pointer to a model.
----	---------------	---------------------

10.66.4.49 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.
----	----------------------	--------------------------------

10.66.4.50 `void gazebo::physics::Joint::SetState (const JointState & _state)`

Set the joint state.

Parameters

in	<code>_state</code>	Joint (p. 411) state
----	---------------------	-----------------------------

10.66.4.51 `virtual void gazebo::physics::Joint::SetVelocity (int _index, double _vel)` [pure virtual]

Set the velocity of an axis(index).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_vel</code>	Velocity.

Implemented in **`gazebo::physics::SimbodyScrewJoint`** (p. 849), **`gazebo::physics::SimbodyHinge2Joint`** (p. 798), **`gazebo::physics::SimbodyUniversalJoint`** (p. 864), **`gazebo::physics::SimbodyBallJoint`** (p. 784), **`gazebo::physics::SimbodyHingeJoint`** (p. 804), and **`gazebo::physics::SimbodySliderJoint`** (p. 855).

10.66.4.52 `void gazebo::physics::Joint::Update ()` [virtual]

Update the joint.

Reimplemented from **`gazebo::physics::Base`** (p. 163).

10.66.4.53 `virtual void gazebo::physics::Joint::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::Base`** (p. 163).

10.66.5 Member Data Documentation

10.66.5.1 `LinkPtr gazebo::physics::Joint::anchorLink` [protected]

Anchor link.

10.66.5.2 `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

10.66.5.3 `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).

10.66.5.4 `gazebo::event::ConnectionPtr gazebo::physics::Joint::applyDamping` [protected]

apply damping for adding viscous damping forces on updates

10.66.5.5 `LinkPtr gazebo::physics::Joint::childLink` [protected]

The first link this joint connects to.

10.66.5.6 `double gazebo::physics::Joint::dampingCoefficient` [protected]

joint dampingCoefficient

10.66.5.7 `double gazebo::physics::Joint::effortLimit[2]` [protected]

Store **Joint** (p. 411) effort limit as specified in SDF.

10.66.5.8 `double gazebo::physics::Joint::inertiaRatio[2]` [protected]

Store **Joint** (p. 411) inertia ratio.

This is a measure of how well this model behaves using interative LCP solvers.

10.66.5.9 `math::Angle gazebo::physics::Joint::lowerLimit[2]` [protected]

Store **Joint** (p. 411) position lower limit as specified in SDF.

10.66.5.10 `ModelPtr gazebo::physics::Joint::model` [protected]

Pointer to the parent model.

10.66.5.11 `LinkPtr gazebo::physics::Joint::parentLink` [protected]

The second link this joint connects to.

10.66.5.12 `bool gazebo::physics::Joint::provideFeedback` [protected]

Provide Feedback data for contact forces.

10.66.5.13 `math::Angle gazebo::physics::Joint::upperLimit[2]` [protected]

Store **Joint** (p. 411) position upper limit as specified in SDF.

10.66.5.14 `bool gazebo::physics::Joint::useCFMDamping` [protected]

option to use CFM damping

10.66.5.15 `double gazebo::physics::Joint::velocityLimit[2]` [protected]

Store **Joint** (p. 411) velocity limit as specified in SDF.

10.66.5.16 `JointWrench gazebo::physics::Joint::wrench` [protected]

Cache **Joint** (p. 411) force torque values in case physics engine clears them at the end of update step.

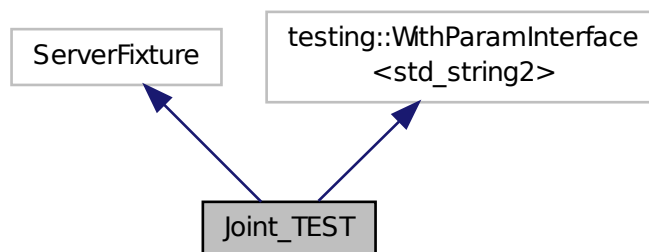
The documentation for this class was generated from the following file:

- **Joint.hh**

10.67 Joint_TEST Class Reference

```
#include <Joint_TEST.hh>
```

Inheritance diagram for Joint_TEST:



Classes

- class **SpawnJointOptions**

Class to hold parameters for spawning joints.

Public Member Functions

- void **ForceTorque1** (const std::string &_physicsEngine)
Load example world with a few joints Measure / verify static force torques against analytical answers.
- void **ForceTorque2** (const std::string &_physicsEngine)
Load example world with a few joints Measure / verify static force torques against analytical answers.
- void **GetForceTorqueWithAppliedForce** (const std::string &_physicsEngine)
Load example world with a few joints.
- void **JointCreationDestructionTest** (const std::string &_physicsEngine)
Create and destroy joints repeatedly, monitors memory usage.
- void **JointTorqueTest** (const std::string &_physicsEngine)
Create a hinge joint between link and world.
- virtual void **SetUp** ()
- **physics::JointPtr SpawnJoint** (const std::string &_type, bool _worldChild=false, bool _worldParent=false, **common::Time** _wait=**common::Time**(99, 0))
Spawn a model with a joint connecting to the world.
- **physics::JointPtr SpawnJoint** (const **SpawnJointOptions** &_opt)
Spawn a model with a joint connecting to the world.
- void **SpawnJointRotational** (const std::string &_physicsEngine, const std::string &_jointType)
Spawn model with rotational joints.
- void **SpawnJointRotationalWorld** (const std::string &_physicsEngine, const std::string &_jointType)
Spawn model with rotational joints.
- void **SpawnJointTypes** (const std::string &_physicsEngine, const std::string &_jointType)
Spawn model with each type of joint.

Protected Member Functions

- **Joint_TEST** ()

Protected Attributes

- std::string **jointType**
Joint type for test.
- std::string **physicsEngine**
Physics engine for test.

10.67.1 Constructor & Destructor Documentation

10.67.1.1 **Joint_TEST::Joint_TEST** () [`inline`],[`protected`]

10.67.2 Member Function Documentation

10.67.2.1 void **Joint_TEST::ForceTorque1** (const std::string & *_physicsEngine*)

Load example world with a few joints Measure / verify static force torques against analytical answers.

Parameters

in	<code>_physicsEngine</code>	Type of physics engine to use.
----	-----------------------------	--------------------------------

10.67.2.2 void Joint_TEST::ForceTorque2 (const std::string & *_physicsEngine*)

Load example world with a few joints Measure / verify static force torques against analytical answers.

Change gravity to tip over the joints. Wait until joint stops are hit and joint motion settles, then check force torques values against analytical values.

Parameters

<i>in</i>	<i>_physicsEngine</i>	Type of physics engine to use.
-----------	-----------------------	--------------------------------

10.67.2.3 void Joint_TEST::GetForceTorqueWithAppliedForce (const std::string & *_physicsEngine*)

Load example world with a few joints.

Servo the joints to a fixed target position using simple PID controller. Measure / verify static force torques against analytical answers.

Parameters

<i>in</i>	<i>_physicsEngine</i>	Type of physics engine to use.
-----------	-----------------------	--------------------------------

10.67.2.4 void Joint_TEST::JointCreationDestructionTest (const std::string & *_physicsEngine*)

Create and destroy joints repeatedly, monitors memory usage.

Parameters

<i>in</i>	<i>_physicsEngine</i>	Type of physics engine to use.
-----------	-----------------------	--------------------------------

10.67.2.5 void Joint_TEST::JointTorqueTest (const std::string & *_physicsEngine*)

Create a hinge joint between link and world.

Apply force and check acceleration against analytical solution.

Parameters

<i>in</i>	<i>_physicsEngine</i>	Type of physics engine to use.
-----------	-----------------------	--------------------------------

10.67.2.6 virtual void Joint_TEST::SetUp () [inline],[virtual]

References gzdbg.

10.67.2.7 physics::JointPtr Joint_TEST::SpawnJoint (const std::string & *_type*, bool *_worldChild* = false, bool *_worldParent* = false, common::Time *_wait* = common::Time(99, 0)) [inline]

Spawn a model with a joint connecting to the world.

The function will wait for duration *_wait* for the model to spawn and attempt to return a pointer to the spawned joint. This function is not guaranteed to return a valid JointPtr, so the output should be checked.

Parameters

in	<code>_type</code>	Type of joint to create.
in	<code>_worldChild</code>	Flag to set child link to the world.
in	<code>_worldParent</code>	Flag to set parent link to the world.
in	<code>_wait</code>	Length of time to wait for model to spawn in order to return Joint pointer.

References `Joint_TEST::SpawnJointOptions::type`, `Joint_TEST::SpawnJointOptions::wait`, `Joint_TEST::SpawnJointOptions::worldChild`, and `Joint_TEST::SpawnJointOptions::worldParent`.

10.67.2.8 `physics::JointPtr Joint_TEST::SpawnJoint (const SpawnJointOptions & .opt) [inline]`

Spawn a model with a joint connecting to the world.

The function will wait for duration `_wait` for the model to spawn and attempt to return a pointer to the spawned joint. This function is not guaranteed to return a valid `JointPtr`, so the output should be checked.

Parameters

in	<code>_opt</code>	Options for spawned model and joint.
----	-------------------	--------------------------------------

References `Joint_TEST::SpawnJointOptions::axis`, `Joint_TEST::SpawnJointOptions::childLinkPose`, `gazebo::physics::get_world()`, `gazebo::common::Time::GetWallTime()`, `gzwarn`, `Joint_TEST::SpawnJointOptions::jointPose`, `Joint_TEST::SpawnJointOptions::modelPose`, `gazebo::common::Time::MSleep()`, `NULL`, `Joint_TEST::SpawnJointOptions::parentLinkPose`, `Joint_TEST::SpawnJointOptions::type`, `Joint_TEST::SpawnJointOptions::wait`, `Joint_TEST::SpawnJointOptions::worldChild`, `Joint_TEST::SpawnJointOptions::worldParent`, and `gazebo::common::Time::Zero`.

10.67.2.9 `void Joint_TEST::SpawnJointRotational (const std::string & .physicsEngine, const std::string & .jointType)`

Spawn model with rotational joints.

Set velocity on parent and make sure child follows.

Parameters

in	<code>_physicsEngine</code>	Type of physics engine to use.
in	<code>_jointType</code>	Type of joint to spawn and test.

10.67.2.10 `void Joint_TEST::SpawnJointRotationalWorld (const std::string & .physicsEngine, const std::string & .jointType)`

Spawn model with rotational joints.

Attach to world and make sure it doesn't fall.

Parameters

in	<code>_physicsEngine</code>	Type of physics engine to use.
in	<code>_jointType</code>	Type of joint to spawn and test.

10.67.2.11 `void Joint_TEST::SpawnJointTypes (const std::string & .physicsEngine, const std::string & .jointType)`

Spawn model with each type of joint.

Parameters

in	<code>_physicsEngine</code>	Type of physics engine to use.
in	<code>_jointType</code>	Type of joint to spawn and test.

10.67.3 Member Data Documentation

10.67.3.1 `std::string Joint_TEST::jointType` [protected]

Joint type for test.

10.67.3.2 `std::string Joint_TEST::physicsEngine` [protected]

Physics engine for test.

The documentation for this class was generated from the following file:

- `Joint_TEST.hh`

10.68 gazebo::physics::JointController Class Reference

A class for manipulating `physics::Joint` (p. 411).

```
#include <physics/physics.hh>
```

Public Member Functions

- **JointController** (`ModelPtr _model`)
Constructor.
- void **AddJoint** (`JointPtr _joint`)
Add a joint to control.
- void **Reset** ()
Reset all commands.
- void **SetJointPosition** (`const std::string &_name`, `double _position`, `int _index=0`)
*Set the positions of a **Joint** (p. 411) by name.*
- void **SetJointPosition** (`JointPtr _joint`, `double _position`, `int _index=0`)
*Set the positions of a **Joint** (p. 411) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 886) and radians for **HingeJoint** (p. 387), etc.*
- void **SetJointPositions** (`const std::map< std::string, double > &_jointPositions`)
*Set the positions of a set of **Joint** (p. 411)'s.*
- void **Update** ()
Update the joint control.

10.68.1 Detailed Description

A class for manipulating `physics::Joint` (p. 411).

10.68.2 Constructor & Destructor Documentation

10.68.2.1 gazebo::physics::JointController::JointController (**ModelPtr** *_model*) [explicit]

Constructor.

Parameters

<i>in</i>	<i>_model</i>	Model (p. 537) that uses this joint controller.
-----------	---------------	--

10.68.3 Member Function Documentation

10.68.3.1 void gazebo::physics::JointController::AddJoint (**JointPtr** *_joint*)

Add a joint to control.

Parameters

<i>in</i>	<i>_joint</i>	Joint (p. 411) to control.
-----------	---------------	-----------------------------------

10.68.3.2 void gazebo::physics::JointController::Reset ()

Reset all commands.

10.68.3.3 void gazebo::physics::JointController::SetJointPosition (const std::string & *_name*, double *_position*, int *_index* = 0)

Set the positions of a **Joint** (p. 411) by name.

See Also

JointController::SetJointPosition(JointPtr, double)

10.68.3.4 void gazebo::physics::JointController::SetJointPosition (**JointPtr** *_joint*, double *_position*, int *_index* = 0)

Set the positions of a **Joint** (p. 411) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 886) and radians for **HingeJoint** (p. 387), etc.

Implementation: In order to change the position of a **Joint** (p. 411) inside a **Model** (p. 537), this call must recursively crawl through all the connected children **Link** (p. 455)'s in this **Model** (p. 537), and update each **Link** (p. 455) Pose affected by this **Joint** (p. 411) 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.

Parameters

<i>in</i>	<i>_joint</i>	Joint (p. 411) to set.
<i>in</i>	<i>_position</i>	Position of the joint.

10.68.3.5 void gazebo::physics::JointController::SetJointPositions (const std::map< std::string, double > & *_jointPositions*)

Set the positions of a set of **Joint** (p. 411)'s.

See Also

JointController::SetJointPosition(JointPtr, double)

10.68.3.6 void gazebo::physics::JointController::Update ()

Update the joint control.

The documentation for this class was generated from the following file:

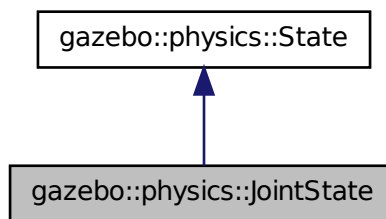
- **JointController.hh**

10.69 gazebo::physics::JointState Class Reference

keeps track of state of a **physics::Joint** (p. 411)

```
#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

10.69.1 Detailed Description

keeps track of state of a **physics::Joint** (p. 411)

10.69.2 Constructor & Destructor Documentation

10.69.2.1 gazebo::physics::JointState::JointState ()

Default constructor.

10.69.2.2 gazebo::physics::JointState::JointState (**JointPtr** _joint, const **common::Time** &_realTime, const **common::Time** &_simTime)

Constructor.

Parameters

in	<i>_joint</i>	Joint (p. 411) to get the state of.
in	<i>_realTime</i>	Real time stamp.
in	<i>_simTime</i>	Sim time stamp.

10.69.2.3 gazebo::physics::JointState::JointState (JointPtr *_joint*) [explicit]

Constructor.

Parameters

in	<i>_joint</i>	Joint (p. 411) to get the state of.
----	---------------	-------------------------------------

10.69.2.4 gazebo::physics::JointState::JointState (const sdf::ElementPtr *_sdf*) [explicit]

Constructor.

Build a JointState (p. 436) from SDF data

Parameters

in	<i>_sdf</i>	SDF data to load a joint state from.
----	-------------	--------------------------------------

10.69.2.5 virtual gazebo::physics::JointState::~~JointState () [virtual]

Destructor.

10.69.3 Member Function Documentation

10.69.3.1 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.
-----	-------------	--------------------------

10.69.3.2 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

common::Exception (p. 331)	When <i>_axis</i> is invalid.
--------------------------------------	-------------------------------

10.69.3.3 `unsigned int gazebo::physics::JointState::GetAngleCount () const`

Get the number of angles.

Returns

The number of angles.

10.69.3.4 `const std::vector<math::Angle>& gazebo::physics::JointState::GetAngles () const`

Get the angles.

Returns

Vector of angles.

10.69.3.5 `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.

10.69.3.6 `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. 411) to get the state of.
<code>in</code>	<code>_realTime</code>	Real time stamp.
<code>in</code>	<code>_simTime</code>	Sim time stamp.

10.69.3.7 `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. 913).

10.69.3.8 `JointState gazebo::physics::JointState::operator+ (const JointState & state) const`

Addition operator.

Parameters

in	<code>_pt</code>	A state to add.
----	------------------	-----------------

Returns

The resulting state.

10.69.3.9 `JointState gazebo::physics::JointState::operator- (const JointState & _state) const`

Subtraction operator.

Parameters

in	<code>_pt</code>	A state to subtract.
----	------------------	----------------------

Returns

The resulting state.

10.69.3.10 `JointState& gazebo::physics::JointState::operator= (const JointState & _state)`

Assignment operator.

Parameters

in	<code>_state</code>	State (p. 910) value
----	---------------------	-----------------------------

Returns

this

10.69.4 Friends And Related Function Documentation

10.69.4.1 `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. 411) state to output.

Returns

The stream.

The documentation for this class was generated from the following file:

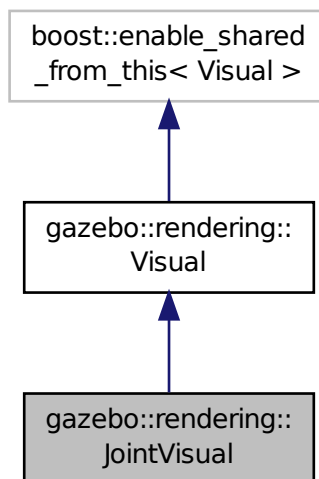
- **JointState.hh**

10.70 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

10.70.1 Detailed Description

Visualization for joints.

10.70.2 Constructor & Destructor Documentation

10.70.2.1 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

10.70.2.2 `virtual gazebo::rendering::JointVisual::~JointVisual() [virtual]`

Destructor.

10.70.3 Member Function Documentation

10.70.3.1 `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](#)

10.71 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.

10.71.1 Detailed Description

Wrench information from a joint.

These are forces and torques on parent and child Links, relative to the **Joint** (p. 411) frame immediately after rotation.

10.71.2 Member Function Documentation

10.71.2.1 JointWrench& gazebo::physics::JointWrench::operator+ (const JointWrench & *wrench*) [inline]

Operator +.

Parameters

<i>in</i>	<i>_wrench</i>	Joint (p. 411) wrench to add
-----------	----------------	-------------------------------------

Returns

*this

References body1Force, body1Torque, body2Force, and body2Torque.

10.71.2.2 JointWrench& gazebo::physics::JointWrench::operator- (const JointWrench & *wrench*) [inline]

Operator -.

Parameters

<i>in</i>	<i>_wrench</i>	Joint (p. 411) wrench to subtract
-----------	----------------	--

Returns

*this

References body1Force, body1Torque, body2Force, and body2Torque.

10.71.2.3 JointWrench& gazebo::physics::JointWrench::operator= (const JointWrench & *wrench*) [inline]

Operator =.

Parameters

<i>in</i>	<i>_wrench</i>	Joint (p. 411) wrench to set from.
-----------	----------------	---

Returns

*this

References body1Force, body1Torque, body2Force, and body2Torque.

10.71.3 Member Data Documentation

10.71.3.1 `math::Vector3 gazebo::physics::JointWrench::body1Force`

Force on the first link.

Referenced by `operator+()`, `operator-()`, and `operator=()`.

10.71.3.2 `math::Vector3 gazebo::physics::JointWrench::body1Torque`

Torque on the first link.

Referenced by `operator+()`, `operator-()`, and `operator=()`.

10.71.3.3 `math::Vector3 gazebo::physics::JointWrench::body2Force`

Force on the second link.

Referenced by `operator+()`, `operator-()`, and `operator=()`.

10.71.3.4 `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**

10.72 `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.

10.72.1 Detailed Description

Generic description of a keyboard event.

10.72.2 Member Enumeration Documentation

10.72.2.1 enum gazebo::common::KeyEvent::EventType

Key event types enumeration.

Enumerator

NO_EVENT

PRESS

RELEASE

10.72.3 Constructor & Destructor Documentation

10.72.3.1 gazebo::common::KeyEvent::KeyEvent () [inline]

Constructor.

10.72.4 Member Data Documentation

10.72.4.1 int gazebo::common::KeyEvent::key

10.72.4.2 EventType gazebo::common::KeyEvent::type

Event type.

The documentation for this class was generated from the following file:

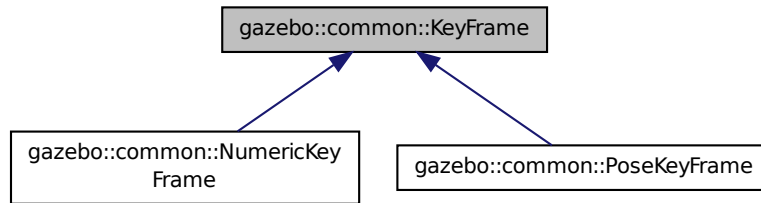
- **KeyEvent.hh**

10.73 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

10.73.1 Detailed Description

A key frame in an animation.

10.73.2 Constructor & Destructor Documentation

10.73.2.1 gazebo::common::KeyFrame::KeyFrame (double *_time*)

Constructor.

Parameters

in	<i>_time</i>	Time (p. 944) of the keyframe in seconds
----	--------------	---

10.73.2.2 virtual gazebo::common::KeyFrame::~~KeyFrame () [virtual]

Destructor.

10.73.3 Member Function Documentation

10.73.3.1 double gazebo::common::KeyFrame::GetTime () const

Get the time of the keyframe.

Returns

the time

10.73.4 Member Data Documentation

10.73.4.1 double gazebo::common::KeyFrame::time [protected]

time of key frame

The documentation for this class was generated from the following file:

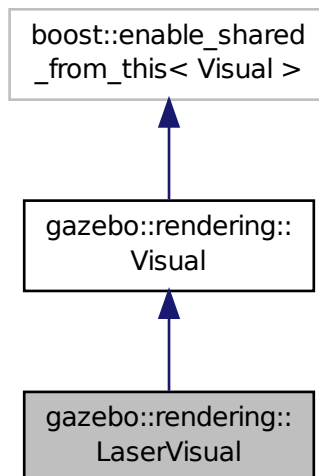
- **KeyFrame.hh**

10.74 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 ~**LaserVisual** ()
Destructor.
- virtual void **SetEmissive** (const **common::Color** &_color)
Documentation inherited from parent.

Additional Inherited Members

10.74.1 Detailed Description

Visualization for laser data.

10.74.2 Constructor & Destructor Documentation

10.74.2.1 gazebo::rendering::LaserVisual::LaserVisual (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. 1034).
in	<code>_topicName</code>	Name of the topic that has laser data.

10.74.2.2 virtual gazebo::rendering::LaserVisual::~~LaserVisual () [virtual]

Destructor.

10.74.3 Member Function Documentation

10.74.3.1 virtual void gazebo::rendering::LaserVisual::SetEmissive (const **common::Color** & _color) [virtual]

Documentation inherited from parent.

Reimplemented from **gazebo::rendering::Visual** (p. 1050).

The documentation for this class was generated from the following file:

- **LaserVisual.hh**

10.75 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.
- 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 **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.

10.75.1 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.

10.75.2 Constructor & Destructor Documentation

10.75.2.1 gazebo::rendering::Light::Light (ScenePtr _scene)

Constructor.

Parameters

in	<code>_scene</code>	Pointer to the scene that contains the Light (p. 448).
----	---------------------	---

10.75.2.2 virtual gazebo::rendering::Light::~Light () [virtual]

Destructor.

10.75.3 Member Function Documentation

10.75.3.1 void gazebo::rendering::Light::FillMsg (msgs::Light & _msg) const

Fill the contents of a light message.

Parameters

out	<code>_msg</code>	Message to fill.
-----	-------------------	------------------

10.75.3.2 `common::Color gazebo::rendering::Light::GetDiffuseColor () const`

Get the diffuse color.

Returns

The light's diffuse color.

10.75.3.3 `math::Vector3 gazebo::rendering::Light::GetDirection () const`

Get the direction.

Returns

The light's direction.

10.75.3.4 `std::string gazebo::rendering::Light::GetName () const`

Get the name of the visual.

Returns

The light's name.

10.75.3.5 `math::Vector3 gazebo::rendering::Light::GetPosition () const`

Get the position of the light.

Returns

The position of the light

10.75.3.6 `common::Color gazebo::rendering::Light::GetSpecularColor () const`

Get the specular color.

Returns

The specular color

10.75.3.7 `std::string gazebo::rendering::Light::GetType () const`

Get the type of the light.

Returns

The light type: "point", "spot", "directional".

10.75.3.8 void gazebo::rendering::Light::Load (sdf::ElementPtr *_sdf*)

Load the light using a set of SDF parameters.

Parameters

in	<i>_sdf</i>	Pointer to the SDF containing the Light (p. 448) description.
----	-------------	--

10.75.3.9 void gazebo::rendering::Light::Load ()

Load the light using default parameters.

10.75.3.10 void gazebo::rendering::Light::LoadFromMsg (ConstLightPtr & *_msg*)

Load from a light message.

Parameters

in	<i>_msg</i>	Containing the light information.
----	-------------	-----------------------------------

10.75.3.11 virtual void gazebo::rendering::Light::OnPoseChange () [inline],[protected],[virtual]

On pose change callback.

10.75.3.12 void gazebo::rendering::Light::SetAttenuation (double *_constant*, double *_linear*, double *_quadratic*)

Set the attenuation.

Parameters

in	<i>_constant</i>	Constant attenuation
in	<i>_linear</i>	Linear attenuation
in	<i>_quadratic</i>	Quadratic attenuation

10.75.3.13 void gazebo::rendering::Light::SetCastShadows (const bool & *_cast*)

Set cast shadows.

Parameters

in	<i>_cast</i>	Set to true to cast shadows.
----	--------------	------------------------------

10.75.3.14 void gazebo::rendering::Light::SetDiffuseColor (const common::Color & *_color*)

Set the diffuse color.

Parameters

in	<code>_color</code>	Light (p. 448) diffuse color.
----	---------------------	-------------------------------

10.75.3.15 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.
----	-------------------	--

10.75.3.16 void gazebo::rendering::Light::SetLightType (const std::string & *_type*)

Set the light type.

Parameters

in	<code>_type</code>	The light type: "point", "spot", "directional"
----	--------------------	--

10.75.3.17 void gazebo::rendering::Light::SetName (const std::string & *_name*)

Set the name of the visual.

Parameters

in	<code>_name</code>	Name of the light source.
----	--------------------	---------------------------

10.75.3.18 void gazebo::rendering::Light::SetPosition (const math::Vector3 & *_p*)

Set the position of the light.

Parameters

in	<code>_p</code>	New position for the light
----	-----------------	----------------------------

10.75.3.19 void gazebo::rendering::Light::SetRange (const double & *_range*)

Set the range.

Parameters

in	<code>_range</code>	Range of the light in meters.
----	---------------------	-------------------------------

10.75.3.20 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.
----	----	---

10.75.3.21 void gazebo::rendering::Light::SetSpecularColor (const common::Color & *_color*)

Set the specular color.

Parameters

in	_color	The specular color
----	--------	--------------------

10.75.3.22 void gazebo::rendering::Light::SetSpotFalloff (const double & *_value*)

Set the spot light falloff.

Parameters

in	_value	Falloff value
----	--------	---------------

10.75.3.23 void gazebo::rendering::Light::SetSpotInnerAngle (const double & *_angle*)

Set the spot light inner angle.

Parameters

in	_angle	Inner angle in radians
----	--------	------------------------

10.75.3.24 void gazebo::rendering::Light::SetSpotOuterAngle (const double & *_angle*)

Set the spot light outer angle.

Parameters

in	_angle	Outer angle in radians
----	--------	------------------------

10.75.3.25 void gazebo::rendering::Light::ShowVisual (bool *_s*)

Set whether to show the visual.

Parameters

in	_s	Set to true to draw a representation of the light.
----	----	--

10.75.3.26 void gazebo::rendering::Light::ToggleShowVisual ()

10.75.3.27 void gazebo::rendering::Light::UpdateFromMsg (ConstLightPtr & _msg)

Update a light source from a message.

Parameters

in	_msg	Light (p. 448) message to update from
----	------	---------------------------------------

The documentation for this class was generated from the following file:

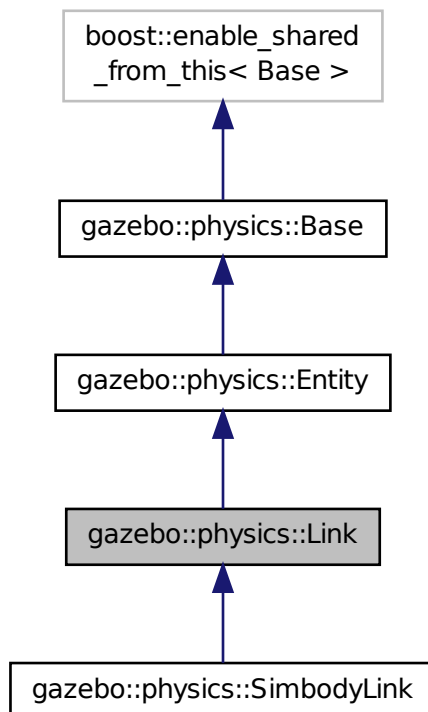
- **Light.hh**

10.76 gazebo::physics::Link Class Reference

Link (p. 455) 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. 455) as a parent **Link** (p. 455).*
- 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. 455) as a child **Link** (p. 455).*
- 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.
- **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.
- virtual **math::Vector3 GetWorldForce** () const =0
Get the force applied to the body in the world frame.
- **math::Vector3 GetWorldLinearAccel** () const

- Get the linear acceleration of the body in the world frame.*

 - virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset=**math::Vector3**(0, 0, 0)) 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. 455) as a parent **Link** (p. 455).*

 - 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. 455) as a child **Link** (p. 455).*

 - 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. 398) properties.
- **math::Vector3** **linearAccel**

Linear acceleration.

- **Visuals_M visuals**

Link (p. 455) visual elements.

Additional Inherited Members

10.76.1 Detailed Description

Link (p. 455) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

10.76.2 Member Typedef Documentation

10.76.2.1 `typedef std::map<uint32_t, msgs::Visual> gazebo::physics::Link::Visuals_M` [protected]

10.76.3 Constructor & Destructor Documentation

10.76.3.1 `gazebo::physics::Link::Link (EntityPtr _parent)` [explicit]

Constructor.

Parameters

in	<code>_parent</code>	Parent of this link.
----	----------------------	----------------------

10.76.3.2 `virtual gazebo::physics::Link::~~Link ()` [virtual]

Destructor.

10.76.4 Member Function Documentation

10.76.4.1 `void gazebo::physics::Link::AddChildJoint (JointPtr _joint)`

Joints that have this **Link** (p. 455) as a parent **Link** (p. 455).

Parameters

in	<code>_joint</code>	Joint (p. 411) that is a child of this link.
----	---------------------	---

10.76.4.2 `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. 816).

10.76.4.3 `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

<code>in</code>	<code><i>_force</i></code>	Force to add.
<code>in</code>	<code><i>_relPos</i></code>	Position on the link to add the force.

Implemented in `gazebo::physics::SimbodyLink` (p. 817).

10.76.4.4 `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><i>_force</i></code>	Force to add.
<code>in</code>	<code><i>_pos</i></code>	Position in global coord frame to add the force.

Implemented in `gazebo::physics::SimbodyLink` (p. 817).

10.76.4.5 `void gazebo::physics::Link::AddParentJoint (JointPtr _joint)`

Joints that have this `Link` (p. 455) as a child `Link` (p. 455).

Parameters

<code>in</code>	<code><i>_joint</i></code>	<code>Joint</code> (p. 411) that is a parent of this link.
-----------------	----------------------------	--

10.76.4.6 `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><i>_force</i></code>	Force to add.
-----------------	----------------------------	---------------

Implemented in `gazebo::physics::SimbodyLink` (p. 817).

10.76.4.7 `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><i>_torque</i></code>	Torque value to add.
-----------------	-----------------------------	----------------------

Implemented in `gazebo::physics::SimbodyLink` (p. 817).

10.76.4.8 `virtual void gazebo::physics::Link::AddTorque (const math::Vector3 & _torque)` [pure virtual]

Add a torque to the body.

Parameters

in	<code><i>_torque</i></code>	Torque value to add to the link.
----	-----------------------------	----------------------------------

Implemented in `gazebo::physics::SimbodyLink` (p. 817).

10.76.4.9 `void gazebo::physics::Link::AttachStaticModel (ModelIPtr & _model, const math::Pose & _offset)`

Attach a static model to this link.

Parameters

in	<code><i>_model</i></code>	Pointer to a static model.
in	<code><i>_offset</i></code>	Pose relative to this link to place the model.

10.76.4.10 `template<typename T > event::ConnectionPtr gazebo::physics::Link::ConnectEnabled (T _subscriber)`
[inline]

Connect to the add entity signal.

Parameters

in	<code><i>_subscriber</i></code>	Subscriber callback function.
----	---------------------------------	-------------------------------

Returns

Pointer to the connection, which must be kept in scope.

References `gazebo::event::EventT< T >::Connect()`.

10.76.4.11 `void gazebo::physics::Link::DetachAllStaticModels ()`

Detach all static models from this link.

10.76.4.12 `void gazebo::physics::Link::DetachStaticModel (const std::string & _modelName)`

Detach a static model from this link.

Parameters

in	<code><i>_modelName</i></code>	Name of an attached model to detach.
----	--------------------------------	--------------------------------------

10.76.4.13 `void gazebo::physics::Link::DisconnectEnabled (event::ConnectionPtr & _conn)` [inline]

Disconnect to the add entity signal.

Parameters

in	<code>_conn</code>	Connection pointer to disconnect.
----	--------------------	-----------------------------------

References gazebo::event::EventT< T >::Disconnect().

10.76.4.14 void gazebo::physics::Link::FillMsg (msgs::Link & *_msg*)

Fill a link message.

Parameters

out	<code>_msg</code>	Message to fill
-----	-------------------	-----------------

10.76.4.15 void gazebo::physics::Link::Fini () [virtual]

Finalize the body.

Reimplemented from **gazebo::physics::Entity** (p. 297).

Reimplemented in **gazebo::physics::SimbodyLink** (p. 818).

10.76.4.16 double gazebo::physics::Link::GetAngularDamping () const

Get the angular damping factor.

Returns

Angular damping.

10.76.4.17 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. 297).

10.76.4.18 Joint_V gazebo::physics::Link::GetChildJoints () const

Get the child joints.

10.76.4.19 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.

10.76.4.20 CollisionPtr gazebo::physics::Link::GetCollision (const std::string & *_name*)

Get a child collision by name.

Parameters

<i>in</i>	<i>_name</i>	Name of the collision object.
-----------	--------------	-------------------------------

Returns

Pointer to the collision, NULL if the name was not found.

10.76.4.21 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.

10.76.4.22 Collision_V gazebo::physics::Link::GetCollisions () const

Get all the child collisions.

Returns

A std::vector of all the child collisions.

10.76.4.23 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::SimbodyLink** (p. 818).

10.76.4.24 virtual bool gazebo::physics::Link::GetGravityMode () const [pure virtual]

Get the gravity mode.

Returns

True if gravity is enabled.

Implemented in **gazebo::physics::SimbodyLink** (p. 818).

10.76.4.25 `InertiaPtr gazebo::physics::Link::GetInertia () const [inline]`

Get the inertia of the link.

Returns

Inertia of the link.

References `Inertia`.

10.76.4.26 `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.

10.76.4.27 `double gazebo::physics::Link::GetLinearDamping () const`

Get the linear damping factor.

Returns

Linear damping.

10.76.4.28 `ModelPtr gazebo::physics::Link::GetModel () const`

Get the model that this body belongs to.

Returns

Model (p. 537) that this body belongs to.

10.76.4.29 `Joint_V gazebo::physics::Link::GetParentJoints () const`

Get the parent joints.

10.76.4.30 `Link_V gazebo::physics::Link::GetParentJointsLinks () const`

Returns a vector of parent Links connected by joints.

Returns

Vector of parent Links connected by joints.

10.76.4.31 **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. 298).

10.76.4.32 **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. 299).

10.76.4.33 **math::Vector3** gazebo::physics::Link::GetRelativeForce () const

Get the force applied to the body.

Returns

Force applied to the body.

10.76.4.34 **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. 299).

10.76.4.35 **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. 299).

10.76.4.36 `math::Vector3 gazebo::physics::Link::GetRelativeTorque () const`

Get the torque applied to the body.

Returns

Torque applied to the body.

10.76.4.37 `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.

10.76.4.38 `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. 467).

Returns

The number of sensors created by the link.

10.76.4.39 `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. 467).

Note

A **Link** (p. 455) does not manage or maintain a pointer to a **sensors::Sensor** (p. 751). Access to a Sensor object is accomplished through the **sensors::SensorManager** (p. 764). 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.

10.76.4.40 `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. 299).

10.76.4.41 `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::SimbodyLink** (p. 818).

10.76.4.42 `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.

10.76.4.43 `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::SimbodyLink** (p. 819).

10.76.4.44 `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. 300).

10.76.4.45 `virtual math::Vector3 gazebo::physics::Link::GetWorldLinearVel (const math::Vector3 & _offset = math::Vector3(0, 0, 0)) 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. 455) frame will be returned.

Parameters

<code>in</code>	<code>_offset</code>	Offset of the point from the origin of the Link (p. 455) frame, expressed in the body-fixed frame.
-----------------	----------------------	---

Returns

Linear velocity of the point on the body

Implemented in **gazebo::physics::SimbodyLink** (p. 819).

10.76.4.46 `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::SimbodyLink** (p. 819).

10.76.4.47 `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::SimbodyLink** (p. 819).

10.76.4.48 `virtual void gazebo::physics::Link::Init ()` [virtual]

Initialize the body.

Reimplemented from **gazebo::physics::Base** (p. 160).

Reimplemented in **gazebo::physics::SimbodyLink** (p. 820).

10.76.4.49 `virtual void gazebo::physics::Link::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::Entity** (p. 301).

Reimplemented in **gazebo::physics::SimbodyLink** (p. 820).

10.76.4.50 `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. 301).

Reimplemented in **gazebo::physics::SimbodyLink** (p. 820).

10.76.4.51 `void gazebo::physics::Link::ProcessMsg (const msgs::Link & _msg)`

Update parameters from a message.

Parameters

in	<code>_msg</code>	Message to read.
----	-------------------	------------------

10.76.4.52 `virtual void gazebo::physics::Link::RemoveChild (EntityPtr _child) [virtual]`

10.76.4.53 `void gazebo::physics::Link::RemoveChildJoint (const std::string & _jointName)`

Remove Joints that have this **Link** (p. 455) as a parent **Link** (p. 455).

Parameters

in	<code>_jointName</code>	Child Joint (p. 411) name.
----	-------------------------	-----------------------------------

10.76.4.54 `void gazebo::physics::Link::RemoveCollision (const std::string & _name)`

Remove a collision from the link.

Parameters

<code>intj</code>	<code>_name</code>	Name of the collision to remove.
-------------------	--------------------	----------------------------------

10.76.4.55 `void gazebo::physics::Link::RemoveParentJoint (const std::string & _jointName)`

Remove Joints that have this **Link** (p. 455) as a child **Link** (p. 455).

Parameters

in	<code>_jointName</code>	Parent Joint (p. 411) name.
----	-------------------------	------------------------------------

10.76.4.56 `void gazebo::physics::Link::Reset () [virtual]`

Reset the link.

Reimplemented from **gazebo::physics::Entity** (p. 301).

10.76.4.57 `void gazebo::physics::Link::ResetPhysicsStates ()`

Reset the link.

10.76.4.58 `void gazebo::physics::Link::SetAngularAccel (const math::Vector3 & _accel)`

Set the angular acceleration of the body.

Parameters

in	<code>_accel</code>	Angular acceleration.
----	---------------------	-----------------------

10.76.4.59 `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::SimbodyLink** (p. 820).

10.76.4.60 `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::SimbodyLink** (p. 820).

10.76.4.61 `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::SimbodyLink** (p. 820).

10.76.4.62 `void gazebo::physics::Link::SetCollideMode (const std::string & _mode)`

Set the collide mode of the body.

Parameters

in	<code>_mode</code>	Collision (p.213) 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
----	--------------------	---

10.76.4.63 `virtual void gazebo::physics::Link::SetEnabled (bool _enable) const` [pure virtual]

Set whether this body is enabled.

Parameters

in	<code>_enable</code>	True to enable the link in the physics engine.
----	----------------------	--

Implemented in `gazebo::physics::SimbodyLink` (p.821).

10.76.4.64 `virtual void gazebo::physics::Link::SetForce (const math::Vector3 & _force)` [pure virtual]

Set the force applied to the body.

Parameters

in	<code>_force</code>	Force value.
----	---------------------	--------------

Implemented in `gazebo::physics::SimbodyLink` (p.821).

10.76.4.65 `virtual void gazebo::physics::Link::SetGravityMode (bool _mode)` [pure virtual]

Set whether gravity affects this body.

Parameters

in	<code>_mode</code>	True to enable gravity.
----	--------------------	-------------------------

Implemented in `gazebo::physics::SimbodyLink` (p.821).

10.76.4.66 `void gazebo::physics::Link::SetInertial (const InertialPtr & _inertial)`

Set the mass of the link.

[in] `_inertial` **Inertial** (p.398) value for the link.

10.76.4.67 `virtual void gazebo::physics::Link::SetKinematic (const bool & _kinematic)` [virtual]

Implement this function.

Set whether this body is in the kinematic state.

Parameters

in	<code>_kinematic</code>	True to make the link kinematic only.
----	-------------------------	---------------------------------------

10.76.4.68 void gazebo::physics::Link::SetLaserRetro (float *_retro*)

Set the laser retro reflectiveness.

Parameters

in	<i>_retro</i>	Retro value for all child collisions.
----	---------------	---------------------------------------

10.76.4.69 void gazebo::physics::Link::SetLinearAccel (const math::Vector3 & *_accel*)

Set the linear acceleration of the body.

Parameters

in	<i>_accel</i>	Linear acceleration.
----	---------------	----------------------

10.76.4.70 virtual void gazebo::physics::Link::SetLinearDamping (double *_damping*) [pure virtual]

Set the linear damping factor.

Parameters

in	<i>_damping</i>	Linear damping factor.
----	-----------------	------------------------

Implemented in **gazebo::physics::SimbodyLink** (p. 821).

10.76.4.71 virtual void gazebo::physics::Link::SetLinearVel (const math::Vector3 & *_vel*) [pure virtual]

Set the linear velocity of the body.

Parameters

in	<i>_vel</i>	Linear velocity.
----	-------------	------------------

Implemented in **gazebo::physics::SimbodyLink** (p. 821).

10.76.4.72 virtual void gazebo::physics::Link::SetLinkStatic (bool *_static*) [pure virtual]

Freeze link to ground (inertial frame).

Parameters

in	<i>_static</i>	if true, freeze link to ground. Otherwise unfreeze link.
----	----------------	--

Implemented in **gazebo::physics::SimbodyLink** (p. 822).

10.76.4.73 void gazebo::physics::Link::SetPublishData (bool *_enable*)

Enable/Disable link data publishing.

Parameters

in	<code>_enable</code>	True to enable publishing, false to stop publishing
----	----------------------	---

10.76.4.74 `void gazebo::physics::Link::SetScale (const math::Vector3 & _scale)`

Set the scale of the link.

Parameters

in	<code>_scale</code>	Scale to set the link to.
----	---------------------	---------------------------

10.76.4.75 `virtual bool gazebo::physics::Link::SetSelected (bool _set)` [virtual]

Set whether this entity has been selected by the user through the gui.

Parameters

in	<code>_set</code>	True to set the link as selected.
----	-------------------	-----------------------------------

Reimplemented from `gazebo::physics::Base` (p. 163).

10.76.4.76 `virtual void gazebo::physics::Link::SetSelfCollide (bool _collide)` [pure virtual]

Set whether this body will collide with others in the model.

Parameters

in	<code>_collid</code>	True to enable collisions.
----	----------------------	----------------------------

Implemented in `gazebo::physics::SimbodyLink` (p. 822).

10.76.4.77 `void gazebo::physics::Link::SetState (const LinkState & _state)`

Set the current link state.

Parameters

in	<code>_state</code>	The state to set the link to.
----	---------------------	-------------------------------

10.76.4.78 `virtual void gazebo::physics::Link::SetTorque (const math::Vector3 & _torque)` [pure virtual]

Set the torque applied to the body.

Parameters

in	<code>_torque</code>	Torque value.
----	----------------------	---------------

Implemented in `gazebo::physics::SimbodyLink` (p. 822).

10.76.4.79 void gazebo::physics::Link::Update (const common::UpdateInfo & *_info*)

Update the collision.

Parameters

in	<i>_info</i>	Update information.
----	--------------	---------------------

10.76.4.80 virtual void gazebo::physics::Link::UpdateMass () [inline],[virtual]

Update the mass matrix.

10.76.4.81 virtual void gazebo::physics::Link::UpdateParameters (sdf::ElementPtr *_sdf*) [virtual]

Update the parameters using new sdf values.

Parameters

in	<i>_sdf</i>	SDF values to load from.
----	-------------	--------------------------

Reimplemented from **gazebo::physics::Entity** (p. 303).

10.76.4.82 virtual void gazebo::physics::Link::UpdateSurface () [inline],[virtual]

Update surface parameters.

10.76.5 Member Data Documentation

10.76.5.1 **math::Vector3** gazebo::physics::Link::angularAccel [protected]

Angular acceleration.

10.76.5.2 **std::vector<math::Pose>** gazebo::physics::Link::attachedModelsOffset [protected]

Offsets for the attached models.

10.76.5.3 **std::vector<std::string>** gazebo::physics::Link::cgVisuals [protected]

Center of gravity visual elements.

10.76.5.4 **InertialPtr** gazebo::physics::Link::inertial [protected]

Inertial (p. 398) properties.

Referenced by GetInertial().

10.76.5.5 `math::Vector3 gazebo::physics::Link::linearAccel` [protected]

Linear acceleration.

10.76.5.6 `Visuals_M gazebo::physics::Link::visuals` [protected]

Link (p. 455) visual elements.

The documentation for this class was generated from the following file:

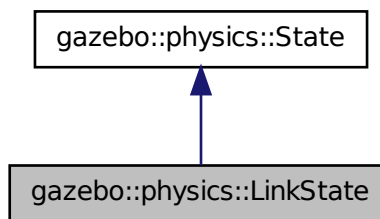
- **Link.hh**

10.77 `gazebo::physics::LinkState` Class Reference

Store state information of a **physics::Link** (p. 455) 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. 455).*
- 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. 476) from a **Link** (p. 455) 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

10.77.1 Detailed Description

Store state information of a **physics::Link** (p. 455) object.

This class captures the entire state of a **Link** (p. 455) at one specific time during a simulation run.

State (p. 910) of a **Link** (p. 455) includes the state of itself all its child **Collision** (p. 213) entities.

10.77.2 Constructor & Destructor Documentation

10.77.2.1 gazebo::physics::LinkState::LinkState ()

Default constructor.

10.77.2.2 gazebo::physics::LinkState::LinkState (const LinkPtr *_link*, const common::Time & *_realTime*, const common::Time & *_simTime*)

Constructor.

Build a **LinkState** (p. 476) from an existing **Link** (p. 455).

Parameters

in	<i>_model</i>	Pointer to the Link (p. 455) from which to gather state info.
in	<i>_realTime</i>	Real time stamp.
in	<i>_simTime</i>	Sim time stamp

10.77.2.3 gazebo::physics::LinkState::LinkState (const LinkPtr *_link*) [explicit]

Constructor.

Build a **LinkState** (p. 476) from an existing **Link** (p. 455).

Parameters

in	<i>_model</i>	Pointer to the Link (p. 455) from which to gather state info.
----	---------------	--

10.77.2.4 gazebo::physics::LinkState::LinkState (const sdf::ElementPtr *_sdf*) [explicit]

Constructor.

Build a **LinkState** (p. 476) from SDF data

Parameters

in	<i>_sdf</i>	SDF data to load a link state from.
----	-------------	-------------------------------------

10.77.2.5 virtual gazebo::physics::LinkState::~~LinkState () [virtual]

Destructor.

10.77.3 Member Function Documentation

10.77.3.1 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.
-----	-------------------	--------------------------

10.77.3.2 `const math::Pose& gazebo::physics::LinkState::GetAcceleration () const`

Get the link acceleration.

Returns

The acceleration represented as a **math::Pose** (p. 648).

10.77.3.3 `CollisionState gazebo::physics::LinkState::GetCollisionState (unsigned int _index) const`

Get a collision state.

Get a **Collision** (p. 213) **State** (p. 910) based on an index, where index is in the range of 0...**LinkState::GetCollisionStateCount** (p. 480).

Parameters

in	<code>_index</code>	Index of the CollisionState (p. 222).
----	---------------------	--

Returns

State (p. 910) of the **Collision** (p. 213).

Exceptions

<i>common::Exception</i> (p. 331)	When <code>_index</code> is invalid.
---	--------------------------------------

10.77.3.4 `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. 222) with the matching name, if any.

Parameters

in	<code>_collisionName</code>	Name of the CollisionState (p. 222)
----	-----------------------------	--

Returns

State (p. 910) of the **Collision** (p. 213).

Exceptions

<i>common::Exception</i> (p. 331)	When <code>_collisionName</code> is invalid
---	---

10.77.3.5 `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. 222) recorded.

10.77.3.6 `const std::vector<CollisionState>& gazebo::physics::LinkState::GetCollisionStates () const`

Get the collision states.

Returns

A vector of collision states.

10.77.3.7 `const math::Pose& gazebo::physics::LinkState::GetPose () const`

Get the link pose.

Returns

The **math::Pose** (p. 648) of the **Link** (p. 455).

10.77.3.8 `const math::Pose& gazebo::physics::LinkState::GetVelocity () const`

Get the link velocity.

Returns

The velocity represented as a **math::Pose** (p. 648).

10.77.3.9 `const math::Pose& gazebo::physics::LinkState::GetWrench () const`

Get the force applied to the **Link** (p. 455).

Returns

Magnitude of the force.

10.77.3.10 `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.

10.77.3.11 void gazebo::physics::LinkState::Load (const LinkPtr *_link*, const common::Time & *_realTime*, const common::Time & *_simTime*)

Load a **LinkState** (p. 476) from a **Link** (p. 455) pointer.

Build a **LinkState** (p. 476) from an existing **Link** (p. 455).

Parameters

in	<i>_model</i>	Pointer to the Link (p. 455) from which to gather state info.
in	<i>_realTime</i>	Real time stamp.
in	<i>_simTime</i>	Sim time stamp

10.77.3.12 virtual void gazebo::physics::LinkState::Load (const sdf::ElementPtr *_elem*) [virtual]

Load state from SDF element.

Load **LinkState** (p. 476) information from stored data in and SDF::Element.

Parameters

in	<i>_elem</i>	Pointer to the SDF::Element containing state info.
----	--------------	--

Reimplemented from **gazebo::physics::State** (p. 913).

10.77.3.13 **LinkState** gazebo::physics::LinkState::operator+ (const LinkState & *_state*) const

Addition operator.

Parameters

in	<i>_pt</i>	A state to add.
----	------------	-----------------

Returns

The resulting state.

10.77.3.14 **LinkState** gazebo::physics::LinkState::operator- (const LinkState & *_state*) const

Subtraction operator.

Parameters

in	<i>_pt</i>	A state to subtract.
----	------------	----------------------

Returns

The resulting state.

10.77.3.15 LinkState& gazebo::physics::LinkState::operator= (const LinkState & _state)

Assignment operator.

Parameters

in	_state	State (p. 910) value
----	--------	-----------------------------

Returns

this

10.77.3.16 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. 914).

10.77.3.17 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. 914).

10.77.3.18 virtual void gazebo::physics::LinkState::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. 910) data was recorded.
----	-------	---

Reimplemented from **gazebo::physics::State** (p. 914).

10.77.4 Friends And Related Function Documentation

10.77.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::physics::LinkState & _state)` [friend]

Stream insertion operator.

Parameters

in	<code>_out</code>	output stream
in	<code>_state</code>	Link (p. 455) state to output

Returns

the stream

Disabling this for efficiency.

Disabling this for efficiency.

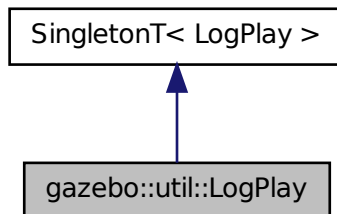
The documentation for this class was generated from the following file:

- [LinkState.hh](#)

10.78 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

10.78.1 Member Function Documentation

10.78.1.1 bool gazebo::util::LogPlay::GetChunk (unsigned int _index, std::string & _data)

Get data for a particular chunk index.

Parameters

in	_index	Index of the chunk.
out	_data	Storage for the chunk's data.

Returns

True if the _index was valid.

10.78.1.2 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.

10.78.1.3 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. 486) has not been called at least once.

10.78.1.4 `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.

10.78.1.5 `std::string gazebo::util::LogPlay::GetHeader () const`

Get the header that was read from a log file.

Should call **LogPlay::Open** (p. 485) first.

Returns

Header of the open log file.

10.78.1.6 `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.

10.78.1.7 `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. 690).

10.78.1.8 `bool gazebo::util::LogPlay::IsOpen () const`

Return true if a file is open.

Returns

True if a log file is open.

10.78.1.9 `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>

10.78.1.10 `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**

10.79 Logplay Class Reference

Open and playback log files that were recorded using LogRecord.

10.79.1 Detailed Description

Open and playback log files that were recorded using LogRecord.

Use **Logplay** (p. 486) 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:

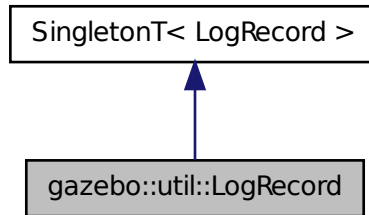
- **LogPlay.hh**

10.80 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

10.80.1 Detailed Description

addtogroup gazebo_util

Handles logging of data to disk

The **LogRecord** (p. 486) 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. 483) for playback functionality.

State information for an entity may be logged through the **LogRecord::Add** (p. 488) function, and stopped through the **LogRecord::Remove** (p. 491) function. Data may be logged into a single file, or split into many separate files by specifying different filenames for the **LogRecord::Add** (p. 488) function.

The **LogRecord** (p. 486) is updated at the start of each simulation step. This guarantees that all data is stored.

See Also

Logplay (p. 486), State

10.80.2 Member Function Documentation

10.80.2.1 void gazebo::util::LogRecord::Add (const std::string & _name, const std::string & _filename, boost::function< bool(std::ostringstream &)> _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	<code>_name</code>	Name of the object to log.
in	<code>_filename</code>	Filename of the log file.
in	<code>_logCallback</code>	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>

10.80.2.2 void gazebo::util::LogRecord::Fini ()

Finalize, and shutdown.

10.80.2.3 std::string gazebo::util::LogRecord::GetBasePath () const

Get the base path for a log recording.

Returns

Path for log recording.

10.80.2.4 unsigned int gazebo::util::LogRecord::GetBufferSize () const

Get the size of the buffer.

Returns

Size of the buffer, in bytes.

10.80.2.5 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.

10.80.2.6 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.

10.80.2.7 unsigned int gazebo::util::LogRecord::GetFileSize (const std::string & _name = " ") const

Get the file size for a log object.

Parameters

in	<code>_name</code>	Name of the log object.
----	--------------------	-------------------------

Returns

Size in bytes.

10.80.2.8 `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.

10.80.2.9 `bool gazebo::util::LogRecord::GetPaused () const`

Get whether logging is paused.

Returns

True if logging is paused.

See Also

LogRecord::SetPaused (p. 491)

10.80.2.10 `bool gazebo::util::LogRecord::GetRunning () const`

Get whether logging is running.

Returns

True if logging has been started.

10.80.2.11 `common::Time gazebo::util::LogRecord::GetRunTime () const`

Get the run time in sim time.

Returns

Run sim time.

10.80.2.12 `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.

10.80.2.13 `bool gazebo::util::LogRecord::IsReadyToStart () const`

Get whether the logger is ready to start, which implies that any previous runs have finished.

10.80.2.14 `void gazebo::util::LogRecord::Notify ()`

Tell the recorder that an update should occur.

10.80.2.15 `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.

10.80.2.16 `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.
-----------------	--------------------	-----------------------------------

10.80.2.17 `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. 490)

10.80.2.18 `bool gazebo::util::LogRecord::Start (const std::string & _encoding = "zlib", const std::string & _path = "")`

Start the logger.

Parameters

<code>in</code>	<code><i>_encoding</i></code>	The type of encoding (txt, zlib, or bz2).
<code>in</code>	<code><i>_path</i></code>	Path in which to store log files.

10.80.2.19 `void gazebo::util::LogRecord::Stop ()`

Stop the logger.

10.80.2.20 `void gazebo::util::LogRecord::Write (bool _force = false)`

Write all logs.

Parameters

<code>in</code>	<code><i>_force</i></code>	True to skip waiting on dataAvailableCondition.
-----------------	----------------------------	---

The documentation for this class was generated from the following file:

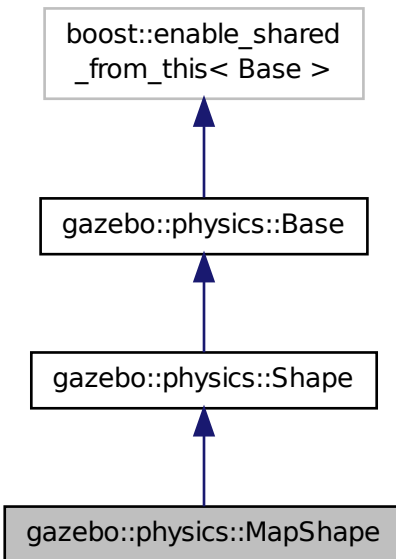
- **LogRecord.hh**

10.81 gazebo::physics::MapShape Class Reference

Creates box extrusions based on an image.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::MapShape:



Public Member Functions

- **MapShape** (*CollisionPtr* _parent)
Constructor.
- virtual **~MapShape** ()
Destructor.
- void **FillMsg** (*msgs::Geometry* &_msg)
Fills out a msgs::Geometry message containing information about this map geometry object.
- int **GetGranularity** () const
Returns granularity of this geometry.
- double **GetHeight** () const
Returns height of this geometry.
- virtual **math::Vector3** **GetScale** () const
Returns scaling factor for this geometry.
- int **GetThreshold** () const
Returns image threshold for this geometry.
- std::string **GetURI** () const
Returns the image URI for this geometry.
- virtual void **Init** ()
Init the map.
- virtual void **Load** (*sdf::ElementPtr* _sdf)
Load the map.

- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
: *Implement this function.*
- void **SetScale** (const math::Vector3 &_scale)
: *Set the scale of the map shape.*
- void **Update** ()
: *Update function.*

Additional Inherited Members

10.81.1 Detailed Description

Creates box extrusions based on an image.

This function is not yet complete, to be implemented.

10.81.2 Constructor & Destructor Documentation

10.81.2.1 gazebo::physics::MapShape::MapShape (CollisionPtr _parent) [explicit]

Constructor.

Parameters

in	<code>_parent</code>	Parent collision object.
----	----------------------	--------------------------

10.81.2.2 virtual gazebo::physics::MapShape::~~MapShape () [virtual]

Destructor.

10.81.3 Member Function Documentation

10.81.3.1 void gazebo::physics::MapShape::FillMsg (msgs::Geometry & _msg) [virtual]

Fills out a msgs::Geometry message containing information about this map geometry object.

Parameters

in	<code>_msg</code>	Message to fill with this object's data.
----	-------------------	--

Implements **gazebo::physics::Shape** (p. 777).

10.81.3.2 int gazebo::physics::MapShape::GetGranularity () const

Returns granularity of this geometry.

Returns

Granularity (amount of error between the image pixels and the 3D shapes created).

10.81.3.3 `double gazebo::physics::MapShape::GetHeight () const`

Returns height of this geometry.

All regions in image with value larger than **MapShape::scale** (p. 778) will be replaced by boxes with MapShape::height.

Returns

Height of the map shapes.

10.81.3.4 `virtual math::Vector3 gazebo::physics::MapShape::GetScale () const` [virtual]

Returns scaling factor for this geometry.

Returns

Scaling factor.

Reimplemented from **gazebo::physics::Shape** (p. 777).

10.81.3.5 `int gazebo::physics::MapShape::GetThreshold () const`

Returns image threshold for this geometry.

All regions in image with value larger than **MapShape::scale** (p. 778) will be replaced by boxes with MapShape::height.

Returns

Image threshold value.

10.81.3.6 `std::string gazebo::physics::MapShape::GetURI () const`

Returns the image URI for this geometry.

Returns

The image URI that was used to load the map.

10.81.3.7 `virtual void gazebo::physics::MapShape::Init ()` [virtual]

Init the map.

Implements **gazebo::physics::Shape** (p. 777).

10.81.3.8 `virtual void gazebo::physics::MapShape::Load (sdf::ElementPtr _sdf)` [virtual]

Load the map.

Parameters

in	_sdf	Load the map from SDF values.
----	------	-------------------------------

Reimplemented from `gazebo::physics::Base` (p. 161).

10.81.3.9 `virtual void gazebo::physics::MapShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]`

: Implement this function.

Parameters

in	_msg	Message to process, which will alter the map.
----	------	---

Implements `gazebo::physics::Shape` (p. 778).

10.81.3.10 `void gazebo::physics::MapShape::SetScale (const math::Vector3 & _scale) [virtual]`

Set the scale of the map shape.

Parameters

in	_scale	Scale to set the map shape to.
----	--------	--------------------------------

Implements `gazebo::physics::Shape` (p. 778).

10.81.3.11 `void gazebo::physics::MapShape::Update () [virtual]`

Update function.

Reimplemented from `gazebo::physics::Base` (p. 163).

The documentation for this class was generated from the following file:

- **MapShape.hh**

10.82 gazebo::Master Class Reference

A ROS Master-like manager that directs gztopic 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.

10.82.1 Detailed Description

A ROS Master-like manager that directs gztopic 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.496), runs World update and sensor generation loops.

10.82.2 Constructor & Destructor Documentation

10.82.2.1 gazebo::Master::Master ()

Constructor.

10.82.2.2 virtual gazebo::Master::~~Master () [virtual]

Destructor.

10.82.3 Member Function Documentation

10.82.3.1 void gazebo::Master::Fini ()

Finalize the master.

10.82.3.2 void gazebo::Master::Init (uint16_t _port)

Initialize.

Parameters

in	<i>_port</i>	The master's port
----	--------------	-------------------

10.82.3.3 void gazebo::Master::Run ()

Run the master.

10.82.3.4 void gazebo::Master::RunOnce ()

Run the master one iteration.

10.82.3.5 void gazebo::Master::RunThread ()

Run the master in a new thread.

10.82.3.6 void gazebo::Master::Stop ()

Stop the master.

The documentation for this class was generated from the following file:

- **Master.hh**

10.83 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.

10.83.1 Detailed Description

Encapsulates description of a material.

10.83.2 Member Enumeration Documentation

10.83.2.1 enum gazebo::common::Material::BlendMode

Enumerator

ADD

MODULATE

REPLACE

BLEND_COUNT

10.83.2.2 enum gazebo::common::Material::ShadeMode

Enumerator

FLAT**GOURAUD****PHONG****BLINN****SHADE_COUNT**

10.83.3 Constructor & Destructor Documentation

10.83.3.1 gazebo::common::Material::Material ()

Constructor.

10.83.3.2 virtual gazebo::common::Material::~Material () [virtual]

Destructor.

10.83.3.3 gazebo::common::Material::Material (const Color & _clr)

Create a material with a default color.

Parameters

in	_clr	Color (p. 226) of the material
----	------	---------------------------------------

10.83.4 Member Function Documentation

10.83.4.1 Color gazebo::common::Material::GetAmbient () const

Get the ambient color.

Returns

The ambient color

10.83.4.2 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

10.83.4.3 BlendMode gazebo::common::Material::GetBlendMode () const

Get the blending mode.

Returns

the blend mode

10.83.4.4 bool gazebo::common::Material::GetDepthWrite () const

Get depth write.

Returns

the depth write enabled state

10.83.4.5 Color gazebo::common::Material::GetDiffuse () const

Get the diffuse color.

Returns

The diffuse color

10.83.4.6 Color gazebo::common::Material::GetEmissive () const

Get the emissive color.

Returns

The emissive color

10.83.4.7 bool gazebo::common::Material::GetLighting () const

Get lighting enabled.

Returns

the lighting enabled state

10.83.4.8 std::string gazebo::common::Material::GetName () const

Get the name of the material.

Returns

The name of the material

10.83.4.9 `double gazebo::common::Material::GetPointSize () const`

Get the point size.

Returns

the point size

10.83.4.10 `ShadeMode gazebo::common::Material::GetShadeMode () const`

Get the shading mode.

Returns

the shading mode

10.83.4.11 `double gazebo::common::Material::GetShininess () const`

Get the shininess.

Returns

The shininess value

10.83.4.12 `Color gazebo::common::Material::GetSpecular () const`

Get the specular color.

Returns

The specular color

10.83.4.13 `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

10.83.4.14 `double gazebo::common::Material::GetTransparency () const`

Get the transparency percentage (0..1)

Returns

The transparency percentage

10.83.4.15 void gazebo::common::Material::SetAmbient (const Color & *_clr*)

Set the ambient color.

Parameters

in	<i>_clr</i>	The ambient color
----	-------------	-------------------

10.83.4.16 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

10.83.4.17 void gazebo::common::Material::SetBlendMode (BlendMode *_b*)

Set the blending mode.

Parameters

in	<i>_b</i>	the blend mode
----	-----------	----------------

10.83.4.18 void gazebo::common::Material::SetDepthWrite (bool *_value*)

Set depth write.

Parameters

in	<i>_value</i>	the depth write enabled state
----	---------------	-------------------------------

10.83.4.19 void gazebo::common::Material::SetDiffuse (const Color & *_clr*)

Set the diffuse color.

Parameters

in	<i>_clr</i>	The diffuse color
----	-------------	-------------------

10.83.4.20 void gazebo::common::Material::SetEmissive (const Color & *_clr*)

Set the emissive color.

Parameters

in	<code>_clr</code>	The emissive color
----	-------------------	--------------------

10.83.4.21 void gazebo::common::Material::SetLighting (bool *_value*)

Set lighting enabled.

Parameters

in	<code>_value</code>	the lighting enabled state
----	---------------------	----------------------------

10.83.4.22 void gazebo::common::Material::SetPointSize (double *_size*)

Set the point size.

Parameters

in	<code>_size</code>	the size
----	--------------------	----------

10.83.4.23 void gazebo::common::Material::SetShadeMode (ShadeMode *_b*)

Set the shading mode param[in] the shading mode.

10.83.4.24 void gazebo::common::Material::SetShininess (double *_t*)

Set the shininess.

Parameters

in	<code>_t</code>	The shininess value
----	-----------------	---------------------

10.83.4.25 void gazebo::common::Material::SetSpecular (const Color & *_clr*)

Set the specular color.

Parameters

in	<code>_clr</code>	The specular color
----	-------------------	--------------------

10.83.4.26 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
----	-------------------	--

10.83.4.27 `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>

10.83.4.28 `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)
-----------------	-----------------	-----------------------------------

10.83.5 Friends And Related Function Documentation

10.83.5.1 `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.

10.83.6 Member Data Documentation

10.83.6.1 **Color** `gazebo::common::Material::ambient` `[protected]`

the ambient light color

10.83.6.2 **BlendMode** `gazebo::common::Material::blendMode` `[protected]`

blend mode

10.83.6.3 `std::string gazebo::common::Material::BlendModeStr[BLEND_COUNT]` `[static]`

10.83.6.4 **Color** `gazebo::common::Material::diffuse` `[protected]`

the diffuse ligh color

10.83.6.5 **Color** `gazebo::common::Material::emissive` `[protected]`

the emissive light color

10.83.6.6 `std::string gazebo::common::Material::name` `[protected]`

the name of the material

10.83.6.7 double gazebo::common::Material::pointSize [protected]

point size

10.83.6.8 **ShadeMode** gazebo::common::Material::shadeMode [protected]

the shade mode

10.83.6.9 std::string gazebo::common::Material::ShadeModeStr[SHADE_COUNT] [static]

10.83.6.10 double gazebo::common::Material::shininess [protected]

shininess value (0 to 1)

10.83.6.11 **Color** gazebo::common::Material::specular [protected]

the specular light color

10.83.6.12 std::string gazebo::common::Material::texImage [protected]

the texture image file name

10.83.6.13 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**

10.84 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.
- **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.

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.

10.84.1 Detailed Description

A 3x3 matrix class.

10.84.2 Constructor & Destructor Documentation

10.84.2.1 gazebo::math::Matrix3::Matrix3 ()

Constructor.

Referenced by operator*(*l*), operator+(*l*), and operator-(*l*).

10.84.2.2 gazebo::math::Matrix3::Matrix3 (const Matrix3 & _m)

Copy constructor.

Parameters

_m	Matrix to copy
----	----------------

10.84.2.3 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

10.84.2.4 virtual gazebo::math::Matrix3::~~Matrix3 () [virtual]

Desctructor.

10.84.3 Member Function Documentation

10.84.3.1 Matrix3 gazebo::math::Matrix3::operator* (const double & _s) const [inline]

returns the element wise scalar multiplication

References m, and Matrix3().

10.84.3.2 Matrix3 gazebo::math::Matrix3::operator* (const Matrix3 & _m) const [inline]

Matrix multiplication operator.

Parameters

in	_m	Matrix3 (p. 507) to multiply
----	----	-------------------------------------

Returns

product of this * _m

References m, and Matrix3().

10.84.3.3 **Matrix3** gazebo::math::Matrix3::operator+ (const Matrix3 & *_m*) const [inline]

returns the element wise sum of two matrices

References *m*, and Matrix3().

10.84.3.4 **Matrix3** gazebo::math::Matrix3::operator- (const Matrix3 & *_m*) const [inline]

returns the element wise difference of two matrices

References *m*, and Matrix3().

10.84.3.5 **bool** gazebo::math::Matrix3::operator== (const Matrix3 & *_m*) const

Equality test operator.

Parameters

<i>in</i>	<i>_m</i>	Matrix3 (p. 507) to test
-----------	-----------	---------------------------------

Returns

True if equal (using the default tolerance of 1e-6)

10.84.3.6 **const double*** gazebo::math::Matrix3::operator[] (size_t *_row*) const [inline]

Array subscript operator.

Parameters

<i>in</i>	<i>_row</i>	row index
-----------	-------------	-----------

Returns

a pointer to the row

References *m*.

10.84.3.7 **double*** gazebo::math::Matrix3::operator[] (size_t *_row*) [inline]

Array subscript operator.

Parameters

<i>in</i>	<i>_row</i>	row index
-----------	-------------	-----------

Returns

a pointer to the row

References *m*.

10.84.3.8 void gazebo::math::Matrix3::SetCol (unsigned int *_c*, const Vector3 & *_v*)

Set a column.

Parameters

in	<i>_c</i>	The column index (0, 1, 2)
in	<i>_v</i>	The value to set in each row of the column

10.84.3.9 void gazebo::math::Matrix3::SetFromAxes (const Vector3 & *_xAxis*, const Vector3 & *_yAxis*, const Vector3 & *_zAxis*)

Set the matrix from three axis (1 per column)

Parameters

in	<i>_xAxis</i>	The x axis
in	<i>_yAxis</i>	The y axis
in	<i>_zAxis</i>	The z axis

10.84.3.10 void gazebo::math::Matrix3::SetFromAxis (const Vector3 & *_axis*, double *_angle*)

Set the matrix from an axis and angle.

Parameters

in	<i>_axis</i>	the axis
in	<i>_angle</i>	ccw rotation around the axis in radians

10.84.4 Friends And Related Function Documentation

10.84.4.1 Matrix3 operator* (double *_s*, const Matrix3 & *_m*) [*friend*]

Multiplication operators.

Parameters

in	<i>_s</i>	the scaling factor
in	<i>_m</i>	input matrix

Returns

a scaled matrix

10.84.4.2 std::ostream& operator<< (std::ostream & *_out*, const gazebo::math::Matrix3 & *_m*) [*friend*]

Stream insertion operator.

Parameters

in	<i>_out</i>	Output stream
in	<i>_m</i>	Matrix to output

Returns

the stream

10.84.5 Member Data Documentation**10.84.5.1** `double gazebo::math::Matrix3::m[3][3]` `[protected]`

the 3x3 matrix

Referenced by `operator*()`, `operator+()`, `operator-()`, and `operator[]()`.

The documentation for this class was generated from the following file:

- **Matrix3.hh**

10.85 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. 648).*
- **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. 1004).*
- **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.

10.85.1 Detailed Description

A 3x3 matrix class.

10.85.2 Constructor & Destructor Documentation

10.85.2.1 gazebo::math::Matrix4::Matrix4 ()

Constructor.

10.85.2.2 gazebo::math::Matrix4::Matrix4 (const Matrix4 & _m)

Copy constructor.

Parameters

_m	Matrix to copy
----	----------------

10.85.2.3 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

in	_v00	Row 0, Col 0 value
in	_v01	Row 0, Col 1 value
in	_v02	Row 0, Col 2 value
in	_v03	Row 0, Col 3 value
in	_v10	Row 1, Col 0 value
in	_v11	Row 1, Col 1 value
in	_v12	Row 1, Col 2 value
in	_v13	Row 1, Col 3 value
in	_v20	Row 2, Col 0 value
in	_v21	Row 2, Col 1 value
in	_v22	Row 2, Col 2 value
in	_v23	Row 2, Col 3 value
in	_v30	Row 3, Col 0 value
in	_v31	Row 3, Col 1 value
in	_v32	Row 3, Col 2 value
in	_v33	Row 3, Col 3 value

10.85.2.4 virtual gazebo::math::Matrix4::~~Matrix4 () [virtual]

Destructor.

10.85.3 Member Function Documentation

10.85.3.1 math::Pose gazebo::math::Matrix4::GetAsPose () const

Get the transformation as **math::Pose** (p. 648).

Returns

the pose

10.85.3.2 Vector3 gazebo::math::Matrix4::GetEulerRotation (unsigned int *solution_number* = 1) const

Get the rotation as a Euler angles.

Returns

the rotation

10.85.3.3 Quaternion gazebo::math::Matrix4::GetRotation () const

Get the rotation as a quaternion.

Returns

the rotation

10.85.3.4 Vector3 gazebo::math::Matrix4::GetTranslation () const

Get the translational values as a **Vector3** (p. 1004).

Returns

x,y,z

10.85.3.5 Matrix4 gazebo::math::Matrix4::Inverse () const

Return the inverse matrix.

10.85.3.6 bool gazebo::math::Matrix4::IsAffine () const

Return true if the matrix is affine.

Returns

true if the matrix is affine, false otherwise

10.85.3.7 Matrix4 gazebo::math::Matrix4::operator* (const Matrix4 & *_mat*) const

Multiplication operator.

Parameters

<i>_mat</i>	Incoming matrix
-------------	-----------------

Returns

This matrix * `_mat`

10.85.3.8 Matrix4 gazebo::math::Matrix4::operator* (const Matrix3 & *_mat*) const

Multiplication operator.

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

Returns

This matrix * `_mat`

10.85.3.9 Vector3 gazebo::math::Matrix4::operator* (const Vector3 & *_vec*) const

Multiplication operator.

Parameters

<code>_vec</code>	Vector3 (p. 1004)
-------------------	--------------------------

Returns

Resulting vector from multiplication

10.85.3.10 Matrix4& gazebo::math::Matrix4::operator= (const Matrix4 & *_mat*)

Equal operator.

this = `_mat`

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

Returns

itself

10.85.3.11 const Matrix4& gazebo::math::Matrix4::operator= (const Matrix3 & *_mat*)

Equal operator for 3x3 matrix.

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

Returns

itself

10.85.3.12 `bool gazebo::math::Matrix4::operator==(const Matrix4 & _m) const`

Equality operator.

Parameters

<code>in</code>	<code>_m</code>	Matrix3 (p. 507) to test
-----------------	-----------------	---------------------------------

Returns

true if the 2 matrices are equal (using the tolerance 1e-6), false otherwise

10.85.3.13 `double* gazebo::math::Matrix4::operator[](size_t _row) [inline]`

Array subscript operator.

Parameters

<code>in</code>	<code>_row</code>	the row index
-----------------	-------------------	---------------

Returns

the row

References m.

10.85.3.14 `const double* gazebo::math::Matrix4::operator[](size_t _row) const [inline]`

Parameters

<code>in</code>	<code>_row</code>	the row index
-----------------	-------------------	---------------

Returns

the row

References m.

10.85.3.15 `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

10.85.3.16 void gazebo::math::Matrix4::SetScale (const Vector3 & _s)

Set the scale.

Parameters

in	<code>_s</code>	scale
----	-----------------	-------

10.85.3.17 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
----	-----------------	---------------

10.85.3.18 Vector3 gazebo::math::Matrix4::TransformAffine (const Vector3 & _v) const

Perform an affine transformation.

Parameters

<code>_v</code>	Vector3 (p. 1004) value for the transformation
-----------------	---

Returns

The result of the transformation

10.85.4 Friends And Related Function Documentation

10.85.4.1 `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

10.85.5 Member Data Documentation

10.85.5.1 `const Matrix4 gazebo::math::Matrix4::IDENTITY` [static]

Identity matrix.

10.85.5.2 `double gazebo::math::Matrix4::m[4][4]` [protected]

The 4x4 matrix.

Referenced by operator[()].

10.85.5.3 `const Matrix4 gazebo::math::Matrix4::ZERO` [static]

Zero matrix.

The documentation for this class was generated from the following file:

- **Matrix4.hh**

10.86 gazebo::common::Mesh Class Reference

A 3D mesh.

```
#include <common/common.hh>
```

Public Member Functions

- **Mesh** ()
Constructor.
- virtual **~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`.

10.86.1 Detailed Description

A 3D mesh.

10.86.2 Constructor & Destructor Documentation

10.86.2.1 gazebo::common::Mesh::Mesh ()

Constructor.

10.86.2.2 virtual gazebo::common::Mesh::~~Mesh () [virtual]

Destructor.

10.86.3 Member Function Documentation

10.86.3.1 int gazebo::common::Mesh::AddMaterial (**Material** * `_mat`)

Add a material to the mesh.

Parameters

<code>in</code>	<code>_mat</code>	the material
-----------------	-------------------	--------------

Returns

Index of this material

10.86.3.2 void gazebo::common::Mesh::AddSubMesh (**SubMesh** * `_child`)

Add a submesh mesh.

The **Mesh** (p. 519) object takes ownership of the submesh.

Parameters

<code>in</code>	<code>_child</code>	the submesh
-----------------	---------------------	-------------

10.86.3.3 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.
----	----------------------	------------------------------

10.86.3.4 `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

10.86.3.5 `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
----	----------------------	------------------------------

10.86.3.6 `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

10.86.3.7 `unsigned int gazebo::common::Mesh::GetIndexCount () const`

Return the number of indices.

Returns

the count

10.86.3.8 `const Material* gazebo::common::Mesh::GetMaterial (int _index) const`

Get a material.

Parameters

in	<code>_index</code>	the index
----	---------------------	-----------

Returns

the material or NULL if the index is out of bounds

10.86.3.9 unsigned int gazebo::common::Mesh::GetMaterialCount () const

Get the number of materials.

Returns

the count

10.86.3.10 math::Vector3 gazebo::common::Mesh::GetMax () const

Get the maximum X, Y, Z values.

Returns

the upper bounds of the bounding box

10.86.3.11 math::Vector3 gazebo::common::Mesh::GetMin () const

Get the minimum X, Y, Z values.

Returns

the lower bounds of the bounding box

10.86.3.12 std::string gazebo::common::Mesh::GetName () const

Get the name of this mesh.

Returns

the name

10.86.3.13 unsigned int gazebo::common::Mesh::GetNormalCount () const

Return the number of normals.

Returns

the count

10.86.3.14 std::string gazebo::common::Mesh::GetPath () const

Get the path which contains the mesh resource.

Returns

the path to the mesh resource

10.86.3.15 **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.

10.86.3.16 **const SubMesh*** gazebo::common::Mesh::GetSubMesh (unsigned int *_i*) const

Get a child mesh.

Parameters

<i>in</i>	<i>_i</i>	the index
-----------	-----------	-----------

Returns

the submesh. An exception is thrown if the index is out of bounds

10.86.3.17 **const SubMesh*** gazebo::common::Mesh::GetSubMesh (const std::string & *_name*) const

Get a child mesh by name.

Parameters

<i>in</i>	<i>_name</i>	Name of the submesh.
-----------	--------------	----------------------

Returns

The submesh, NULL if the *_name* is not found.

10.86.3.18 **unsigned int** gazebo::common::Mesh::GetSubMeshCount () const

Get the number of children.

Returns

the count

10.86.3.19 **unsigned int** gazebo::common::Mesh::GetTexCoordCount () const

Return the number of texture coordinates.

Returns

the count

10.86.3.20 `unsigned int gazebo::common::Mesh::GetVertexCount () const`

Return the number of vertices.

Returns

the count

10.86.3.21 `bool gazebo::common::Mesh::HasSkeleton () const`

Return true if mesh is attached to a skeleton.

10.86.3.22 `void gazebo::common::Mesh::RecalculateNormals ()`

Recalculate all the normals of each face defined by three indices.

10.86.3.23 `void gazebo::common::Mesh::Scale (double _factor)`

Scale all vertices by *_factor*.

Parameters

<i>_factor</i>	Scaling factor
----------------	----------------

10.86.3.24 `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
-----------	-----------	-----------------

10.86.3.25 `void gazebo::common::Mesh::SetPath (const std::string & _path)`

Set the path which contains the mesh resource.

Parameters

<i>in</i>	<i>_path</i>	the file path
-----------	--------------	---------------

10.86.3.26 `void gazebo::common::Mesh::SetScale (const math::Vector3 & _factor)`

Scale all vertices by the *_factor* vector.

Parameters

<i>in</i>	<i>_factor</i>	Scaling vector
-----------	----------------	----------------

10.86.3.27 void gazebo::common::Mesh::SetSkeleton (Skeleton * *_skel*)

Set the mesh skeleton.

10.86.3.28 void gazebo::common::Mesh::Translate (const math::Vector3 & *_vec*)

Move all vertices in all submeshes by *_vec*.

Parameters

<i>in</i>	<i>_vec</i>	Amount to translate vertices.
-----------	-------------	-------------------------------

The documentation for this class was generated from the following file:

- **Mesh.hh**

10.87 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.

10.87.1 Detailed Description

Creates CSG meshes.

10.87.2 Member Enumeration Documentation

10.87.2.1 enum gazebo::common::MeshCSG::BooleanOperation

An enumeration of the boolean operations.

Enumerator

UNION

INTERSECTION

DIFFERENCE

10.87.3 Constructor & Destructor Documentation

10.87.3.1 gazebo::common::MeshCSG::MeshCSG ()

Constructor.

10.87.3.2 virtual gazebo::common::MeshCSG::~~MeshCSG () [virtual]

Destructor.

10.87.4 Member Function Documentation

10.87.4.1 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>	<i>_m2</i> 's pose offset from <i>_m1</i>

Returns

a pointer to the created mesh

The documentation for this class was generated from the following file:

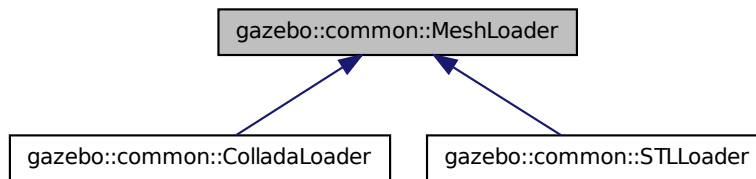
- **MeshCSG.hh**

10.88 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.

10.88.1 Detailed Description

Base class for loading meshes.

10.88.2 Constructor & Destructor Documentation

10.88.2.1 gazebo::common::MeshLoader::MeshLoader ()

Constructor.

10.88.2.2 virtual gazebo::common::MeshLoader::~~MeshLoader () [virtual]

Destructor.

10.88.3 Member Function Documentation

10.88.3.1 virtual Mesh* gazebo::common::MeshLoader::Load (const std::string & .filename) [pure virtual]

Load a 3D mesh.

Parameters

in	<code>_filename</code>	the path to the mesh
----	------------------------	----------------------

Returns

a pointer to the created mesh

Implemented in `gazebo::common::ColladaLoader` (p. 212), and `gazebo::common::STLLoader` (p. 916).

The documentation for this class was generated from the following file:

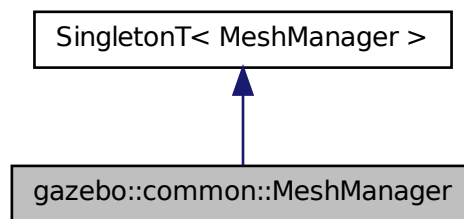
- `MeshLoader.hh`

10.89 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

10.89.1 Detailed Description

Maintains and manages all meshes.

10.89.2 Member Function Documentation

10.89.2.1 void gazebo::common::MeshManager::AddMesh (**Mesh** * _mesh)

Add a mesh to the manager.

This **MeshManager** (p. 529) takes ownership of the mesh and will destroy it. See `~MeshManager`.

Parameters

in	<i>the</i>	mesh to add.
----	------------	--------------

10.89.2.2 void gazebo::common::MeshManager::CreateBox (const std::string & .name, const **math::Vector3** & .sides, const **math::Vector2d** & .uvCoords)

Create a Box mesh.

Parameters

in	_name	the name of the new mesh
in	_sides	the x y x dimentionions of eah side in meter
in	_uvCoords	the texture coordinates

10.89.2.3 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

10.89.2.4 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

10.89.2.5 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

10.89.2.6 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

10.89.2.7 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

10.89.2.8 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

10.89.2.9 void gazebo::common::MeshManager::CreateTube (const std::string & *_name*, float *_innerRadius*, float *_outterRadius*, 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>_outterRadius</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

10.89.2.10 void gazebo::common::MeshManager::GenSphericalTexCoord (const Mesh * *_mesh*, math::Vector3 *_center*)

generate spherical texture coordinates

10.89.2.11 const Mesh* gazebo::common::MeshManager::GetMesh (const std::string & *_name*) const

Get a mesh by name.

Parameters

in	<i>_name</i>	the name of the mesh to look for
----	--------------	----------------------------------

Returns

the mesh or NULL if not found

10.89.2.12 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	<i>_mesh</i>	the mesh
out	<i>_center</i>	the AAB center position
out	<i>_min_xyz</i>	the bounding box minimum
out	<i>_max_xyz</i>	the bounding box maximum

10.89.2.13 bool gazebo::common::MeshManager::HasMesh (const std::string & *_name*) const

Return true if the mesh exists.

Parameters

in	<i>_name</i>	the name of the mesh
----	--------------	----------------------

10.89.2.14 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

10.89.2.15 const Mesh* gazebo::common::MeshManager::Load (const std::string & *_filename*)

Load a mesh from a file.

Parameters

in	<i>_filename</i>	the path to the mesh
----	------------------	----------------------

Returns

a pointer to the created mesh

The documentation for this class was generated from the following file:

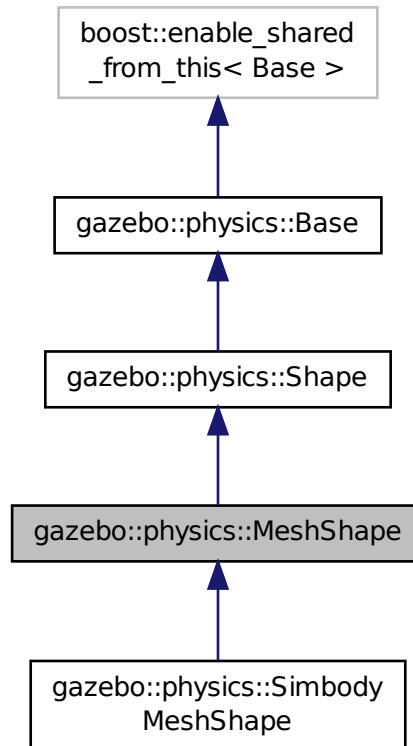
- **MeshManager.hh**

10.90 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

10.90.1 Detailed Description

Triangle mesh collision shape.

10.90.2 Constructor & Destructor Documentation

10.90.2.1 gazebo::physics::MeshShape::MeshShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent collision.
----	----------------	-------------------

10.90.2.2 virtual gazebo::physics::MeshShape::~~MeshShape () [virtual]

Destructor.

10.90.3 Member Function Documentation

10.90.3.1 void gazebo::physics::MeshShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Populate a msgs::Geometry message with data from this shape.

Parameters

out	<i>_msg</i>	Message to fill.
-----	-------------	------------------

Implements **gazebo::physics::Shape** (p. 777).

10.90.3.2 `std::string gazebo::physics::MeshShape::GetMeshURI () const`

Get the URI of the mesh data.

Returns

The URI of the mesh data.

10.90.3.3 `virtual math::Vector3 gazebo::physics::MeshShape::GetSize () const [virtual]`

Get the size of the triangle mesh.

Returns

The size of the triangle mesh.

10.90.3.4 `virtual void gazebo::physics::MeshShape::Init () [virtual]`

Initialize the shape.

Implements **gazebo::physics::Shape** (p. 777).

Reimplemented in **gazebo::physics::SimbodyMeshShape** (p. 824).

10.90.3.5 `virtual void gazebo::physics::MeshShape::ProcessMsg (const msgs::Geometry & .msg) [virtual]`

Update this shape from a message.

Parameters

in	_msg	Message that contains triangle mesh info.
----	------	---

Implements **gazebo::physics::Shape** (p. 778).

10.90.3.6 `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	_uri	Filename of the mesh file to load from.
in	_submesh	Name of the submesh to use within the mesh
in	_center	True to center the submesh. specified in the _uri.

10.90.3.7 `void gazebo::physics::MeshShape::SetScale (const math::Vector3 & _scale) [virtual]`

Set the scaling factor.

Parameters

in	<code>_scale</code>	Scaling factor.
----	---------------------	-----------------

Implements `gazebo::physics::Shape` (p. 778).

10.90.3.8 `virtual void gazebo::physics::MeshShape::Update () [inline],[virtual]`

Update the tri mesh.

Reimplemented from `gazebo::physics::Base` (p. 163).

10.90.4 Member Data Documentation

10.90.4.1 `const common::Mesh* gazebo::physics::MeshShape::mesh [protected]`

Pointer to the mesh data.

10.90.4.2 `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:

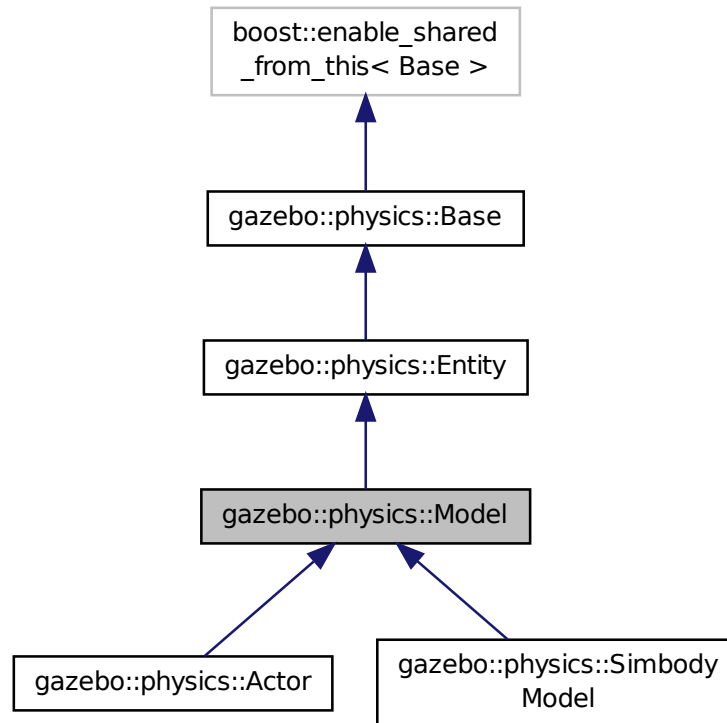
- `MeshShape.hh`

10.91 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.
- **Link_V GetLinks** () const
*Construct and return a vector of **Link** (p. 455)'s in this model Note this constructs the vector of **Link** (p. 455)'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.
- 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. 455), 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** > _anim, boost::function< void()> _onComplete=NULL)
***Joint** (p. 411) Animation.*
- void **SetJointPosition** (const std::string &_jointName, double _position, int _index=0)
*Set the positions of a **Joint** (p. 411) 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. 537) by specifying desired Pose of a **Link** (p. 455) within the **Model** (p. 537).*
- void **SetLinkWorldPose** (const **math::Pose** &_pose, const **LinkPtr** &_link)
*Set the Pose of the entire **Model** (p. 537) by specifying desired Pose of a **Link** (p. 455) within the **Model** (p. 537).*
- 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. 541)*
- std::vector< **math::Pose** > **attachedModelsOffset**
*used by **Model::AttachStaticModel** (p. 541)*
- **transport::PublisherPtr** **jointPub**
Publisher for joint info.

Additional Inherited Members

10.91.1 Detailed Description

A model is a collection of links, joints, and plugins.

10.91.2 Constructor & Destructor Documentation

10.91.2.1 gazebo::physics::Model::Model (**BasePtr** *_parent*) [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent object.
----	----------------	----------------

10.91.2.2 virtual gazebo::physics::Model::~~Model () [virtual]

Destructor.

10.91.3 Member Function Documentation

10.91.3.1 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. 537), which is a **Model** (p. 537) that has been marked as static (no physics simulation), and attaches it to this **Model** (p. 537) 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. 537), to place <code>_model</code> .

10.91.3.2 `void gazebo::physics::Model::DetachStaticModel (const std::string & _model)`

Detach a static model from this model.

Parameters

in	<code>_model</code>	Name of an attached static model to remove.
----	---------------------	---

See Also

Model::AttachStaticModel (p. 541).

10.91.3.3 `virtual void gazebo::physics::Model::FillMsg (msgs::Model & _msg) [virtual]`

Fill a model message.

Parameters

in	<code>_msg</code>	Message to fill using this model's data.
----	-------------------	--

10.91.3.4 `virtual void gazebo::physics::Model::Fini () [virtual]`

Finalize the model.

Reimplemented from **gazebo::physics::Entity** (p. 297).

Reimplemented in **gazebo::physics::Actor** (p. 128).

10.91.3.5 `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.

10.91.3.6 `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. 297).

10.91.3.7 GripperPtr gazebo::physics::Model::GetGripper (size_t *_index*) const

Get a gripper based on an index.

Returns

A pointer to a **Gripper** (p. 368). Null if the `_index` is invalid.

10.91.3.8 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. 543)

10.91.3.9 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

10.91.3.10 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.

10.91.3.11 unsigned int gazebo::physics::Model::GetJointCount () const

Get the number of joints.

Returns

Get the number of joints.

10.91.3.12 `const Joint_V& gazebo::physics::Model::GetJoints () const`

Get the joints.

Returns

Vector of joints.

10.91.3.13 `LinkPtr gazebo::physics::Model::GetLink (const std::string & _name = "canonical") const`

Get a link by name.

Parameters

in	_name	Name of the link to get.
----	-------	--------------------------

Returns

Pointer to the link, NULL if the name is invalid.

10.91.3.14 `Link_V gazebo::physics::Model::GetLinks () const`

Construct and return a vector of **Link** (p. 455)'s in this model Note this constructs the vector of **Link** (p. 455)'s on the fly, could be costly.

Returns

a vector of **Link** (p. 455)'s in this model

10.91.3.15 `unsigned int gazebo::physics::Model::GetPluginCount () const`

Get the number of plugins this model has.

Returns

Number of plugins associated with this model.

10.91.3.16 `virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularAccel () const` [virtual]

Get the angular acceleration of the entity.

Returns

math::Vector3 (p. 1004), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 298).

10.91.3.17 virtual **math::Vector3** gazebo::physics::Model::GetRelativeAngularVel () const [virtual]

Get the angular velocity of the entity.

Returns

math::Vector3 (p. 1004), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 299).

10.91.3.18 virtual **math::Vector3** gazebo::physics::Model::GetRelativeLinearAccel () const [virtual]

Get the linear acceleration of the entity.

Returns

math::Vector3 (p. 1004), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 299).

10.91.3.19 virtual **math::Vector3** gazebo::physics::Model::GetRelativeLinearVel () const [virtual]

Get the linear velocity of the entity.

Returns

math::Vector3 (p. 1004), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 299).

10.91.3.20 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. 160).

Reimplemented in **gazebo::physics::Actor** (p. 128).

10.91.3.21 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.

10.91.3.22 `virtual math::Vector3 gazebo::physics::Model::GetWorldAngularAccel () const [virtual]`

Get the angular acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 1004), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 299).

10.91.3.23 `virtual math::Vector3 gazebo::physics::Model::GetWorldAngularVel () const [virtual]`

Get the angular velocity of the entity in the world frame.

Returns

math::Vector3 (p. 1004), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 300).

10.91.3.24 `virtual math::Vector3 gazebo::physics::Model::GetWorldLinearAccel () const [virtual]`

Get the linear acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 1004), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 300).

10.91.3.25 `virtual math::Vector3 gazebo::physics::Model::GetWorldLinearVel () const [virtual]`

Get the linear velocity of the entity in the world frame.

Returns

math::Vector3 (p. 1004), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 300).

10.91.3.26 `virtual void gazebo::physics::Model::Init () [virtual]`

Initialize the model.

Reimplemented from **gazebo::physics::Base** (p. 160).

Reimplemented in **gazebo::physics::Actor** (p. 128), and **gazebo::physics::SimbodyModel** (p. 826).

10.91.3.27 `void gazebo::physics::Model::Load (sdf::ElementPtr _sdf) [virtual]`

Load the model.

Parameters

in	_sdf	SDF parameters to load from.
----	------	------------------------------

Reimplemented from **gazebo::physics::Entity** (p. 301).

Reimplemented in **gazebo::physics::SimbodyModel** (p. 826).

10.91.3.28 void gazebo::physics::Model::LoadJoints ()

Load all the joints.

10.91.3.29 void gazebo::physics::Model::LoadPlugins ()

Load all plugins.

Load all plugins specified in the SDF for the model.

10.91.3.30 virtual void gazebo::physics::Model::OnPoseChange () [protected],[virtual]

Callback when the pose of the model has been changed.

Implements **gazebo::physics::Entity** (p. 301).

10.91.3.31 void gazebo::physics::Model::ProcessMsg (const msgs::Model & _msg)

Update parameters from a model message.

Parameters

in	_msg	Message to process.
----	------	---------------------

10.91.3.32 virtual void gazebo::physics::Model::RemoveChild (EntityPtr _child) [virtual]

Remove a child.

Parameters

in	_child	Remove a child entity.
----	--------	------------------------

10.91.3.33 void gazebo::physics::Model::Reset () [virtual]

Reset the model.

Reimplemented from **gazebo::physics::Entity** (p. 301).

10.91.3.34 void gazebo::physics::Model::SetAngularAccel (const math::Vector3 & _vel)

Set the angular acceleration of the model, and all its links.

Parameters

in	<code>_vel</code>	The new angular acceleration
----	-------------------	------------------------------

10.91.3.35 `void gazebo::physics::Model::SetAngularVel (const math::Vector3 & _vel)`

Set the angular velocity of the model, and all its links.

Parameters

in	<code>_vel</code>	The new angular velocity.
----	-------------------	---------------------------

10.91.3.36 `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.
----	-----------------------	--

10.91.3.37 `void gazebo::physics::Model::SetCollideMode (const std::string & _mode)`

This is not implemented in **Link** (p. 455), which means this function doesn't do anything.

Set the collide mode of the model.

Parameters

in	<code>_mode</code>	The collision mode
----	--------------------	--------------------

10.91.3.38 `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.
----	-----------------------	-------------------------------

10.91.3.39 `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.
----	---------------------	---

10.91.3.40 void gazebo::physics::Model::SetJointAnimation (const std::map< std::string, common::NumericAnimationPtr > _anim, boost::function< void()> _onComplete = NULL)

Joint (p. 411) Animation.

Parameters

in	<i>_anim</i>	Map of joint names to their position animation.
in	<i>_onComplete</i>	Callback function for when the animation completes.

10.91.3.41 void gazebo::physics::Model::SetJointPosition (const std::string & _jointName, double _position, int _index = 0)

Set the positions of a **Joint** (p. 411) by name.

See Also

JointController::SetJointPosition (p. 435)

Parameters

in	<i>_jointName</i>	Name of the joint to set.
in	<i>_position</i>	Position to set the joint to.

10.91.3.42 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. 435).

Parameters

in	<i>_jointPositions</i>	Map of joint names to their positions.
----	------------------------	--

10.91.3.43 void gazebo::physics::Model::SetLaserRetro (const float _retro)

Set the laser retro reflectiveness of the model.

Parameters

in	<i>_retro</i>	Retro reflectance value.
----	---------------	--------------------------

10.91.3.44 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.
----	-------------------	------------------------------

10.91.3.45 `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.
----	-------------------	--------------------------

10.91.3.46 `void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & _pose, std::string _linkName)`

Set the Pose of the entire **Model** (p. 537) by specifying desired Pose of a **Link** (p. 455) within the **Model** (p. 537).

Doing so, keeps the configuration of the **Model** (p. 537) unchanged, i.e. all **Joint** (p. 411) angles are unchanged.

Parameters

in	<code>_pose</code>	Pose to set the link to.
in	<code>_linkName</code>	Name of the link to set.

10.91.3.47 `void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & _pose, const LinkPtr & _link)`

Set the Pose of the entire **Model** (p. 537) by specifying desired Pose of a **Link** (p. 455) within the **Model** (p. 537).

Doing so, keeps the configuration of the **Model** (p. 537) unchanged, i.e. all **Joint** (p. 411) angles are unchanged.

Parameters

in	<code>_pose</code>	Pose to set the link to.
in	<code>_link</code>	Pointer to the link to set.

10.91.3.48 `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.
----	---------------------	----------------------------

10.91.3.49 `void gazebo::physics::Model::SetState (const ModelState & _state)`

Set the current model state.

Parameters

in	<code>_state</code>	State (p. 910) to set the model to.
----	---------------------	--

10.91.3.50 virtual void gazebo::physics::Model::StopAnimation () [virtual]

Stop the current animations.

Reimplemented from **gazebo::physics::Entity** (p. 303).

10.91.3.51 void gazebo::physics::Model::Update () [virtual]

Update the model.

Reimplemented from **gazebo::physics::Base** (p. 163).

10.91.3.52 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. 303).

Reimplemented in **gazebo::physics::Actor** (p. 129).

10.91.4 Member Data Documentation

10.91.4.1 `std::vector<ModelPtr>` gazebo::physics::Model::attachedModels [protected]

used by **Model::AttachStaticModel** (p. 541)

10.91.4.2 `std::vector<math::Pose>` gazebo::physics::Model::attachedModelsOffset [protected]

used by **Model::AttachStaticModel** (p. 541)

10.91.4.3 `transport::PublisherPtr` gazebo::physics::Model::jointPub [protected]

Publisher for joint info.

The documentation for this class was generated from the following file:

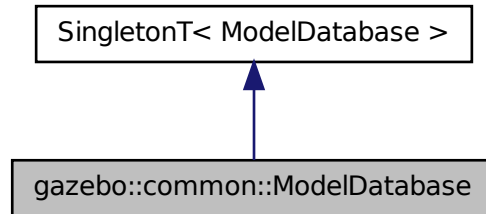
- **Model.hh**

10.92 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.
- void **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

10.92.1 Detailed Description

Connects to model database, and has utility functions to find models.

The documentation for this class was generated from the following file:

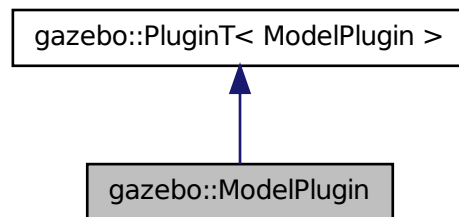
- **ModelDatabase.hh**

10.93 gazebo::ModelPlugin Class Reference

A plugin with access to **physics::Model** (p. 537).

```
#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

10.93.1 Detailed Description

A plugin with access to **physics::Model** (p. 537).

See [reference](#).

10.93.2 Constructor & Destructor Documentation

10.93.2.1 gazebo::ModelPlugin::ModelPlugin () [inline]

Constructor.

References gazebo::MODEL_PLUGIN, and gazebo::PluginT< ModelPlugin >::type.

10.93.2.2 virtual gazebo::ModelPlugin::~~ModelPlugin () [inline],[virtual]

Destructor.

10.93.3 Member Function Documentation

10.93.3.1 virtual void gazebo::ModelPlugin::Init () [inline],[virtual]

Override this method for custom plugin initialization behavior.

10.93.3.2 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	<i>_model</i>	Pointer to the Model
in	<i>_sdf</i>	Pointer to the SDF element of the plugin.

10.93.3.3 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:

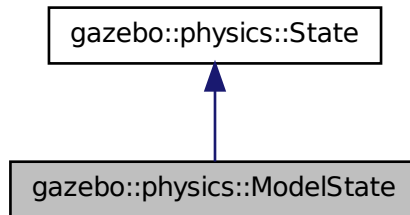
- [Plugin.hh](#)

10.94 gazebo::physics::ModelState Class Reference

Store state information of a [physics::Model](#) (p. 537) 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. 411) state.*
- **JointState GetJointState** (const std::string &_jointName) const
*Get a **Joint** (p. 411) state by **Joint** (p. 411) 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. 455) 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. 537) 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

10.94.1 Detailed Description

Store state information of a **physics::Model** (p. 537) object.

This class captures the entire state of a **Model** (p. 537) at one specific time during a simulation run.

State (p. 910) of a **Model** (p. 537) includes the state of all its child Links and Joints.

10.94.2 Constructor & Destructor Documentation

10.94.2.1 gazebo::physics::ModelState::ModelState ()

Default constructor.

10.94.2.2 `gazebo::physics::ModelState::ModelState (const ModelPtr _model, const common::Time & _realTime, const common::Time & _simTime)`

Constructor.

Build a **ModelState** (p. 554) from an existing **Model** (p. 537).

Parameters

in	<code><i>_model</i></code>	Pointer to the model from which to gather state info.
in	<code><i>_realTime</i></code>	Real time stamp.
in	<code><i>_simTime</i></code>	Sim time stamp.

10.94.2.3 `gazebo::physics::ModelState::ModelState (const ModelPtr _model) [explicit]`

Constructor.

Build a **ModelState** (p. 554) from an existing **Model** (p. 537).

Parameters

in	<code><i>_model</i></code>	Pointer to the model from which to gather state info.
----	----------------------------	---

10.94.2.4 `gazebo::physics::ModelState::ModelState (const sdf::ElementPtr _sdf) [explicit]`

Constructor.

Build a **ModelState** (p. 554) from SDF data

Parameters

in	<code><i>_sdf</i></code>	SDF data to load a model state from.
----	--------------------------	--------------------------------------

10.94.2.5 `virtual gazebo::physics::ModelState::~ModelState () [virtual]`

Destructor.

10.94.3 Member Function Documentation

10.94.3.1 `void gazebo::physics::ModelState::FillSDF (sdf::ElementPtr _sdf)`

Populate a state SDF element with data from the object.

Parameters

out	<code><i>_sdf</i></code>	SDF element to populate.
-----	--------------------------	--------------------------

10.94.3.2 `JointState gazebo::physics::ModelState::GetJointState (unsigned int _index) const`

Get a **Joint** (p. 411) state.

Return a **JointState** (p. 436) based on a index, where index is between 0...**ModelState::GetJointStateCount()** (p. 558).

Parameters

in	_index	Index of a JointState (p. 436).
----	--------	--

Returns

State (p. 910) of a **Joint** (p. 411).

Exceptions

common::Exception (p. 331)	When _index is out of range.
--------------------------------------	------------------------------

10.94.3.3 **JointState** gazebo::physics::ModelState::GetJointState (const std::string & _jointName) const

Get a **Joint** (p. 411) state by **Joint** (p. 411) name.

Searches through all JointStates. Returns the **JointState** (p. 436) with the matching name, if any.

Parameters

in	_jointName	Name of the JointState (p. 436).
----	------------	---

Returns

State (p. 910) of the **Joint** (p. 411).

Exceptions

common::Exception (p. 331)	When _jointName is invalid.
--------------------------------------	-----------------------------

10.94.3.4 unsigned int gazebo::physics::ModelState::GetJointStateCount () const

Get the number of joint states.

Returns the number of JointStates recorded.

Returns

Number of JointStates.

10.94.3.5 **JointState_M** gazebo::physics::ModelState::GetJointStates (const boost::regex & _regex) const

Get joint states based on a regular expression.

Parameters

in	_regex	The regular expression.
----	--------	-------------------------

Returns

List of joint states whose names match the regular expression.

10.94.3.6 `const JointState_M& gazebo::physics::ModelState::GetJointStates () const`

Get the joint states.

Returns

A map of joint states.

10.94.3.7 `LinkState gazebo::physics::ModelState::GetLinkState (const std::string & _linkName) const`

Get a link state by **Link** (p. 455) name.

Searches through all LinkStates. Returns the **LinkState** (p. 476) with the matching name, if any.

Parameters

<code>in</code>	<code>_linkName</code>	Name of the LinkState (p. 476)
-----------------	------------------------	---------------------------------------

Returns

State (p. 910) of the **Link** (p. 455).

Exceptions

<i>common::Exception</i> (p. 331)	When <code>_linkName</code> is invalid.
---	---

10.94.3.8 `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. 476) recorded.

10.94.3.9 `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.

10.94.3.10 `const LinkState_M& gazebo::physics::ModelState::GetLinkStates () const`

Get the link states.

Returns

A map of link states.

10.94.3.11 `const math::Pose& gazebo::physics::ModelState::GetPose () const`

Get the stored model pose.

Returns

The **math::Pose** (p. 648) of the **Model** (p. 537).

10.94.3.12 `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.

10.94.3.13 `bool gazebo::physics::ModelState::HasLinkState (const std::string & _linkName) const`

Return true if there is a link with the specified name.

Parameters

<code>in</code>	<code>_linkName</code>	Name of the LinkState (p. 476).
-----------------	------------------------	--

Returns

True if the link exists in the model.

10.94.3.14 `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.

10.94.3.15 `void gazebo::physics::ModelState::Load (const ModelPtr _model, const common::Time & _realTime, const common::Time & _simTime)`

Load state from **Model** (p. 537) pointer.

Build a **ModelState** (p. 554) from an existing **Model** (p. 537).

Parameters

in	<code>_model</code>	Pointer to the model from which to gather state info.
in	<code>_realTime</code>	Real time stamp.
in	<code>_simTime</code>	Sim time stamp.

10.94.3.16 `virtual void gazebo::physics::ModelState::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Load **ModelState** (p. 554) information from stored data in and SDF::Element

Parameters

in	<code>_elem</code>	Pointer to the SDF::Element containing state info.
----	--------------------	--

Reimplemented from **gazebo::physics::State** (p. 913).

10.94.3.17 `ModelState gazebo::physics::ModelState::operator+ (const ModelState & _state) const`

Addition operator.

Parameters

in	<code>_pt</code>	A state to subtract.
----	------------------	----------------------

Returns

The resulting state.

10.94.3.18 `ModelState gazebo::physics::ModelState::operator- (const ModelState & _state) const`

Subtraction operator.

Parameters

in	<code>_pt</code>	A state to subtract.
----	------------------	----------------------

Returns

The resulting state.

10.94.3.19 **ModelState& gazebo::physics::ModelState::operator= (const ModelState & _state)**

Assignment operator.

Parameters

in	_state	State (p. 910) value
----	--------	-----------------------------

Returns

this

10.94.3.20 **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. 914).

10.94.3.21 **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. 914).

10.94.3.22 **virtual void gazebo::physics::ModelState::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. 910) data was recorded.
----	-------	---

Reimplemented from **gazebo::physics::State** (p. 914).

10.94.4 Friends And Related Function Documentation

10.94.4.1 `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. 537) state to output.

Returns

The stream.

The documentation for this class was generated from the following file:

- **ModelState.hh**

10.95 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.

10.95.1 Detailed Description

Generic description of a mouse event.

10.95.2 Member Enumeration Documentation

10.95.2.1 enum gazebo::common::MouseEvent::Buttons

Standard mouse buttons enumeration.

Enumerator

NO_BUTTON

LEFT

MIDDLE

RIGHT

10.95.2.2 enum gazebo::common::MouseEvent::EventType

Mouse event types enumeration.

Enumerator

NO_EVENT

MOVE

PRESS

RELEASE

SCROLL

10.95.3 Constructor & Destructor Documentation

10.95.3.1 gazebo::common::MouseEvent::MouseEvent () [inline]

Constructor.

10.95.4 Member Data Documentation

10.95.4.1 bool gazebo::common::MouseEvent::alt

Alt key press flag.

10.95.4.2 unsigned int gazebo::common::MouseEvent::button

The button which caused the event.

10.95.4.3 unsigned int gazebo::common::MouseEvent::buttons

State of the buttons when the event was generated.

10.95.4.4 bool gazebo::common::MouseEvent::control

Control key press flag.

10.95.4.5 bool gazebo::common::MouseEvent::dragging

Flag for mouse drag motion.

10.95.4.6 float gazebo::common::MouseEvent::moveScale

Scaling factor.

10.95.4.7 math::Vector2i gazebo::common::MouseEvent::pos

Mouse pointer position on the screen.

10.95.4.8 math::Vector2i gazebo::common::MouseEvent::pressPos

Position of button press.

10.95.4.9 math::Vector2i gazebo::common::MouseEvent::prevPos

Previous position.

10.95.4.10 `math::Vector2i gazebo::common::MouseEvent::scroll`

Scroll position.

10.95.4.11 `bool gazebo::common::MouseEvent::shift`

Shift key press flag.

10.95.4.12 `EventType gazebo::common::MouseEvent::type`

Event type.

The documentation for this class was generated from the following file:

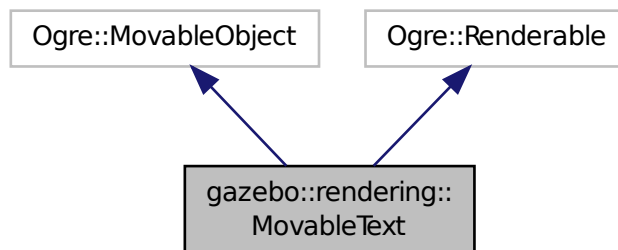
- **MouseEvent.hh**

10.96 gazebo::rendering::MovableText Class Reference

Movable text.

```
#include <rendering/rendering.hh>
```

Inheritance diagram for gazebo::rendering::MovableText:



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

10.96.1 Detailed Description

Movable text.

10.96.2 Member Enumeration Documentation

10.96.2.1 enum gazebo::rendering::MovableText::HorizAlign

Horizontal alignment.

Enumerator

H_LEFT Left alignment.

H_CENTER Center alignment.

10.96.2.2 enum gazebo::rendering::MovableText::VertAlign

vertical alignment

Enumerator

V_BELOW Align below.

V_ABOVE Align above.

10.96.3 Constructor & Destructor Documentation

10.96.3.1 gazebo::rendering::MovableText::MovableText ()

Constructor.

10.96.3.2 virtual gazebo::rendering::MovableText::~MovableText () [virtual]

Destructor.

10.96.4 Member Function Documentation

10.96.4.1 void gazebo::rendering::MovableText::_setupGeometry () [protected]

10.96.4.2 void gazebo::rendering::MovableText::_updateColors () [protected]

10.96.4.3 math::Box gazebo::rendering::MovableText::GetAABB ()

Get the axis aligned bounding box of the text.

Returns

The axis aligned bounding box.

10.96.4.4 float gazebo::rendering::MovableText::GetBaseline () const

Get the baseline height.

Returns

Baseline height

10.96.4.5 float gazebo::rendering::MovableText::getBoundingRadius () const [protected]

10.96.4.6 float gazebo::rendering::MovableText::GetCharHeight () const

Set the height of a characters return Height of the characters.

10.96.4.7 const common::Color& gazebo::rendering::MovableText::GetColor () const

Get the text color.

Returns

Texture color.

10.96.4.8 const std::string& gazebo::rendering::MovableText::GetFont () const

Get the font.

Returns

The font name

10.96.4.9 const Ogre::LightList& gazebo::rendering::MovableText::getLights (void) const [protected]

10.96.4.10 const Ogre::MaterialPtr& gazebo::rendering::MovableText::getMaterial (void) const [protected]

10.96.4.11 void gazebo::rendering::MovableText::getRenderOperation (Ogre::RenderOperation & *op*) [protected]

10.96.4.12 bool gazebo::rendering::MovableText::GetShowOnTop () const

True = text is displayed on top.

Returns

True if MovableText::SetShownOnTop(true) was called.

10.96.4.13 float gazebo::rendering::MovableText::GetSpaceWidth () const

Get the width of a space.

Returns

Space width

10.96.4.14 float gazebo::rendering::MovableText::getSquaredViewDepth (const Ogre::Camera * *cam*) const [protected]

10.96.4.15 const std::string& gazebo::rendering::MovableText::GetText () const

Get the displayed text.

Returns

The displayed text.

10.96.4.16 void gazebo::rendering::MovableText::getWorldTransforms (Ogre::Matrix4 * *xform*) const [protected]

10.96.4.17 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

10.96.4.18 void gazebo::rendering::MovableText::SetBaseline (float *_height*)

Set the baseline height of the text.

Parameters

in	<i>_height</i>	Baseline height
----	----------------	-----------------

10.96.4.19 void gazebo::rendering::MovableText::SetCharHeight (float *_height*)

Set the height of a character.

Parameters

in	<i>_height</i>	Height of the characters.
----	----------------	---------------------------

10.96.4.20 void gazebo::rendering::MovableText::SetColor (const common::Color & *_color*)

Set the text color.

Parameters

in	<i>_color</i>	Text color.
----	---------------	-------------

10.96.4.21 void gazebo::rendering::MovableText::SetFontName (const std::string & *_font*)

Set the font.

Parameters

in	<i>_font</i>	Name of the font
----	--------------	------------------

10.96.4.22 void gazebo::rendering::MovableText::SetShowOnTop (bool *_show*)

True = text always is displayed on top.

Parameters

in	<i>_show</i>	Set to true to render the text on top of all other drawables.
----	--------------	---

10.96.4.23 void gazebo::rendering::MovableText::SetSpaceWidth (float *_width*)

Set the width of a space.

Parameters

in	<i>_width</i>	space width
----	---------------	-------------

10.96.4.24 void gazebo::rendering::MovableText::SetText (const std::string & *_text*)

Set the text to display.

Parameters

in	<code>_text</code>	The text to display.
----	--------------------	----------------------

10.96.4.25 `void gazebo::rendering::MovableText::SetTextAlignment (const HorizAlign & .hAlign, const VertAlign & .vAlign)`

Set the alignment of the text.

Parameters

in	<code>_hAlign</code>	Horizontal alignment
in	<code>_vAlign</code>	Vertical alignment

10.96.4.26 `void gazebo::rendering::MovableText::Update ()`

Update the text.

10.96.4.27 `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**

10.97 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.

10.97.1 Detailed Description

A factory that generates protobuf message based on a string type.

10.97.2 Member Function Documentation

10.97.2.1 `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.
-----	---------------	---

10.97.2.2 `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.

10.97.2.3 `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:

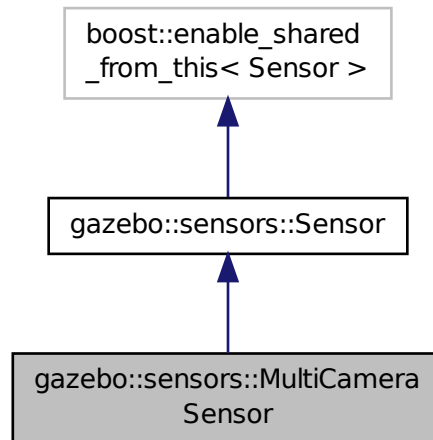
- **MsgFactory.hh**

10.98 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. 179).*
- 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 void **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.98.1 Detailed Description

Multiple camera sensor.

This sensor type can create one or more synchronized cameras.

10.98.2 Constructor & Destructor Documentation

10.98.2.1 gazebo::sensors::MultiCameraSensor::MultiCameraSensor ()

Constructor.

10.98.2.2 virtual gazebo::sensors::MultiCameraSensor::~~MultiCameraSensor () [virtual]

Destructor.

10.98.3 Member Function Documentation

10.98.3.1 virtual void gazebo::sensors::MultiCameraSensor::Fini () [protected],[virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 755).

10.98.3.2 rendering::CameraPtr gazebo::sensors::MultiCameraSensor::GetCamera (unsigned int _index) const

Returns a pointer to a **rendering::Camera** (p. 179).

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. 576)

10.98.3.3 `unsigned int gazebo::sensors::MultiCameraSensor::GetCameraCount () const`

Get the number of cameras.

Returns

The number of cameras.

10.98.3.4 `const unsigned char* gazebo::sensors::MultiCameraSensor::GetImageData (unsigned int _index)`

Gets the raw image data from the sensor.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the camera
-----------------	----------------------------	---------------------

Returns

The pointer to the image data array.

See Also

MultiCameraSensor::GetCameraCount (p. 576)

10.98.3.5 `unsigned int gazebo::sensors::MultiCameraSensor::GetImageHeight (unsigned int _index) const`

Gets the height of the image in pixels.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the camera
-----------------	----------------------------	---------------------

Returns

The image height in pixels.

See Also

MultiCameraSensor::GetCameraCount (p. 576)

10.98.3.6 `unsigned int gazebo::sensors::MultiCameraSensor::GetImageWidth (unsigned int _index) const`

Gets the width of the image in pixels.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the camera
-----------------	----------------------------	---------------------

Returns

The image width in pixels.

See Also

MultiCameraSensor::GetCameraCount (p. 576)

10.98.3.7 `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. 757).

10.98.3.8 `virtual void gazebo::sensors::MultiCameraSensor::Init ()` [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 758).

10.98.3.9 `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. 758).

10.98.3.10 `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. 759).

10.98.3.11 `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. 576)

10.98.3.12 `virtual void 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
----	---------------------	--

Reimplemented from **gazebo::sensors::Sensor** (p. 760).

The documentation for this class was generated from the following file:

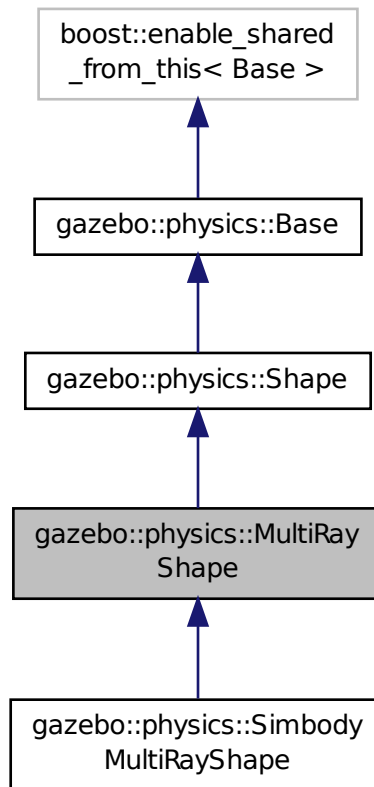
- **MultiCameraSensor.hh**

10.99 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** (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** (int _index)

Get detected range for a ray.
- double **GetResRange** () const

Get the range resolution.
- double **GetRetro** (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

10.99.1 Detailed Description

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

10.99.2 Constructor & Destructor Documentation

10.99.2.1 gazebo::physics::MultiRayShape::MultiRayShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent collision shape.
----	----------------	-------------------------

10.99.2.2 virtual gazebo::physics::MultiRayShape::~~MultiRayShape () [virtual]

Destructor.

10.99.3 Member Function Documentation

10.99.3.1 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::SimbodyMultiRayShape`** (p. 828).

10.99.3.2 `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.

References `gazebo::event::EventT< T >::Connect()`, and `newLaserScans`.

10.99.3.3 `void gazebo::physics::MultiRayShape::DisconnectNewLaserScans (event::ConnectionPtr & _conn) [inline]`

Disconnect from the new laser scans signal.

Parameters

in	<code>_conn</code>	Connection to remove.
----	--------------------	-----------------------

References `gazebo::event::EventT< T >::Disconnect()`, and `newLaserScans`.

10.99.3.4 `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. 777).

10.99.3.5 `int gazebo::physics::MultiRayShape::GetFiducial (int _index)`

Get detected fiducial value for a ray.

Parameters

in	<code>_index</code>	Index of the ray.
----	---------------------	-------------------

Returns

Fiducial value for the ray.

10.99.3.6 math::Angle gazebo::physics::MultiRayShape::GetMaxAngle () const

Get the maximum angle.

Returns

Maximum angle of ray scan.

10.99.3.7 double gazebo::physics::MultiRayShape::GetMaxRange () const

Get the maximum range.

Returns

Maximum range of all the rays.

10.99.3.8 math::Angle gazebo::physics::MultiRayShape::GetMinAngle () const

Get the minimum angle.

Returns

Minimum angle of ray scan.

10.99.3.9 double gazebo::physics::MultiRayShape::GetMinRange () const

Get the minimum range.

Returns

Minimum range of all the rays.

10.99.3.10 double gazebo::physics::MultiRayShape::GetRange (int *_index*)

Get detected range for a ray.

Parameters

<i>in</i>	<i>_index</i>	Index of the ray.
-----------	---------------	-------------------

Returns

Returns DBL_MAX for no detection.

10.99.3.11 `double gazebo::physics::MultiRayShape::GetResRange () const`

Get the range resolution.

Returns

Range resolution of all the rays.

10.99.3.12 `double gazebo::physics::MultiRayShape::GetRetro (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.

10.99.3.13 `int gazebo::physics::MultiRayShape::GetSampleCount () const`

Get the horizontal sample count.

Returns

Horizontal sample count.

10.99.3.14 `double gazebo::physics::MultiRayShape::GetScanResolution () const`

Get the horizontal resolution.

Returns

Horizontal resolution.

10.99.3.15 `math::Angle gazebo::physics::MultiRayShape::GetVerticalMaxAngle () const`

Get the vertical max angle.

Returns

Vertical max angle.

10.99.3.16 `math::Angle gazebo::physics::MultiRayShape::GetVerticalMinAngle () const`

Get the vertical min angle.

Returns

Vertical min angle.

10.99.3.17 `int gazebo::physics::MultiRayShape::GetVerticalSampleCount () const`

Get the vertical sample count.

Returns

Verical sample count.

10.99.3.18 `double gazebo::physics::MultiRayShape::GetVerticalScanResolution () const`

Get the vertical range resolution.

Returns

Vertical range resolution.

10.99.3.19 `virtual void gazebo::physics::MultiRayShape::Init () [virtual]`

Init the shape.

Implements **gazebo::physics::Shape** (p. 777).

10.99.3.20 `virtual void gazebo::physics::MultiRayShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]`

This function is not implemented.

Update the ray based on a message.

Parameters

in	_msg	Message to update from.
----	------	-------------------------

Implements **gazebo::physics::Shape** (p. 778).

10.99.3.21 `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. 778).

10.99.3.22 `void gazebo::physics::MultiRayShape::Update () [virtual]`

Update the ray collisions.

Reimplemented from **gazebo::physics::Base** (p. 163).

10.99.3.23 `virtual void gazebo::physics::MultiRayShape::UpdateRays ()` [protected],[pure virtual]

Physics engine specific method for updating the rays.

Implemented in `gazebo::physics::SimbodyMultiRayShape` (p. 828).

10.99.4 Member Data Documentation

10.99.4.1 `sdf::ElementPtr gazebo::physics::MultiRayShape::horzElem` [protected]

Horizontal SDF element pointer.

10.99.4.2 `event::EventT<void()> gazebo::physics::MultiRayShape::newLaserScans` [protected]

New laser scans event.

Referenced by `ConnectNewLaserScans()`, and `DisconnectNewLaserScans()`.

10.99.4.3 `math::Pose gazebo::physics::MultiRayShape::offset` [protected]

Pose offset of all the rays.

10.99.4.4 `sdf::ElementPtr gazebo::physics::MultiRayShape::rangeElem` [protected]

Range SDF element pointer.

10.99.4.5 `sdf::ElementPtr gazebo::physics::MultiRayShape::rayElem` [protected]

Ray SDF element pointer.

10.99.4.6 `std::vector<RayShapePtr> gazebo::physics::MultiRayShape::rays` [protected]

Ray data.

10.99.4.7 `sdf::ElementPtr gazebo::physics::MultiRayShape::scanElem` [protected]

Scan SDF element pointer.

10.99.4.8 `sdf::ElementPtr gazebo::physics::MultiRayShape::vertElem` [protected]

Vertical SDF element pointer.

The documentation for this class was generated from the following file:

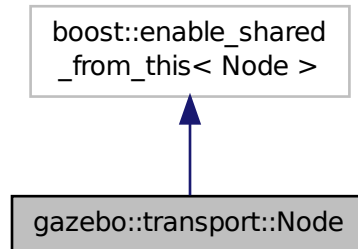
- `MultiRayShape.hh`

10.100 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>
```

Inheritance diagram for gazebo::transport::Node:



Public Member Functions

- **Node** ()
Constructor.
- virtual **~Node** ()
Destructor.
- template<typename M >
transport::PublisherPtr Advertise (const std::string &_topic, unsigned int _queueLimit=1000, double _hz-Rate=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.

10.100.1 Detailed Description

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

10.100.2 Constructor & Destructor Documentation

10.100.2.1 gazebo::transport::Node::Node ()

Constructor.

10.100.2.2 virtual gazebo::transport::Node::~~Node () [virtual]

Destructor.

10.100.3 Member Function Documentation

10.100.3.1 `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 DecodeTopicName(), and SingletonT< T >::Instance().

10.100.3.2 `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

Referenced by Advertise(), and Subscribe().

10.100.3.3 `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

10.100.3.4 `void gazebo::transport::Node::Fini ()`

Finalize the node.

10.100.3.5 `unsigned int gazebo::transport::Node::GetId () const`

Get the unique ID of the node.

Returns

The unique ID of the node

Referenced by `Subscribe()`.

10.100.3.6 `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

10.100.3.7 `std::string gazebo::transport::Node::GetTopicNamespace () const`

Get the topic namespace for this node.

Returns

The namespace

10.100.3.8 `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

10.100.3.9 `bool gazebo::transport::Node::HandleMessage (const std::string & _topic, MessagePtr _msg)`

Handle incoming msg.

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

10.100.3.10 `bool gazebo::transport::Node::HasLatchedSubscriber (const std::string & _topic) const`

Return true if a subscriber on a specific topic is latched.

Parameters

<i>in</i>	<i>_topic</i>	Name of the topic to check.
-----------	---------------	-----------------------------

Returns

True if a latched subscriber exists.

10.100.3.11 `void gazebo::transport::Node::Init (const std::string & _space = " ")`

Init the node.

Parameters

<i>in</i>	<i>_space</i>	Set the global namespace of all topics. If left blank, the topic will initialize to the first namespace on the Master (p. 496)
-----------	---------------	---

10.100.3.12 `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

<i>in</i>	<i>_topic</i>	Name of the topic to publish data on.
<i>in</i>	<i>_msg</i>	The message to publish.

10.100.3.13 `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

<i>in</i>	<i>_topic</i>	Name of the topic to publish data on.
<i>in</i>	<i>_msg</i>	The message to publish.

10.100.3.14 `void gazebo::transport::Node::ProcessIncoming ()`

Process incoming messages.

10.100.3.15 `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

10.100.3.16 `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. 589) followed by **Publisher::Publish** (p. 673). 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

10.100.3.17 `void gazebo::transport::Node::RemoveCallback (const std::string & _topic, unsigned int _id)`

10.100.3.18 `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. 929) object

References `DecodeTopicName()`, `GetId()`, and `SingletonT< T >::Instance()`.

10.100.3.19 `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. 929) object

References DecodeTopicName(), GetId(), and SingletonT< T >::Instance().

10.100.3.20 `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. 929) object

References DecodeTopicName(), GetId(), gazebo::transport::SubscribeOptions::Init(), and SingletonT< T >::Instance().

10.100.3.21 `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. 929) object

References DecodeTopicName(), GetId(), gazebo::transport::SubscribeOptions::Init(), and SingletonT< T >::Instance().

The documentation for this class was generated from the following file:

- **Node.hh**

10.101 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

10.101.1 Detailed Description

Node animation.

10.101.2 Constructor & Destructor Documentation

10.101.2.1 gazebo::common::NodeAnimation::NodeAnimation (const std::string & *_name*)

constructor

Parameters

in	<i>_name</i>	the name of the node
----	--------------	----------------------

10.101.2.2 gazebo::common::NodeAnimation::~~NodeAnimation ()

Destructor. It empties the key frames list.

10.101.3 Member Function Documentation

10.101.3.1 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

10.101.3.2 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

10.101.3.3 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)

10.101.3.4 unsigned int gazebo::common::NodeAnimation::GetFrameCount () const

Returns the number of key frames.

Returns

the count

10.101.3.5 `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

10.101.3.6 `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.944) is -1 if the index is out of bounds

10.101.3.7 `double gazebo::common::NodeAnimation::GetLength () const`

Returns the duration of the animations.

Returns

the time of the last animation

10.101.3.8 `std::string gazebo::common::NodeAnimation::GetName () const`

Returns the name.

Returns

the name

10.101.3.9 `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.
----	----	---

10.101.3.10 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
----	--------	--------------------

10.101.3.11 void gazebo::common::NodeAnimation::SetName (const std::string & *_name*)

Changes the name of the animation.

Parameters

in	<i>the</i>	new name
----	------------	----------

10.101.4 Member Data Documentation

10.101.4.1 std::map<double, math::Matrix4> gazebo::common::NodeAnimation::keyFrames [protected]

the dictionary of key frames, indexed by time

10.101.4.2 double gazebo::common::NodeAnimation::length [protected]

the duration of the animations (time of last key frame)

10.101.4.3 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**

10.102 gazebo::common::NodeAssignment Struct Reference

Vertex to node weighted assignment for skeleton animation visualization.

```
#include <Mesh.hh>
```

Public Attributes

- unsigned int **nodeIndex**

node (or bone) index

- unsigned int **vertexIndex**

index of the vertex

- float **weight**

the weight (between 0 and 1)

10.102.1 Detailed Description

Vertex to node weighted assignment for skeleton animation visualization.

10.102.2 Member Data Documentation

10.102.2.1 unsigned int gazebo::common::NodeAssignment::nodeIndex

node (or bone) index

10.102.2.2 unsigned int gazebo::common::NodeAssignment::vertexIndex

index of the vertex

10.102.2.3 float gazebo::common::NodeAssignment::weight

the weight (between 0 and 1)

The documentation for this struct was generated from the following file:

- **Mesh.hh**

10.103 gazebo::common::NodeTransform Class Reference

NodeTransform (p. 598) **Skeleton.hh** (p. 1268) 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 the 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 rotation.
- **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

10.103.1 Detailed Description

NodeTransform (p. 598) **Skeleton.hh** (p. 1268) common/common.hh

A transformation node

10.103.2 Member Enumeration Documentation

10.103.2.1 enum gazebo::common::NodeTransform::TransformType

Enumeration of the transform types.

Enumerator

TRANSLATE

ROTATE

SCALE

MATRIX

10.103.3 Constructor & Destructor Documentation

10.103.3.1 gazebo::common::NodeTransform::NodeTransform (TransformType _type = MATRIX)

Constructor.

Parameters

in	<i>_type</i>	the type of transform
----	--------------	-----------------------

10.103.3.2 gazebo::common::NodeTransform::NodeTransform (math::Matrix4 _mat, std::string _sid = "_default_", TransformType _type = MATRIX)

Constructor.

Parameters

in	<i>_mat</i>	the matrix
in	<i>_sid</i>	identifier
in	<i>_type</i>	the type of transform

10.103.3.3 gazebo::common::NodeTransform::~~NodeTransform ()

Destructor. It does nothing.

10.103.4 Member Function Documentation

10.103.4.1 math::Matrix4 gazebo::common::NodeTransform::Get ()

Returns the transformation matrix.

Returns

the matrix

10.103.4.2 `std::string gazebo::common::NodeTransform::GetSID ()`

Returns the SID.

Returns

the SID

10.103.4.3 `TransformType gazebo::common::NodeTransform::GetType ()`

Returns the transformation type.

Returns

the type

10.103.4.4 `math::Matrix4 gazebo::common::NodeTransform::operator() ()`

Matrix cast operator.

Returns

the transform

10.103.4.5 `math::Matrix4 gazebo::common::NodeTransform::operator* (NodeTransform _t)`

Node transform multiplication operator.

Parameters

<code>in</code>	<code>_t</code>	a transform
-----------------	-----------------	-------------

Returns

transform matrix multiplied by `_t`'s transform

10.103.4.6 `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`

10.103.4.7 void gazebo::common::NodeTransform::PrintSource ()

Prints the transform matrix to std::err stream.

10.103.4.8 void gazebo::common::NodeTransform::RecalculateMatrix ()

Sets the transform matrix from the source according to the type.

10.103.4.9 void gazebo::common::NodeTransform::Set (math::Matrix4 _mat)

Assign a transformation.

Parameters

in	<i>_mat</i>	the transform
----	-------------	---------------

10.103.4.10 void gazebo::common::NodeTransform::SetComponent (unsigned int *_idx*, double *_value*)

Set a transformation matrix component value.

Parameters

in	<i>_idx</i>	the component index
in	<i>_value</i>	the value

10.103.4.11 void gazebo::common::NodeTransform::SetSID (std::string *_sid*)

Set the SID.

Parameters

in	<i>_sid</i>	the sid
----	-------------	---------

10.103.4.12 void gazebo::common::NodeTransform::SetSourceValues (math::Matrix4 *_mat*)

Set source data values *_param*[in] *_mat* the values.

10.103.4.13 void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 *_vec*)

Set source data values.

10.103.4.14 void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 *_axis*, double *_angle*)

Sets source matrix values from rotation.

Parameters

in	<i>_axis</i>	of rotation
in	<i>_angle</i>	of rotation

10.103.4.15 void gazebo::common::NodeTransform::SetType (TransformType *_type*)

Set transform type.

Parameters

in	<i>_type</i>	the type
----	--------------	----------

10.103.5 Member Data Documentation

10.103.5.1 std::string gazebo::common::NodeTransform::sid [protected]

the sid

10.103.5.2 std::vector<double> gazebo::common::NodeTransform::source [protected]

source data values (can be a matrix, a position or rotation)

10.103.5.3 math::Matrix4 gazebo::common::NodeTransform::transform [protected]

transform

10.103.5.4 TransformType gazebo::common::NodeTransform::type [protected]

transform type

The documentation for this class was generated from the following file:

- **Skeleton.hh**

10.104 gazebo::sensors::Noise Class Reference

Noise (p. 603) models for sensor output signals.

```
#include <sensors/sensors.hh>
```

Public Types

- enum **NoiseType** { **NONE**, **GAUSSIAN**, **GAUSSIAN_QUANTIZED** }

Which noise types we support.

Public Member Functions

- **Noise** ()
Constructor.
- **~Noise** ()
Destructor.
- double **Apply** (double *_in*) const

Apply noise to input data value.

- double **GetBias** () const
Accessor for bias.
- double **GetMean** () const
Accessor for mean.
- **NoiseType GetNoiseType** () const
Accessor for NoiseType.
- double **GetStdDev** () const
Accessor for stddev.
- void **Load** (sdf::ElementPtr _sdf)
Load parameters from sdf.

10.104.1 Detailed Description

Noise (p. 603) models for sensor output signals.

10.104.2 Member Enumeration Documentation

10.104.2.1 enum gazebo::sensors::Noise::NoiseType

Which noise types we support.

Enumerator

NONE

GAUSSIAN

GAUSSIAN_QUANTIZED

10.104.3 Constructor & Destructor Documentation

10.104.3.1 gazebo::sensors::Noise::Noise ()

Constructor.

10.104.3.2 gazebo::sensors::Noise::~~Noise ()

Destructor.

10.104.4 Member Function Documentation

10.104.4.1 double gazebo::sensors::Noise::Apply (double _in) const

Apply noise to input data value.

Parameters

<code>in</code>	<code>_in</code>	Input data value.
-----------------	------------------	-------------------

Returns

Data with noise applied.

10.104.4.2 `double gazebo::sensors::Noise::GetBias () const`

Accessor for bias.

Returns

Bias on output.

10.104.4.3 `double gazebo::sensors::Noise::GetMean () const`

Accessor for mean.

Returns

Mean of Gaussian noise.

10.104.4.4 `NoiseType gazebo::sensors::Noise::GetNoiseType () const`

Accessor for NoiseType.

Returns

Type of noise currently in use.

10.104.4.5 `double gazebo::sensors::Noise::GetStdDev () const`

Accessor for stddev.

Returns

Standard deviation of Gaussian noise.

10.104.4.6 `void gazebo::sensors::Noise::Load (sdf::ElementPtr _sdf)`

Load parameters from sdf.

Parameters

<code>in</code>	<code>_sdf</code>	SDF parameters.
-----------------	-------------------	-----------------

The documentation for this class was generated from the following file:

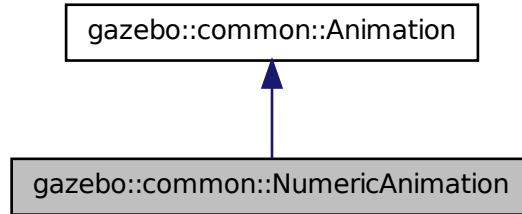
- **Noise.hh**

10.105 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

10.105.1 Detailed Description

A numeric animation.

10.105.2 Constructor & Destructor Documentation

10.105.2.1 `gazebo::common::NumericAnimation::NumericAnimation (const std::string & _name, double _length, bool _loop)`

Constructor.

Parameters

in	<code>_name</code>	String name of the animation. This should be unique.
in	<code>_length</code>	Length of the animation in seconds
in	<code>_loop</code>	True == loop the animation

10.105.2.2 virtual gazebo::common::NumericAnimation::~~NumericAnimation () [virtual]

Destructor.

10.105.3 Member Function Documentation

10.105.3.1 NumericKeyFrame* gazebo::common::NumericAnimation::CreateKeyFrame (double *_time*)

Create a numeric keyframe at the given time.

Parameters

in	<i>_time</i>	Time (p. 944) at which to create the keyframe
----	--------------	--

Returns

Pointer to the new keyframe

10.105.3.2 void gazebo::common::NumericAnimation::GetInterpolatedKeyFrame (NumericKeyFrame & *_kf*) const

Get a keyframe using the animation's current time.

Parameters

out	<i>_kf</i>	NumericKeyFrame (p. 607) reference to hold the interpolated result
-----	------------	---

The documentation for this class was generated from the following file:

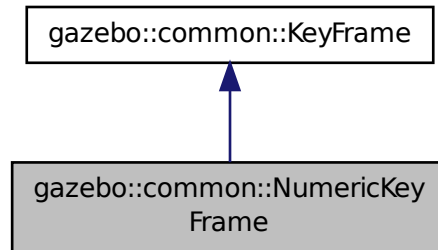
- **Animation.hh**

10.106 gazebo::common::NumericKeyFrame Class Reference

A keyframe for a **NumericAnimation** (p. 606).

```
#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

10.106.1 Detailed Description

A keyframe for a **NumericAnimation** (p. 606).

10.106.2 Constructor & Destructor Documentation

10.106.2.1 gazebo::common::NumericKeyFrame::NumericKeyFrame (double *_time*)

Constructor.

Parameters

in	<i>_time</i>	Time (p. 944) of the keyframe
----	--------------	--------------------------------------

10.106.2.2 virtual gazebo::common::NumericKeyFrame::~~NumericKeyFrame () [virtual]

Destructor.

10.106.3 Member Function Documentation

10.106.3.1 const double& gazebo::common::NumericKeyFrame::GetValue () const

Get the value of the keyframe.

Returns

the value of the keyframe

10.106.3.2 void gazebo::common::NumericKeyFrame::SetValue (const double & *_value*)

Set the value of the keyframe.

Parameters

in	<i>_value</i>	The new value
----	---------------	---------------

10.106.4 Member Data Documentation

10.106.4.1 double gazebo::common::NumericKeyFrame::value [protected]

numeric value

The documentation for this class was generated from the following file:

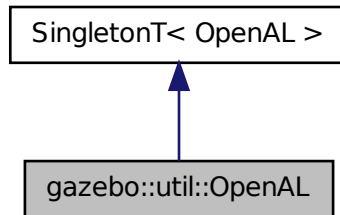
- **KeyFrame.hh**

10.107 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. 612) object.*
- void **Fini** ()
Finalize.
- bool **Load** (sdf::ElementPtr _sdf=sdf::ElementPtr())
*Load the **OpenAL** (p. 609) server.*

Additional Inherited Members

10.107.1 Detailed Description

3D audio setup and playback.

10.107.2 Member Function Documentation

10.107.2.1 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. 611) object.

10.107.2.2 **OpenALSourcePtr** gazebo::util::OpenAL::CreateSource (sdf::ElementPtr _sdf)

Create an **OpenALSource** (p. 612) object.

Parameters

in	_sdf	SDF element parameters for an audio_source.
----	------	---

Returns

A pointer to an **OpenALSource** (p. 612) object.

10.107.2.3 void gazebo::util::OpenAL::Fini ()

Finalize.

10.107.2.4 bool gazebo::util::OpenAL::Load (sdf::ElementPtr _sdf = sdf::ElementPtr())

Load the **OpenAL** (p. 609) server.

Returns

True on success.

The documentation for this class was generated from the following file:

- **OpenAL.hh**

10.108 gazebo::util::OpenALSink Class Reference

OpenAL (p. 609) 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.

10.108.1 Detailed Description

OpenAL (p. 609) Listener.

This can be thought of as a microphone.

10.108.2 Constructor & Destructor Documentation

10.108.2.1 gazebo::util::OpenALSink::OpenALSink ()

Constructor.

10.108.2.2 virtual gazebo::util::OpenALSink::~~OpenALSink () [virtual]

Destructor.

10.108.3 Member Function Documentation

10.108.3.1 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.

10.108.3.2 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**

10.109 gazebo::util::OpenALSource Class Reference

OpenAL (p. 609) Source.

```
#include <OpenAL.hh>
```

Public Member Functions

- **OpenALSource** ()

Constructor.

- virtual **~OpenALSource** ()

Destructor.

- void **FillBufferFromFile** (const std::string &_audioFile)

*Fill the **OpenAL** (p. 609) audio buffer with data from a sound file.*

- bool **FillBufferFromPCM** (uint8_t *_pcmData, unsigned int _dataCount, int _sampleRate)

*Fill the **OpenAL** (p. 609) 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.

10.109.1 Detailed Description

OpenAL (p. 609) Source.

This can be thought of as a speaker.

10.109.2 Constructor & Destructor Documentation

10.109.2.1 gazebo::util::OpenALSource::OpenALSource ()

Constructor.

10.109.2.2 `virtual gazebo::util::OpenALSource::~~OpenALSource () [virtual]`

Destructor.

10.109.3 Member Function Documentation

10.109.3.1 `void gazebo::util::OpenALSource::FillBufferFromFile (const std::string & _audioFile)`

Fill the **OpenAL** (p. 609) audio buffer with data from a sound file.

Parameters

<code>in</code>	<code>_audioFile</code>	Name and an audio file.
-----------------	-------------------------	-------------------------

10.109.3.2 `bool gazebo::util::OpenALSource::FillBufferFromPCM (uint8_t * _pcmData, unsigned int _dataCount, int _sampleRate)`

Fill the **OpenAL** (p. 609) 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.

10.109.3.3 `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.

10.109.3.4 `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()

10.109.3.5 `bool gazebo::util::OpenALSource::HasCollisionName (const std::string & _name) const`

Get whether the source has a collision name set.

Parameters

<code>in</code>	<code><i>_name</i></code>	Name of a collision to check for.
-----------------	---------------------------	-----------------------------------

Returns

True if the collision name was found.

10.109.3.6 `bool gazebo::util::OpenALSource::IsPlaying ()`

Is the audio playing.

10.109.3.7 `bool gazebo::util::OpenALSource::Load (sdf::ElementPtr _sdf)`

Load the source from sdf.

Parameters

<code>in</code>	<code><i>_sdf</i></code>	SDF element parameters for an audio_source.
-----------------	--------------------------	---

Returns

True on success.

10.109.3.8 `void gazebo::util::OpenALSource::Pause ()`

Pause a sound.

10.109.3.9 `void gazebo::util::OpenALSource::Play ()`

Play a sound.

10.109.3.10 `void gazebo::util::OpenALSource::Rewind ()`

Rewind the sound to the beginning.

10.109.3.11 `bool gazebo::util::OpenALSource::SetGain (float _g)`

Set the pitch of the source.

Parameters

<code>in</code>	<code><i>_g</i></code>	Gain value.
-----------------	------------------------	-------------

Returns

True on success.

10.109.3.12 `bool gazebo::util::OpenALSource::SetLoop (bool _state)`

Set whether the source loops the audio.

Parameters

<code>in</code>	<code>_state</code>	True to cause playback to loop.
-----------------	---------------------	---------------------------------

Returns

True on success.

10.109.3.13 `bool gazebo::util::OpenALSource::SetPitch (float _p)`

Set the pitch of the source.

Parameters

<code>in</code>	<code>_p</code>	Pitch value.
-----------------	-----------------	--------------

Returns

True on success.

10.109.3.14 `bool gazebo::util::OpenALSource::SetPose (const math::Pose & _pose)`

Set the position of the source.

Parameters

<code>in</code>	<code>_pose</code>	New pose of the source.
-----------------	--------------------	-------------------------

Returns

True on success.

10.109.3.15 `bool gazebo::util::OpenALSource::SetVelocity (const math::Vector3 & _vel)`

Set the velocity of the source.

Parameters

<code>in</code>	<code>_vel</code>	New velocity of the source.
-----------------	-------------------	-----------------------------

Returns

True on success.

10.109.3.16 void gazebo::util::OpenALSource::Stop ()

Stop a sound.

The documentation for this class was generated from the following file:

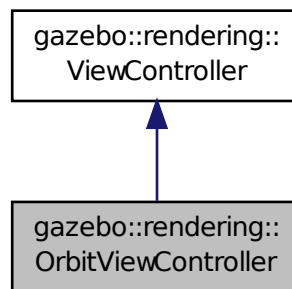
- **OpenAL.hh**

10.110 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

10.110.1 Detailed Description

Orbit view controller.

10.110.2 Constructor & Destructor Documentation

10.110.2.1 gazebo::rendering::OrbitViewController::OrbitViewController (**UserCameraPtr** _camera)

Constructor.

Parameters

in	_camera	Pointer to the camera to control.
----	----------------	-----------------------------------

10.110.2.2 virtual gazebo::rendering::OrbitViewController::~~OrbitViewController () [virtual]

Destructor.

10.110.3 Member Function Documentation

10.110.3.1 **math::Vector3** gazebo::rendering::OrbitViewController::GetFocalPoint () const

Get the focal point.

Returns

The focal point

10.110.3.2 `static std::string gazebo::rendering::OrbitViewController::GetTypeString () [static]`

Get the type name of this view controller.

Returns

The view controller name: "orbit".

10.110.3.3 `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. 1032).

10.110.3.4 `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. 1033).

10.110.3.5 `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. 1033).

10.110.3.6 `virtual void gazebo::rendering::OrbitViewController::Init () [virtual]`

Initialize the controller.

Implements `gazebo::rendering::ViewController` (p. 1033).

10.110.3.7 `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. 1033).

10.110.3.8 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.
----	-----------------	------------------------------------

10.110.3.9 void gazebo::rendering::OrbitViewController::SetFocalPoint (const math::Vector3 & *fp*)

Set the focal point.

Parameters

in	<code>_fp</code>	The focal point
----	------------------	-----------------

10.110.3.10 virtual void gazebo::rendering::OrbitViewController::Update () [virtual]

Update.

Implements **gazebo::rendering::ViewController** (p. 1034).

The documentation for this class was generated from the following file:

- **OrbitViewController.hh**

10.111 gazebo::common::ParamT< T > Class Template Reference

```
#include <CommonTypes.hh>
```

The documentation for this class was generated from the following file:

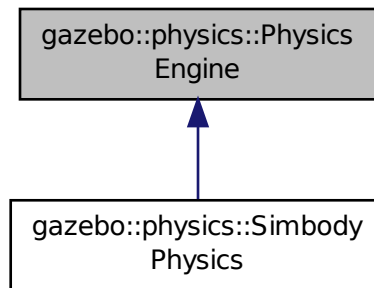
- **CommonTypes.hh**

10.112 gazebo::physics::PhysicsEngine Class Reference

Base (p. 153) 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. 775) 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 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** (std::string _key) const

Get an parameter of the physics engine.
- boost::recursive_mutex * **GetPhysicsUpdateMutex** () const

*returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 634).*
- double **GetRealTimeUpdateRate** () const

Get real time update rate.
- virtual int **GetSORPGSIters** ()

: Remove this function, and replace it with a more generic property map
- virtual int **GetSORPGSPreconlters** ()

: Remove this function, and replace it with a more generic property map
- virtual double **GetSORPGSW** ()

: 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 type of the physics engine (ode|bullet|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** (double _maxContacts)

: Remove this function, and replace it with a more generic property map

- void **SetMaxStepSize** (double _stepSize)
Set max step size.
- virtual void **SetParam** (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)
: Remove this function, and replace it with a more generic property map
- virtual void **SetSORPGSPreconIters** (unsigned int _iters)
: Remove this function, and replace it with a more generic property map
- virtual void **SetSORPGSW** (double _w)
: 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.

10.112.1 Detailed Description

Base (p. 153) class for a physics engine.

10.112.2 Constructor & Destructor Documentation

10.112.2.1 gazebo::physics::PhysicsEngine::PhysicsEngine (WorldPtr _world) [explicit]

Default constructor.

Parameters

in	_world	Pointer to the world.
----	--------	-----------------------

10.112.2.2 virtual gazebo::physics::PhysicsEngine::~PhysicsEngine () [virtual]

Destructor.

10.112.3 Member Function Documentation

10.112.3.1 virtual CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision (const std::string & _shapeType, LinkPtr _link) [pure virtual]

Create a collision.

Parameters

in	_shapeType	Type of collision to create.
in	_link	Parent link.

Implemented in **gazebo::physics::SimbodyPhysics** (p. 831).

10.112.3.2 CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision (const std::string & _shapeType, const std::string & _linkName)

Create a collision.

Parameters

in	<i>_shapeType</i>	Type of collision to create.
in	<i>_linkName</i>	Name of the parent link.

10.112.3.3 virtual `JointPtr` gazebo::physics::PhysicsEngine::CreateJoint (const std::string & *_type*, `ModelPtr` *_parent* = `ModelPtr` ()) [pure virtual]

Create a new joint.

Parameters

in	<i>_type</i>	Type of joint to create.
in	<i>_parent</i>	Model (p. 537) parent.

Implemented in `gazebo::physics::SimbodyPhysics` (p. 831).

10.112.3.4 virtual `LinkPtr` gazebo::physics::PhysicsEngine::CreateLink (`ModelPtr` *_parent*) [pure virtual]

Create a new body.

Parameters

in	<i>_parent</i>	Parent model for the link.
----	----------------	----------------------------

Implemented in `gazebo::physics::SimbodyPhysics` (p. 832).

10.112.3.5 virtual `ModelPtr` gazebo::physics::PhysicsEngine::CreateModel (`BasePtr` *_base*) [virtual]

Create a new model.

Parameters

in	<i>_base</i>	Boost shared pointer to a new model.
----	--------------	--------------------------------------

Reimplemented in `gazebo::physics::SimbodyPhysics` (p. 832).

10.112.3.6 virtual `ShapePtr` gazebo::physics::PhysicsEngine::CreateShape (const std::string & *_shapeType*, `CollisionPtr` *_collision*) [pure virtual]

Create a `physics::Shape` (p. 775) object.

Parameters

in	<i>_shapeType</i>	Type of shape to create.
in	<i>_collision</i>	Collision (p. 213) parent.

Implemented in `gazebo::physics::SimbodyPhysics` (p. 832).

10.112.3.7 `virtual void gazebo::physics::PhysicsEngine::DebugPrint () const [pure virtual]`

Debug print out of the physic engine state.

Implemented in `gazebo::physics::SimbodyPhysics` (p. 832).

10.112.3.8 `virtual void gazebo::physics::PhysicsEngine::Fini () [virtual]`

Finilize the physics engine.

Reimplemented in `gazebo::physics::SimbodyPhysics` (p. 832).

10.112.3.9 `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.

10.112.3.10 `ContactManager* gazebo::physics::PhysicsEngine::GetContactManager () const`

Get a pointer to the contact manger.

Returns

Pointer to the contact manager.

10.112.3.11 `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.

10.112.3.12 `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. 253) suerface layer depth.

10.112.3.13 `virtual math::Vector3 gazebo::physics::PhysicsEngine::GetGravity () const [virtual]`

Return the gavity vector.

Returns

The gavity vector.

10.112.3.14 `virtual 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.

10.112.3.15 `double gazebo::physics::PhysicsEngine::GetMaxStepSize () const`

Get max step size.

Returns

Max step size.

10.112.3.16 `virtual boost::any gazebo::physics::PhysicsEngine::GetParam (std::string _key) const [virtual]`

Get an parameter of the physics engine.

Parameters

<code>in</code>	<code>_attr</code>	String key
-----------------	--------------------	------------

Returns

The value of the parameter

10.112.3.17 `boost::recursive_mutex* gazebo::physics::PhysicsEngine::GetPhysicsUpdateMutex () const [inline]`

returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 634).

Returns

Pointer to the physics mutex.

References physicsUpdateMutex.

10.112.3.18 `double gazebo::physics::PhysicsEngine::GetRealTimeUpdateRate () const`

Get real time update rate.

Returns

Update rate

10.112.3.19 `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.

10.112.3.20 `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.

10.112.3.21 `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.

10.112.3.22 `double gazebo::physics::PhysicsEngine::GetTargetRealTimeFactor () const`

Get target real time factor.

Returns

Target real time factor

10.112.3.23 `virtual std::string gazebo::physics::PhysicsEngine::GetType () const [pure virtual]`

Return the type of the physics engine (ode|bullet|simbody).

Returns

Type of the physics engine.

Implemented in `gazebo::physics::SimbodyPhysics` (p. 833).

10.112.3.24 `double gazebo::physics::PhysicsEngine::GetUpdatePeriod ()`

Get the simulation update period.

Returns

Simulation update period.

10.112.3.25 `virtual double gazebo::physics::PhysicsEngine::GetWorldCFM ()` `[inline],[virtual]`

: Remove this function, and replace it with a more generic property map

Get **World** (p. 1070) CFM.

Returns

World (p. 1070) CFM.

10.112.3.26 `virtual double gazebo::physics::PhysicsEngine::GetWorldERP ()` `[inline],[virtual]`

: Remove this function, and replace it with a more generic property map

Get **World** (p. 1070) ERP.

Returns

World (p. 1070) ERP.

10.112.3.27 `virtual void gazebo::physics::PhysicsEngine::Init ()` `[pure virtual]`

Initialize the physics engine.

Implemented in **gazebo::physics::SimbodyPhysics** (p. 833).

10.112.3.28 `virtual void gazebo::physics::PhysicsEngine::InitForThread ()` `[pure virtual]`

Init the engine for threads.

Implemented in **gazebo::physics::SimbodyPhysics** (p. 833).

10.112.3.29 `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::SimbodyPhysics** (p. 834).

10.112.3.30 `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. 834).

10.112.3.31 `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. 834).

10.112.3.32 `virtual void gazebo::physics::PhysicsEngine::Reset ()` [inline], [virtual]

Rest the physics engine.

Reimplemented in `gazebo::physics::SimbodyPhysics` (p. 835).

10.112.3.33 `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.
----	---------------------------	--

10.112.3.34 `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.
----	-------------------	--------------------------

10.112.3.35 `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
----	--------------------	---------------------

10.112.3.36 `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::SimbodyPhysics` (p. 835).

10.112.3.37 `virtual void gazebo::physics::PhysicsEngine::SetMaxContacts (double _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.
----	---------------------	-----------------------------

10.112.3.38 `void gazebo::physics::PhysicsEngine::SetMaxStepSize (double _stepSize)`

Set max step size.

Parameters

in	<i>_stepSize</i>	Max step size.
----	------------------	----------------

10.112.3.39 `virtual void gazebo::physics::PhysicsEngine::SetParam (std::string _key, const boost::any & _value) [virtual]`

Set a parameter of the physics engine.

Parameters

in	<i>_key</i>	String key
in	<i>_value</i>	The value to set to

10.112.3.40 `void gazebo::physics::PhysicsEngine::SetRealTimeUpdateRate (double _rate)`

Set real time update rate.

Parameters

in	<code>_rate</code>	Update rate
----	--------------------	-------------

10.112.3.41 `virtual void gazebo::physics::PhysicsEngine::SetSeed (uint32_t _seed) [pure virtual]`

Set the random number seed for the physics engine.

Parameters

in	<code>_seed</code>	The random number seed.
----	--------------------	-------------------------

Implemented in `gazebo::physics::SimbodyPhysics` (p. 835).

10.112.3.42 `virtual void gazebo::physics::PhysicsEngine::SetSORPGSIters (unsigned int _iters) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	<code>_iter</code>	Number of iterations.
----	--------------------	-----------------------

10.112.3.43 `virtual void gazebo::physics::PhysicsEngine::SetSORPGSPreconIters (unsigned int _iters) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	<code>_iter</code>	Number of iterations.
----	--------------------	-----------------------

10.112.3.44 `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

in	<code>_w</code>	SORPGSW value.
----	-----------------	----------------

10.112.3.45 `void gazebo::physics::PhysicsEngine::SetTargetRealTimeFactor (double _factor)`

Set target real time factor.

Parameters

in	<code>_factor</code>	Target real time factor
----	----------------------	-------------------------

10.112.3.46 `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

in	<code>_cfm</code>	Constraint force mixing.
----	-------------------	--------------------------

10.112.3.47 `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	<code>_erp</code>	Error reduction parameter.
----	-------------------	----------------------------

10.112.3.48 `virtual void gazebo::physics::PhysicsEngine::UpdateCollision () [pure virtual]`

Update the physics engine collision.

Implemented in `gazebo::physics::SimbodyPhysics` (p. 836).

10.112.3.49 `virtual void gazebo::physics::PhysicsEngine::UpdatePhysics () [inline],[virtual]`

Update the physics engine.

Reimplemented in `gazebo::physics::SimbodyPhysics` (p. 836).

10.112.4 Member Data Documentation

10.112.4.1 `ContactManager* gazebo::physics::PhysicsEngine::contactManager [protected]`

Class that handles all contacts generated by the physics engine.

10.112.4.2 `double gazebo::physics::PhysicsEngine::maxStepSize [protected]`

Real time update rate.

10.112.4.3 `transport::NodePtr gazebo::physics::PhysicsEngine::node [protected]`

Node for communication.

10.112.4.4 `transport::SubscriberPtr gazebo::physics::PhysicsEngine::physicsSub` [protected]

Subscribe to the physics topic.

10.112.4.5 `boost::recursive_mutex* gazebo::physics::PhysicsEngine::physicsUpdateMutex` [protected]

Mutex to protect the update cycle.

Referenced by `GetPhysicsUpdateMutex()`.

10.112.4.6 `double gazebo::physics::PhysicsEngine::realTimeUpdateRate` [protected]

Real time update rate.

10.112.4.7 `transport::SubscriberPtr gazebo::physics::PhysicsEngine::requestSub` [protected]

Subscribe to the request topic.

10.112.4.8 `transport::PublisherPtr gazebo::physics::PhysicsEngine::responsePub` [protected]

Response publisher.

10.112.4.9 `sdf::ElementPtr gazebo::physics::PhysicsEngine::sdf` [protected]

Our SDF values.

10.112.4.10 `double gazebo::physics::PhysicsEngine::targetRealTimeFactor` [protected]

Target real time factor.

10.112.4.11 `WorldPtr gazebo::physics::PhysicsEngine::world` [protected]

Pointer to the world.

The documentation for this class was generated from the following file:

- **PhysicsEngine.hh**

10.113 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.

10.113.1 Detailed Description

The physics factory instantiates different physics engines.

10.113.2 Member Function Documentation

10.113.2.1 static bool gazebo::physics::PhysicsFactory::IsRegistered (const std::string & _name) [static]

Check if a physics engine is registered.

Parameters

in	<code>_name</code>	Name of the physics engine.
----	--------------------	-----------------------------

Returns

True if physics engine is registered, false otherwise.

10.113.2.2 static PhysicsEnginePtr gazebo::physics::PhysicsFactory::NewPhysicsEngine (const std::string & _className, **WorldPtr** _world) [static]

Create a new instance of a physics engine.

Parameters

in	<code>_className</code>	Name of the physics class.
in	<code>_world</code>	World (p. 1070) to pass to the created physics engine.

10.113.2.3 static void gazebo::physics::PhysicsFactory::RegisterAll () [static]

Register everything.

10.113.2.4 static void gazebo::physics::PhysicsFactory::RegisterPhysicsEngine (std::string _className, **PhysicsFactoryFn** _factoryfn) [static]

Register a physics class.

Parameters

in	<code>_className</code>	Name of the physics class.
in	<code>_factoryfn</code>	Function pointer used to create a physics engine.

The documentation for this class was generated from the following file:

- **PhysicsFactory.hh**

10.114 gazebo::common::PID Class Reference

Generic **PID** (p. 636) 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 **~PID** ()
Destructor.
- double **GetCmd** ()
*Return current command for this **PID** (p. 636) controller.*
- void **GetErrors** (double &_pe, double &_ie, double &_de)
*Return **PID** (p. 636) error terms for the controller.*
- 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. 636) 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 derivative 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.

10.114.1 Detailed Description

Generic **PID** (p. 636) 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.

10.114.2 Constructor & Destructor Documentation

10.114.2.1 `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]-[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.

10.114.2.2 `virtual gazebo::common::PID::~PID () [virtual]`

Destructor.

10.114.3 Member Function Documentation

10.114.3.1 `double gazebo::common::PID::GetCmd ()`

Return current command for this **PID** (p. 636) controller.

Returns

the command value

10.114.3.2 `void gazebo::common::PID::GetErrors (double & _pe, double & _ie, double & _de)`

Return **PID** (p. 636) error terms for the controller.

Parameters

in	<code>_pe</code>	The proportional error.
in	<code>_ie</code>	The integral error.
in	<code>_de</code>	The derivative error.

10.114.3.3 `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]-[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.

10.114.3.4 `PID& gazebo::common::PID::operator=(const PID & _p) [inline]`

Assignment operator.

Parameters

in	<code>_p</code>	a reference to a PID (p. 636) to assign values from
----	-----------------	--

Returns

reference to this instance

References Reset().

10.114.3.5 `void gazebo::common::PID::Reset ()`

Reset the errors and command.

Referenced by operator=().

10.114.3.6 `void gazebo::common::PID::SetCmd (double _cmd)`

Set current target command for this **PID** (p. 636) controller.

Parameters

in	<code>_cmd</code>	New command
----	-------------------	-------------

10.114.3.7 `void gazebo::common::PID::SetCmdMax (double _c)`

Set the maximum value for the command.

Parameters

in	<code>_c</code>	The maximum value
----	-----------------	-------------------

10.114.3.8 void gazebo::common::PID::SetCmdMin (double *_c*)

Set the maximum value for the command.

Parameters

<i>in</i>	<i>_c</i>	The maximum value
-----------	-----------	-------------------

10.114.3.9 void gazebo::common::PID::SetDGain (double *_d*)

Set the derivative Gain.

Parameters

<i>in</i>	<i>_p</i>	derivative gain value
-----------	-----------	-----------------------

10.114.3.10 void gazebo::common::PID::SetIGain (double *_i*)

Set the integral Gain.

Parameters

<i>in</i>	<i>_p</i>	integral gain value
-----------	-----------	---------------------

10.114.3.11 void gazebo::common::PID::SetIMax (double *_i*)

Set the integral upper limit.

Parameters

<i>in</i>	<i>_p</i>	integral upper limit value
-----------	-----------	----------------------------

10.114.3.12 void gazebo::common::PID::SetIMin (double *_i*)

Set the integral lower limit.

Parameters

<i>in</i>	<i>_p</i>	integral lower limit value
-----------	-----------	----------------------------

10.114.3.13 void gazebo::common::PID::SetPGain (double *_p*)

Set the proportional Gain.

Parameters

<i>in</i>	<i>_p</i>	proportional gain value
-----------	-----------	-------------------------

10.114.3.14 `double gazebo::common::PID::Update (double _error, common::Time _dt)`

Update the Pid loop with nonuniform time step size.

Parameters

<code>_in]</code>	<code>_error</code> Error since last call (p_state - p_target).
<code>_in]</code>	<code>_dt</code> 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**

10.115 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. 640) offset.
- **Vector3** **normal**
Plane (p. 640) normal.
- **Vector2d** **size**
Plane (p. 640) size.

10.115.1 Detailed Description

A plane and related functions.

10.115.2 Constructor & Destructor Documentation

10.115.2.1 gazebo::math::Plane::Plane ()

Constructor.

10.115.2.2 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

10.115.2.3 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

10.115.2.4 virtual gazebo::math::Plane::~~Plane () [virtual]

Destructor.

10.115.3 Member Function Documentation

10.115.3.1 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

10.115.3.2 `Plane& gazebo::math::Plane::operator= (const Plane & _p)`

Equal operator.

Parameters

<code>_p</code>	another plane
-----------------	---------------

Returns

itself

10.115.3.3 `void gazebo::math::Plane::Set (const Vector3 & _normal, const Vector2d & _size, double offset)`

Set the plane.

Parameters

<code>in</code>	<code>_normal</code>	The plane normal
<code>in</code>	<code>_size</code>	Size of the plane
<code>in</code>	<code>_offset</code>	Offset along the normal

10.115.4 Member Data Documentation

10.115.4.1 `double gazebo::math::Plane::d`

Plane (p. 640) offset.

10.115.4.2 `Vector3 gazebo::math::Plane::normal`

Plane (p. 640) normal.

10.115.4.3 `Vector2d gazebo::math::Plane::size`

Plane (p. 640) size.

The documentation for this class was generated from the following file:

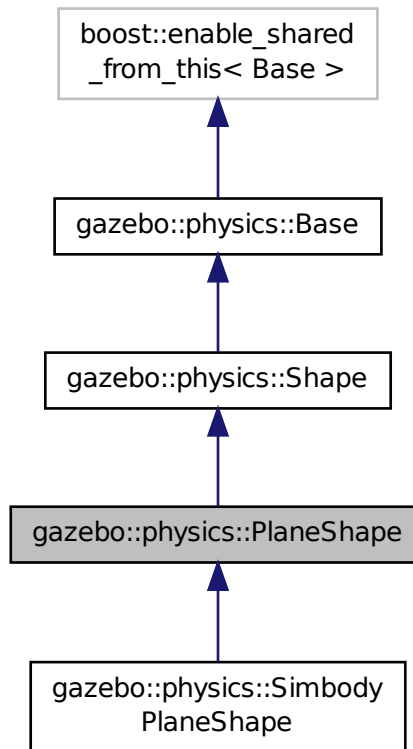
- **Plane.hh**

10.116 `gazebo::physics::PlaneShape` Class Reference

Collision (p. 213) 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

10.116.1 Detailed Description

Collision (p. 213) 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.

10.116.2 Constructor & Destructor Documentation

10.116.2.1 `gazebo::physics::PlaneShape::PlaneShape (CollisionPtr _parent) [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Link (p. 455) to which we are attached.
----	----------------------	--

10.116.2.2 `virtual gazebo::physics::PlaneShape::~~PlaneShape () [virtual]`

Destructor.

10.116.3 Member Function Documentation

10.116.3.1 `virtual void gazebo::physics::PlaneShape::CreatePlane () [virtual]`

Create the plane.

Reimplemented in **gazebo::physics::SimbodyPlaneShape** (p. 839).

10.116.3.2 `void gazebo::physics::PlaneShape::FillMsg (msgs::Geometry & _msg) [virtual]`

Fill a geometry message with data from this object.

Parameters

out	<code>_msg</code>	Message to fill.
-----	-------------------	------------------

Implements **gazebo::physics::Shape** (p. 777).

10.116.3.3 **math::Vector3** gazebo::physics::PlaneShape::GetNormal () const

Get the plane normal.

Returns

The plane normal.

10.116.3.4 **math::Vector2d** gazebo::physics::PlaneShape::GetSize () const

Get the size.

Returns

Size of the plane.

10.116.3.5 **virtual void** gazebo::physics::PlaneShape::Init () [virtual]

Initialize the plane.

Implements **gazebo::physics::Shape** (p. 777).

10.116.3.6 **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	<i>_msg</i>	Message to update from.
----	-------------	-------------------------

Implements **gazebo::physics::Shape** (p. 778).

10.116.3.7 **virtual void** gazebo::physics::PlaneShape::SetAltitude (const math::Vector3 & _pos) [virtual]

Set the altitude of the plane.

Parameters

in	<i>_pos</i>	Position of the plane.
----	-------------	------------------------

Reimplemented in **gazebo::physics::SimbodyPlaneShape** (p. 839).

10.116.3.8 **void** gazebo::physics::PlaneShape::SetNormal (const math::Vector3 & _norm)

Set the normal.

Parameters

in	_norm	Plane normal.
----	-------	---------------

10.116.3.9 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. 778).

10.116.3.10 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**

10.117 gazebo::PluginT< T > Class Template Reference

A class which all plugins must inherit from.

```
#include <common/common.hh>
```

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.

10.117.1 Detailed Description

template<class T>class gazebo::PluginT< T >

A class which all plugins must inherit from.

10.117.2 Member Typedef Documentation

10.117.2.1 template<class T> typedef boost::shared_ptr<T> gazebo::PluginT< T >::TPtr

plugin pointer type definition

10.117.3 Constructor & Destructor Documentation

10.117.3.1 template<class T> gazebo::PluginT< T >::PluginT () [inline]

Constructor.

10.117.3.2 template<class T> virtual gazebo::PluginT< T >::~~PluginT () [inline],[virtual]

Destructor.

10.117.4 Member Function Documentation

10.117.4.1 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	<i>_filename</i>	the path to the shared library.
in	<i>_handle</i>	short name of the handler

Returns

Shared Pointer to this class type

10.117.4.2 `template<class T> std::string gazebo::PluginT< T >::GetFilename () const [inline]`

Get the name of the handler.

10.117.4.3 `template<class T> std::string gazebo::PluginT< T >::GetHandle () const [inline]`

Get the short name of the handler.

10.117.4.4 `template<class T> PluginType gazebo::PluginT< T >::GetType () const [inline]`

Returns the type of the plugin.

Returns

type of the plugin

10.117.5 Member Data Documentation

10.117.5.1 `template<class T> std::string gazebo::PluginT< T >::filename [protected]`

Path to the shared library file.

Referenced by `gazebo::PluginT< ModelPlugin >::GetFilename()`.

10.117.5.2 `template<class T> std::string gazebo::PluginT< T >::handle [protected]`

Short name.

Referenced by `gazebo::PluginT< ModelPlugin >::Create()`, and `gazebo::PluginT< ModelPlugin >::GetHandle()`.

10.117.5.3 `template<class T> PluginType gazebo::PluginT< T >::type [protected]`

Type of plugin.

Referenced by `gazebo::PluginT< ModelPlugin >::GetType()`.

The documentation for this class was generated from the following file:

- **Plugin.hh**

10.118 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 \sim **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. 1004) and a **Quaternion** (p. 675).*
- 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.

10.118.1 Detailed Description

Encapsulates a position and rotation in three space.

10.118.2 Constructor & Destructor Documentation

10.118.2.1 gazebo::math::Pose::Pose ()

Default constructors.

Referenced by operator-().

10.118.2.2 gazebo::math::Pose::Pose (const Vector3 & *_pos*, const Quaternion & *_rot*)

Constructor.

Parameters

in	<i>_pos</i>	A position
in	<i>_rot</i>	A rotation

10.118.2.3 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.

10.118.2.4 gazebo::math::Pose::Pose (const Pose & *_pose*)

Copy constructor.

Parameters

in	<i>_pose</i>	Pose (p. 648) to copy
----	--------------	------------------------------

10.118.2.5 virtual gazebo::math::Pose::~~Pose () [virtual]

Destructor.

10.118.3 Member Function Documentation

10.118.3.1 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
----	-----------	----------------

10.118.3.2 Vector3 gazebo::math::Pose::CoordPositionAdd (const Vector3 & *_pos*) const

Add one point to a vector: $\text{result} = \text{this} + \text{pos}$.

Parameters

in	<code>_pos</code>	Position to add to this pose
----	-------------------	------------------------------

Returns

the resulting position

10.118.3.3 Vector3 gazebo::math::Pose::CoordPositionAdd (const Pose & `_pose`) const

Add one point to another: result = this + pose.

Parameters

in	<code>_pose</code>	The Pose (p. 648) to add
----	--------------------	---------------------------------

Returns

The resulting position

10.118.3.4 Vector3 gazebo::math::Pose::CoordPositionSub (const Pose & `_pose`) const [inline]

Subtract one position from another: result = this - pose.

Parameters

in	<code>_pose</code>	Pose (p. 648) 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.

Referenced by operator-().

10.118.3.5 Quaternion gazebo::math::Pose::CoordRotationAdd (const Quaternion & `_rot`) const

Add one rotation to another: result = this->rot + rot.

Parameters

in	<code>_rot</code>	Rotation to add
----	-------------------	-----------------

Returns

The resulting rotation

10.118.3.6 **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(), gazebo::math::Quaternion::Normalize(), and rot.

Referenced by operator-().

10.118.3.7 **void** gazebo::math::Pose::Correct () `[inline]`

Fix any nan values.

References gazebo::math::Vector3::Correct(), gazebo::math::Quaternion::Correct(), pos, and rot.

10.118.3.8 **Pose** gazebo::math::Pose::GetInverse () const

Get the inverse of this pose.

Returns

the inverse pose

10.118.3.9 **bool** gazebo::math::Pose::IsFinite () const

See if a pose is finite (e.g., not nan)

10.118.3.10 **bool** gazebo::math::Pose::operator!=(const Pose & *_pose*) const

Inequality operator.

Parameters

<i>in</i>	<i>_pose</i>	Pose (p. 648) for comparison
-----------	--------------	-------------------------------------

Returns

True if not equal

10.118.3.11 **Pose** gazebo::math::Pose::operator* (const Pose & *_pose*)

Multiplication operator.

Parameters

in	<code>_pose</code>	the other pose
----	--------------------	----------------

Returns

itself

10.118.3.12 `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	<code>_pose</code>	Pose (p. 648) to add to this pose
----	--------------------	--

Returns

The resulting pose

10.118.3.13 `const Pose& gazebo::math::Pose::operator+= (const Pose & _pose)`

Add-Equals operator.

Parameters

in	<code>_pose</code>	Pose (p. 648) to add to this pose
----	--------------------	--

Returns

The resulting pose

10.118.3.14 `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

References Pose().

10.118.3.15 `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	<code>_pose</code>	Pose (p. 648) to subtract from this one
----	--------------------	--

Returns

The resulting pose

References CoordPositionSub(), CoordRotationSub(), Pose(), and rot.

10.118.3.16 `const Pose& gazebo::math::Pose::operator-= (const Pose & _pose)`

Subtraction operator.

Parameters

<code>in</code>	<code>_pose</code>	Pose (p. 648) to subtract from this one
-----------------	--------------------	--

Returns

The resulting pose

10.118.3.17 `Pose& gazebo::math::Pose::operator= (const Pose & _pose)`

Equal operator.

Parameters

<code>in</code>	<code>_pose</code>	Pose (p. 648) to copy
-----------------	--------------------	------------------------------

10.118.3.18 `bool gazebo::math::Pose::operator== (const Pose & _pose) const`

Equality operator.

Parameters

<code>in</code>	<code>_pose</code>	Pose (p. 648) for comparison
-----------------	--------------------	-------------------------------------

Returns

True if equal

10.118.3.19 `void gazebo::math::Pose::Reset ()`

Reset the pose.

10.118.3.20 `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

10.118.3.21 `void gazebo::math::Pose::Round (int _precision)`

Round all values to `_precision` decimal places.

Parameters

<code>in</code>	<code><i>_precision</i></code>	
-----------------	--------------------------------	--

10.118.3.22 `void gazebo::math::Pose::Set (const Vector3 & _pos, const Quaternion & _rot)`

Set the pose from a **Vector3** (p. 1004) and a **Quaternion** (p. 675).

Parameters

<code>in</code>	<code><i>_pos</i></code>	The position.
<code>in</code>	<code><i>_rot</i></code>	The rotation.

10.118.3.23 `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.

10.118.3.24 `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.

10.118.4 Friends And Related Function Documentation

10.118.4.1 `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

10.118.4.2 `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

10.118.5 Member Data Documentation

10.118.5.1 `Vector3 gazebo::math::Pose::pos`

The position.

Referenced by `CoordPositionSub()`, `Correct()`, and `gazebo::physics::Inertial::GetCoG()`.

10.118.5.2 `Quaternion gazebo::math::Pose::rot`

The rotation.

Referenced by `CoordPositionSub()`, `CoordRotationSub()`, `Correct()`, and `operator-`.

10.118.5.3 `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:

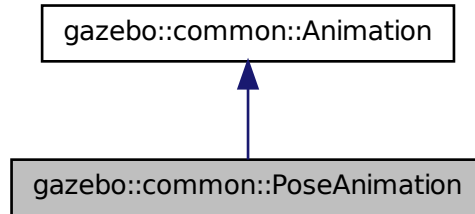
- **Pose.hh**

10.119 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

10.119.1 Detailed Description

A pose animation.

10.119.2 Constructor & Destructor Documentation

10.119.2.1 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

10.119.2.2 virtual gazebo::common::PoseAnimation::~~PoseAnimation () [virtual]

Destructor.

10.119.3 Member Function Documentation

10.119.3.1 void gazebo::common::PoseAnimation::BuildInterpolationSplines () const [protected]

Update the pose splines.

10.119.3.2 PoseKeyFrame* gazebo::common::PoseAnimation::CreateKeyFrame (double *_time*)

Create a pose keyframe at the given time.

Parameters

in	<i>_time</i>	Time (p. 944) at which to create the keyframe
----	--------------	--

Returns

Pointer to the new keyframe

10.119.3.3 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (PoseKeyFrame & *_kf*) const

Get a keyframe using the animation's current time.

Parameters

out	<i>_kf</i>	PoseKeyFrame (p. 660) reference to hold the interpolated result
-----	------------	--

10.119.3.4 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. 944) in seconds
out	<i>_kf</i>	PoseKeyFrame (p. 660) reference to hold the interpolated result

The documentation for this class was generated from the following file:

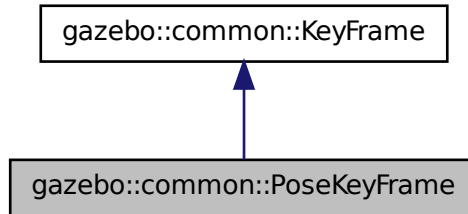
- **Animation.hh**

10.120 gazebo::common::PoseKeyFrame Class Reference

A keyframe for a **PoseAnimation** (p. 657).

```
#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

10.120.1 Detailed Description

A keyframe for a **PoseAnimation** (p. 657).

10.120.2 Constructor & Destructor Documentation

10.120.2.1 gazebo::common::PoseKeyFrame::PoseKeyFrame (double *_time*)

Constructor.

Parameters

in	<i>_time</i>	of the keyframe
----	--------------	-----------------

10.120.2.2 virtual gazebo::common::PoseKeyFrame::~~PoseKeyFrame () [virtual]

Destructor.

10.120.3 Member Function Documentation

10.120.3.1 const math::Quaternion& gazebo::common::PoseKeyFrame::GetRotation () const

Get the rotation of the keyframe.

Returns

The rotation amount

10.120.3.2 const math::Vector3& gazebo::common::PoseKeyFrame::GetTranslation () const

Get the translation of the keyframe.

Returns

The translation amount

10.120.3.3 void gazebo::common::PoseKeyFrame::SetRotation (const math::Quaternion & *_rot*)

Set the rotation for the keyframe.

Parameters

in	<i>_rot</i>	Rotation amount
----	-------------	-----------------

10.120.3.4 void gazebo::common::PoseKeyFrame::SetTranslation (const math::Vector3 & *_trans*)

Set the translation for the keyframe.

Parameters

in	<i>_trans</i>	Translation amount
----	---------------	--------------------

10.120.4 Member Data Documentation

10.120.4.1 `math::Quaternion gazebo::common::PoseKeyFrame::rotate` [protected]

the rotation quaternion

10.120.4.2 `math::Vector3 gazebo::common::PoseKeyFrame::translate` [protected]

the translation vector

The documentation for this class was generated from the following file:

- **KeyFrame.hh**

10.121 gazebo::rendering::Projector Class Reference

Projects a material onto surface, light a light projector.

```
#include <rendering/rendering.hh>
```

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.

10.121.1 Detailed Description

Projects a material onto surface, light a light projector.

10.121.2 Constructor & Destructor Documentation

10.121.2.1 gazebo::rendering::Projector::Projector (VisualPtr *_parent*)

Constructor.

Parameters

in	<i>_parent</i>	Name of the parent visual.
----	----------------	----------------------------

10.121.2.2 virtual gazebo::rendering::Projector::~~Projector () [virtual]

Destructor.

10.121.3 Member Function Documentation

10.121.3.1 VisualPtr gazebo::rendering::Projector::GetParent ()

Get the parent visual.

Returns

Pointer of the parent visual.

10.121.3.2 void gazebo::rendering::Projector::Load (sdf::ElementPtr *_sdf*)

Load from an sdf pointer.

Parameters

in	<i>_sdf</i>	Pointer to the SDF element.
----	-------------	-----------------------------

10.121.3.3 void gazebo::rendering::Projector::Load (const msgs::Projector & *_msg*)

Load from a message.

Parameters

in	<i>_msg</i>	Load from a message.
----	-------------	----------------------

10.121.3.4 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.

10.121.3.5 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.
----	-----------------------	-------------------------------

10.121.3.6 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.
----	---------------------------	---------------------------------

10.121.3.7 void gazebo::rendering::Projector::Toggle ()

Toggle the activation of the projector.

The documentation for this class was generated from the following file:

- **Projector.hh**

10.122 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.*

 - 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)*

 - void **Publish** (**MessagePtr** _msg, boost::function< void(uint32_t)> _cb, uint32_t _id)
- Publish data to remote subscribers.*

 - 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.*

10.122.1 Detailed Description

A publication for a topic.

This facilitates transport of messages

10.122.2 Constructor & Destructor Documentation

10.122.2.1 gazebo::transport::Publication::Publication (const std::string & _topic, const std::string & _msgType)

Constructor.

Parameters

in	<code>_topic</code>	The topic we're publishing
in	<code>_msgType</code>	The type of the topic we're publishing

10.122.2.2 `virtual gazebo::transport::Publication::~~Publication () [virtual]`

Destructor.

10.122.3 Member Function Documentation

10.122.3.1 `void gazebo::transport::Publication::AddPublisher (PublisherPtr _pub)`

Add a publisher.

Parameters

in, out	_pub	Pointer to publisher object to be added
---------	------	---

10.122.3.2 `void gazebo::transport::Publication::AddSubscription (const CallbackHelperPtr _callback)`

Subscribe a callback to our topic.

Parameters

in	_callback	The callback
----	-----------	--------------

10.122.3.3 `void gazebo::transport::Publication::AddSubscription (const NodePtr & _node)`

Subscribe a node to our topic.

Parameters

in	_node	The node
----	-------	----------

10.122.3.4 `void gazebo::transport::Publication::AddTransport (const PublicationTransportPtr & _publink)`

Add a transport.

Parameters

in	_publink	Pointer to publication transport object to be added
----	----------	---

10.122.3.5 `unsigned int gazebo::transport::Publication::GetCallbackCount () const`

Get the number of callbacks.

Returns

The number of callbacks

10.122.3.6 `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

10.122.3.7 `std::string gazebo::transport::Publication::GetMsgType () const`

Get the type of message.

Returns

The type of message

10.122.3.8 `unsigned int gazebo::transport::Publication::GetNodeCount () const`

Get the number of nodes.

Returns

The number of nodes

10.122.3.9 `unsigned int gazebo::transport::Publication::GetRemoteSubscriptionCount ()`

Get the number of remote subscriptions.

Returns

The number of remote subscriptions

10.122.3.10 `unsigned int gazebo::transport::Publication::GetTransportCount () const`

Get the number of transports.

Returns

The number of transports

10.122.3.11 `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

10.122.3.12 `void gazebo::transport::Publication::LocalPublish (const std::string & _data)`

Publish data to local subscribers (skip serialization)

Parameters

in	<i>_data</i>	The data to be published
----	--------------	--------------------------

10.122.3.13 `void gazebo::transport::Publication::Publish (MessagePtr _msg, boost::function< void(uint32_t)> _cb, uint32_t _id)`

Publish data to remote subscribers.

Parameters

in	<i>_msg</i>	Message to be published
in	<i>_cb</i>	Callback to be invoked after publishing is completed

10.122.3.14 `void gazebo::transport::Publication::RemoveSubscription (const NodePtr & _node)`

Unsubscribe a node from our topic.

Parameters

in	<i>_node</i>	The node
----	--------------	----------

10.122.3.15 `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

10.122.3.16 `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

10.122.3.17 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
----	---------------	--

The documentation for this class was generated from the following file:

- **Publication.hh**

10.123 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.

10.123.1 Detailed Description

transport/transport.hh

Reads data from a remote advertiser, and passes the data along to local subscribers

10.123.2 Constructor & Destructor Documentation

10.123.2.1 gazebo::transport::PublicationTransport::PublicationTransport (const std::string & *_topic*, const std::string & *_msgType*)

Constructor.

Parameters

in	<code>_topic</code>	Topic that we're publishing
in	<code>_topic</code>	Type of the topic that we're publishing

10.123.2.2 `virtual gazebo::transport::PublicationTransport::~~PublicationTransport () [virtual]`

Destructor.

10.123.3 Member Function Documentation

10.123.3.1 `void gazebo::transport::PublicationTransport::AddCallback (const boost::function< void(const std::string &);> & _cb)`

Add a callback to the transport.

Parameters

in	<code>_cb</code>	The callback to be added
----	------------------	--------------------------

10.123.3.2 `void gazebo::transport::PublicationTransport::Fini ()`

Finalize the transport.

10.123.3.3 `const ConnectionPtr gazebo::transport::PublicationTransport::GetConnection () const`

Get the underlying connection.

Returns

Pointer to the underlying connection

10.123.3.4 `std::string gazebo::transport::PublicationTransport::GetMsgType () const`

Get the topic type.

Returns

The topic type

10.123.3.5 `std::string gazebo::transport::PublicationTransport::GetTopic () const`

Get the topic name.

Returns

The topic name

10.123.3.6 void gazebo::transport::PublicationTransport::Init (const ConnectionPtr & _conn, bool _latched)

Initialize the transport.

Parameters

in	_conn	The underlying connection.
in	_latched	True to grab the last message sent on the topic.

The documentation for this class was generated from the following file:

- **PublicationTransport.hh**

10.124 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.
- 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.

10.124.1 Detailed Description

A publisher of messages on a topic.

10.124.2 Constructor & Destructor Documentation

10.124.2.1 `gazebo::transport::Publisher::Publisher (const std::string & _topic, const std::string & _msgType, unsigned int _limit, double _hzRate)`

Constructor.

Parameters

in	<code>_topic</code>	Name of topic to be published
in	<code>_msgType</code>	Type of the message to be published
in	<code>_limit</code>	Maximum number of outgoing messages to queue
in	<code>_hz</code>	Update rate for the publisher. Units are 1.0/seconds.

10.124.2.2 `virtual gazebo::transport::Publisher::~~Publisher ()` [virtual]

Destructor.

10.124.3 Member Function Documentation

10.124.3.1 `std::string gazebo::transport::Publisher::GetMsgType ()` const

Get the message type.

Returns

The message type

10.124.3.2 `unsigned int gazebo::transport::Publisher::GetOutgoingCount ()` const

Get the number of outgoing messages.

Returns

The number of outgoing messages

10.124.3.3 `std::string gazebo::transport::Publisher::GetPrevMsg ()` const

Get the previously published message.

Returns

The previously published message, if any

10.124.3.4 MessagePtr gazebo::transport::Publisher::GetPrevMsgPtr () const

Get the previously published message.

Returns

The previously published message, if any

10.124.3.5 std::string gazebo::transport::Publisher::GetTopic () const

Get the topic name.

Returns

The topic name

10.124.3.6 bool gazebo::transport::Publisher::HasConnections () const

Are there any connections?

Returns

true if there are any connections, false otherwise

**10.124.3.7 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

**10.124.3.8 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

10.124.3.9 void gazebo::transport::Publisher::SendMessage ()

Send latest message over the wire. For internal use only.

10.124.3.10 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.
----	--------------	---

10.124.3.11 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
----	---------------------	---

10.124.3.12 void gazebo::transport::Publisher::WaitForConnection () const

Block until a connection has been established with this publisher.

10.124.3.13 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**

10.125 QuadNode Class Reference

```
#include <physics/physics.hh>
```

The documentation for this class was generated from the following file:

- **MapShape.hh**

10.126 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. 1004) 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 **SetToIdentity** ()
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 _shortestPath=false)
Spherical linear interpolation between 2 quaternions, given the ends and an interpolation parameter between 0 and 1.
- static **Quaternion Squad** (double _fT, 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.

10.126.1 Detailed Description

A quaternion class.

10.126.2 Constructor & Destructor Documentation

10.126.2.1 gazebo::math::Quaternion::Quaternion ()

Default Constructor.

Referenced by operator*().

10.126.2.2 gazebo::math::Quaternion::Quaternion (const double & *_w*, const double & *_x*, const double & *_y*, const double & *_z*)

Constructor.

Parameters

in	<i>_w</i>	W param
in	<i>_x</i>	X param
in	<i>_y</i>	Y param
in	<i>_z</i>	Z param

10.126.2.3 gazebo::math::Quaternion::Quaternion (const double & *_roll*, const double & *_pitch*, const double & *_yaw*)

Constructor from Euler angles in radians.

Parameters

in	<i>_roll</i>	roll
in	<i>_pitch</i>	pitch
in	<i>_yaw</i>	yaw

10.126.2.4 gazebo::math::Quaternion::Quaternion (const Vector3 & *_axis*, const double & *_angle*)

Constructor from axis angle.

Parameters

in	<i>_axis</i>	the rotation axis
in	<i>_angle</i>	the rotation angle in radians

10.126.2.5 gazebo::math::Quaternion::Quaternion (const Vector3 & *_rpy*)

Constructor.

Parameters

in	<i>_rpy</i>	euler angles
----	-------------	--------------

10.126.2.6 gazebo::math::Quaternion::Quaternion (const Quaternion & *_qt*)

Copy constructor.

Parameters

<i>qt</i>	Quaternion (p. 675) to copy
-----------	------------------------------------

10.126.2.7 gazebo::math::Quaternion::~~Quaternion ()

Destructor.

10.126.3 Member Function Documentation

10.126.3.1 void gazebo::math::Quaternion::Correct () [inline]

Correct any nan.

References gazebo::math::equal(), w, x, y, and z.

Referenced by gazebo::math::Pose::Correct().

10.126.3.2 double gazebo::math::Quaternion::Dot (const Quaternion & *_q*) const

Dot product.

Parameters

<i>in</i>	<i>_q</i>	the other quaternion
-----------	-----------	----------------------

Returns

the product

10.126.3.3 static Quaternion gazebo::math::Quaternion::EulerToQuaternion (const Vector3 & *_vec*) [static]

Convert euler angles to quaternion.

Parameters

<i>in</i>		
-----------	--	--

10.126.3.4 static Quaternion gazebo::math::Quaternion::EulerToQuaternion (double *_x*, double *_y*, double *_z*) [static]

Convert euler angles to quaternion.

Parameters

<i>in</i>	<i>_x</i>	rotation along x
<i>in</i>	<i>_y</i>	rotation along y
<i>in</i>	<i>_z</i>	rotation along z

10.126.3.5 `void gazebo::math::Quaternion::GetAsAxis (Vector3 & _axis, double & _angle) const`

Return rotation as axis and angle.

Parameters

<code>in</code>	<code><i>_axis</i></code>	rotation axis
<code>in</code>	<code><i>_angle</i></code>	ccw angle in radians

10.126.3.6 `Vector3 gazebo::math::Quaternion::GetAsEuler () const`

Return the rotation in Euler angles.

Returns

This quaternion as an Euler vector

10.126.3.7 `Matrix3 gazebo::math::Quaternion::GetAsMatrix3 () const`

Get the quaternion as a 3x3 matrix.

10.126.3.8 `Matrix4 gazebo::math::Quaternion::GetAsMatrix4 () const`

Get the quaternion as a 4x4 matrix.

Returns

a 4x4 matrix

10.126.3.9 `Quaternion gazebo::math::Quaternion::GetExp () const`

Return the exponent.

Returns

the exp

10.126.3.10 `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()`, `gazebo::math::Pose::CoordRotationSub()`, and `RotateVector()`.

10.126.3.11 `Quaternion gazebo::math::Quaternion::GetLog () const`

Return the logarithm.

Returns

the log

10.126.3.12 `double gazebo::math::Quaternion::GetPitch ()`

Get the Euler pitch angle in radians.

Returns

the pitch

10.126.3.13 `double gazebo::math::Quaternion::GetRoll ()`

Get the Euler roll angle in radians.

Returns

the roll

10.126.3.14 `Vector3 gazebo::math::Quaternion::GetXAxis () const`

Return the X axis.

Returns

the vector

10.126.3.15 `double gazebo::math::Quaternion::GetYaw ()`

Get the Euler yaw angle in radians.

Returns

the yaw

10.126.3.16 `Vector3 gazebo::math::Quaternion::GetYAxis () const`

Return the Y axis.

Returns

the vector

10.126.3.17 `Vector3 gazebo::math::Quaternion::GetZAxis () const`

Return the Z axis.

Returns

the vector

10.126.3.18 `void gazebo::math::Quaternion::Invert ()`

Invert the quaternion.

10.126.3.19 `bool gazebo::math::Quaternion::IsFinite () const`

See if a quatern is finite (e.g., not nan)

Returns

True if quatern is finite

10.126.3.20 `void gazebo::math::Quaternion::Normalize ()`

Normalize the quaternion.

Referenced by `gazebo::math::Pose::CoordRotationSub()`.

10.126.3.21 `bool gazebo::math::Quaternion::operator!=(const Quaternion & _qt) const`

Not equal to operator.

Parameters

<code>in</code>	<code>_qt</code>	Quaternion (p. 675) for comparison
-----------------	------------------	---

Returns

True if not equal

10.126.3.22 `Quaternion gazebo::math::Quaternion::operator*(const Quaternion & _q) const` `[inline]`

Multiplication operator.

Parameters

<code>in</code>	<code>_qt</code>	Quaternion (p. 675) for multiplication
-----------------	------------------	---

Returns

This quaternion multiplied by the parameter

References Quaternion(), w, x, y, and z.

10.126.3.23 Quaternion gazebo::math::Quaternion::operator* (const double & _f) const

Multiplication operator.

Parameters

<code>in</code>	<code>_f</code>	factor
-----------------	-----------------	--------

Returns

quaternion multiplied by `_f`

10.126.3.24 Vector3 gazebo::math::Quaternion::operator* (const Vector3 & _v) const

Vector3 (p. 1004) multiplication operator.

Parameters

<code>in</code>	<code>_v</code>	vector to multiply
-----------------	-----------------	--------------------

10.126.3.25 Quaternion gazebo::math::Quaternion::operator*= (const Quaternion & qt)

Multiplication operator.

Parameters

<code>in</code>	<code>_qt</code>	Quaternion (p. 675) for multiplication
-----------------	------------------	---

Returns

This quatern multiplied by the parameter

10.126.3.26 Quaternion gazebo::math::Quaternion::operator+ (const Quaternion & qt) const

Addition operator.

Parameters

<code>in</code>	<code>_qt</code>	quaternion for addition
-----------------	------------------	-------------------------

Returns

this quaternion + `_qt`

10.126.3.27 Quaternion gazebo::math::Quaternion::operator+= (const Quaternion & _qt)

Addition operator.

Parameters

in	_qt	quaternion for addition
----	-----	-------------------------

Returns

this quaternion + qt

10.126.3.28 Quaternion gazebo::math::Quaternion::operator- (const Quaternion & _qt) const

Substraction operator.

Parameters

in	_qt	quaternion to subtract
----	-----	------------------------

Returns

this quaternion - _qt

10.126.3.29 Quaternion gazebo::math::Quaternion::operator- () const

Unary minus operator.

Returns

negates each component of the quaternion

10.126.3.30 Quaternion gazebo::math::Quaternion::operator-= (const Quaternion & _qt)

Substraction operator.

Parameters

in	_qt	Quaternion (p. 675) for subtraction
----	-----	--

Returns

This quatern - qt

10.126.3.31 Quaternion& gazebo::math::Quaternion::operator= (const Quaternion & _qt)

Equal operator.

Parameters

in	<i>_qt</i>	Quaternion (p. 675) to copy
----	------------	------------------------------------

10.126.3.32 `bool gazebo::math::Quaternion::operator==(const Quaternion & _qt) const`

Equal to operator.

Parameters

in	<i>_qt</i>	Quaternion (p. 675) for comparison
----	------------	---

Returns

True if equal

10.126.3.33 `Vector3 gazebo::math::Quaternion::RotateVector (const Vector3 & _vec) const` `[inline]`

Rotate a vector using the quaternion.

Parameters

in	<i>_vec</i>	vector to rotate
----	-------------	------------------

Returns

the rotated vector

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

10.126.3.34 `Vector3 gazebo::math::Quaternion::RotateVectorReverse (Vector3 _vec) const`

Do the reverse rotation of a vector by this quaternion.

Parameters

in	<i>_vec</i>	the vector
----	-------------	------------

Returns

the

10.126.3.35 `void gazebo::math::Quaternion::Round (int _precision)`

Round all values to `_precision` decimal places.

Parameters

in	<i>_precision</i>	the precision
----	-------------------	---------------

10.126.3.36 void gazebo::math::Quaternion::Scale (double *_scale*)

Scale a Quaternionion.

Parameters

in	<i>_scale</i>	Amount to scale this rotation
----	---------------	-------------------------------

10.126.3.37 void gazebo::math::Quaternion::Set (double *_u*, double *_x*, double *_y*, double *_z*)

Set this quaternion from 4 floating numbers.

Parameters

in	<i>_u</i>	<i>u</i>
in	<i>_x</i>	<i>x</i>
in	<i>_y</i>	<i>y</i>
in	<i>_z</i>	<i>z</i>

10.126.3.38 void gazebo::math::Quaternion::SetFromAxis (double *_x*, double *_y*, double *_z*, double *_a*)

Set the quaternion from an axis and angle.

Parameters

in	<i>_x</i>	X axis
in	<i>_y</i>	Y axis
in	<i>_z</i>	Z axis
in	<i>_a</i>	Angle (p. 131) in radians

10.126.3.39 void gazebo::math::Quaternion::SetFromAxis (const Vector3 & *_axis*, double *_a*)

Set the quaternion from an axis and angle.

Parameters

in	<i>_axis</i>	Axis
in	<i>_a</i>	Angle (p. 131) in radians

10.126.3.40 void gazebo::math::Quaternion::SetFromEuler (const Vector3 & *_vec*)

Set the quaternion from Euler angles.

The order of operations are roll, pitch, yaw.

Parameters

in	<i>vec</i>	Euler angle
----	------------	-------------

10.126.3.41 `void gazebo::math::Quaternion::SetFromEuler (double _roll, double _pitch, double _yaw)`

Set the quaternion from Euler angles.

Parameters

<code>in</code>	<code>_roll</code>	Roll angle (radians).
<code>in</code>	<code>_pitch</code>	Roll angle (radians).
<code>in</code>	<code>_yaw</code>	Roll angle (radians).

10.126.3.42 `void gazebo::math::Quaternion::SetToIdentity ()`

Set the quatern to the identity.

10.126.3.43 `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

<code>in</code>	<code>_ft</code>	the interpolation parameter
<code>in</code>	<code>_rkP</code>	the beginning quaternion
<code>in</code>	<code>_rkQ</code>	the end quaternion
<code>in</code>	<code>_shortestPath</code>	when true, the rotation may be inverted to get to minimize rotation

10.126.3.44 `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

<code>in</code>	<code>_ft</code>	the interpolation parameter
<code>in</code>	<code>_rkP</code>	the beginning quaternion
<code>in</code>	<code>_rkA</code>	first intermediate quaternion
<code>in</code>	<code>_rkB</code>	second intermediate quaternion
<code>in</code>	<code>_rkQ</code>	the end quaternion
<code>in</code>	<code>_shortestPath</code>	when true, the rotation may be inverted to get to minimize rotation

10.126.4 Friends And Related Function Documentation

10.126.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Quaternion & _q) [friend]`

Stream insertion operator.

Parameters

<code>in</code>	<code>_out</code>	output stream
<code>in</code>	<code>_q</code>	quaternion to output

Returns

the stream

10.126.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Quaternion & _q)` [*friend*]

Stream extraction operator.

Parameters

<code>in</code>	<code>_in</code>	input stream
<code>in</code>	<code>_q</code>	Quaternion (p. 675) to read values into

Returns

The istream

10.126.5 Member Data Documentation

10.126.5.1 `double gazebo::math::Quaternion::w`

Attributes of the quaternion.

Referenced by `Correct()`, `GetInverse()`, and `operator*()`.

10.126.5.2 `double gazebo::math::Quaternion::x`

Attributes of the quaternion.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `GetInverse()`, `operator*()`, and `RotateVector()`.

10.126.5.3 `double gazebo::math::Quaternion::y`

Attributes of the quaternion.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `GetInverse()`, `operator*()`, and `RotateVector()`.

10.126.5.4 `double gazebo::math::Quaternion::z`

Attributes of the quaternion.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `GetInverse()`, `operator*()`, and `RotateVector()`.

The documentation for this class was generated from the following file:

- **Quaternion.hh**

10.127 gazebo::math::Rand Class Reference

Random number generator class.

```
#include <gzmath/gzmath.hh>
```


Static Public Member Functions

- static double **GetDbiNormal** (double _mean=0, double _sigma=1)
Get a double from a normal distribution.
- static double **GetDbiUniform** (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.

10.127.1 Detailed Description

Random number generator class.

10.127.2 Member Function Documentation

10.127.2.1 static double gazebo::math::Rand::GetDbiNormal (double *_mean* = 0, double *_sigma* = 1) [static]

Get a double from a normal distribution.

Parameters

in	<i>_mean</i>	Mean value for the distribution
in	<i>_sigma</i>	Sigma value for the distribution

10.127.2.2 static double gazebo::math::Rand::GetDbiUniform (double *_min* = 0, double *_max* = 1) [static]

Get a double from a uniform distribution.

Parameters

in	<i>_min</i>	Minimum bound for the random number
in	<i>_max</i>	Maximum bound for the random number

10.127.2.3 static int gazebo::math::Rand::GetIntNormal (int *_mean*, int *_sigma*) [static]

Get a double from a normal distribution.

Parameters

in	<i>_mean</i>	Mean value for the distribution
in	<i>_sigma</i>	Sigma value for the distribution

10.127.2.4 `static int gazebo::math::Rand::GetIntUniform (int _min, int _max) [static]`

Get a integer from a uniform distribution.

Parameters

<code>in</code>	<code><i>_min</i></code>	Minimum bound for the random number
<code>in</code>	<code><i>_max</i></code>	Maximum bound for the random number

10.127.2.5 `static uint32_t gazebo::math::Rand::GetSeed () [static]`

Get the seed value.

Returns

The seed value used to initialize the random number generator.

10.127.2.6 `static void gazebo::math::Rand::SetSeed (uint32_t _seed) [static]`

Set the seed value.

Parameters

<code>in</code>	<code><i>_seed</i></code>	The seed used to initialize the random number generator.
-----------------	---------------------------	--

The documentation for this class was generated from the following file:

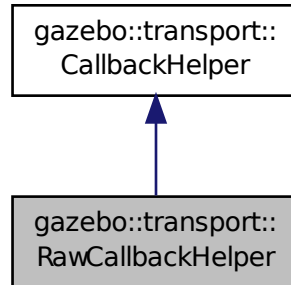
- **Rand.hh**

10.128 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

10.128.1 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.

10.128.2 Constructor & Destructor Documentation

10.128.2.1 gazebo::transport::RawCallbackHelper::RawCallbackHelper (const boost::function< void(const std::string &);> &_cb, bool _latching = false) [inline]

Constructor.

Parameters

in	_cb	boost function to call on incoming messages
in	_latching	Set to true to make the callback helper latching.

10.128.3 Member Function Documentation

10.128.3.1 `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. 175).

10.128.3.2 `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	<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. 175).

10.128.3.3 `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. 176).

10.128.3.4 `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. 176).

The documentation for this class was generated from the following file:

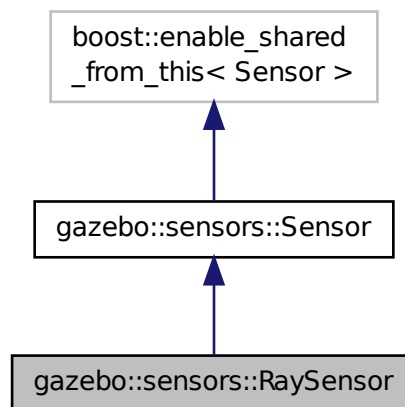
- **CallbackHelper.hh**

10.129 gazebo::sensors::RaySensor Class Reference

Sensor (p. 751) 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** (int _index)

Get detected fiducial value for a ray.

- **physics::MultiRayShapePtr GetLaserShape ()** const
*Returns a pointer to the internal **physics::MultiRayShape** (p. 578).*
- double **GetRange** (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** (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.
- 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 void **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.129.1 Detailed Description

Sensor (p. 751) 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).

10.129.2 Constructor & Destructor Documentation

10.129.2.1 gazebo::sensors::RaySensor::RaySensor ()

Constructor.

10.129.2.2 virtual gazebo::sensors::RaySensor::~~RaySensor () [virtual]

Destructor.

10.129.3 Member Function Documentation

10.129.3.1 virtual void gazebo::sensors::RaySensor::Fini () [protected],[virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 755).

10.129.3.2 math::Angle gazebo::sensors::RaySensor::GetAngleMax () const

Get the maximum angle.

Returns

the maximum angle object

10.129.3.3 math::Angle gazebo::sensors::RaySensor::GetAngleMin () const

Get the minimum angle.

Returns

The minimum angle object

10.129.3.4 double gazebo::sensors::RaySensor::GetAngleResolution () const

Get the angle in radians between each range.

Returns

Resolution of the angle

10.129.3.5 `int gazebo::sensors::RaySensor::GetFiducial (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	_index	Index value of specific ray
----	--------	-----------------------------

Returns

Fiducial value

10.129.3.6 `physics::MultiRayShapePtr gazebo::sensors::RaySensor::GetLaserShape () const [inline]`

Returns a pointer to the internal `physics::MultiRayShape` (p. 578).

Returns

Pointer to ray shape

10.129.3.7 `double gazebo::sensors::RaySensor::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

in	_index	Index of specific ray
----	--------	-----------------------

Returns

Returns `DBL_MAX` for no detection.

10.129.3.8 `int gazebo::sensors::RaySensor::GetRangeCount () const`

Get the range count.

Returns

The number of ranges

10.129.3.9 `double gazebo::sensors::RaySensor::GetRangeMax () const`

Get the maximum range.

Returns

The maximum range

10.129.3.10 `double gazebo::sensors::RaySensor::GetRangeMin () const`

Get the minimum range.

Returns

The minimum range

10.129.3.11 `double gazebo::sensors::RaySensor::GetRangeResolution () const`

Get the range resolution.

Returns

Resolution of the range

10.129.3.12 `void gazebo::sensors::RaySensor::GetRanges (std::vector< double > & _ranges)`

Get all the ranges.

Parameters

<code>_ranges</code>	A vector that will contain all the range data
----------------------	---

10.129.3.13 `int gazebo::sensors::RaySensor::GetRayCount () const`

Get the ray count.

Returns

The number of rays

10.129.3.14 `double gazebo::sensors::RaySensor::GetRetro (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

<code>in</code>	<code>_index</code>	Index of specific ray
-----------------	---------------------	-----------------------

Returns

Retro (intensity) value for ray

10.129.3.15 `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. 757).

10.129.3.16 `math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMax () const`

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.129.3.17 `math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMin () const`

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.129.3.18 `int gazebo::sensors::RaySensor::GetVerticalRangeCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.129.3.19 `int gazebo::sensors::RaySensor::GetVerticalRayCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.129.3.20 `virtual void gazebo::sensors::RaySensor::Init () [virtual]`

Initialize the sensor.

Reimplemented from **`gazebo::sensors::Sensor`** (p. 758).

10.129.3.21 `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. 758).

10.129.3.22 `virtual void gazebo::sensors::RaySensor::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. 759).

10.129.3.23 `virtual void 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
----	---------------------	--

Reimplemented from **`gazebo::sensors::Sensor`** (p. 760).

The documentation for this class was generated from the following file:

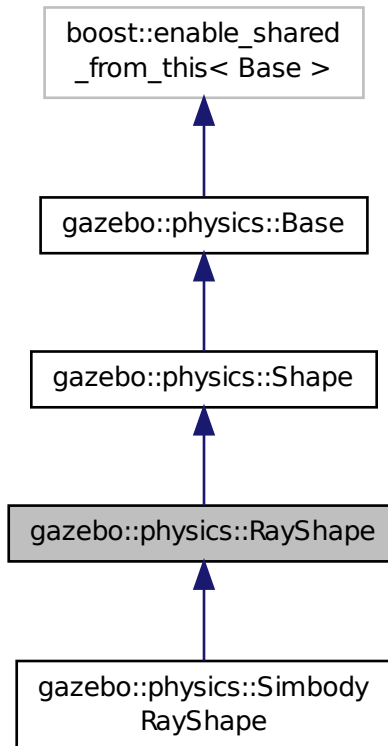
- **`RaySensor.hh`**

10.130 gazebo::physics::RayShape Class Reference

Base (p. 153) 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 **~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

10.130.1 Detailed Description

Base (p. 153) class for Ray collision geometry.

10.130.2 Constructor & Destructor Documentation

10.130.2.1 `gazebo::physics::RayShape::RayShape (PhysicsEnginePtr _physicsEngine) [explicit]`

Constructor for a global ray.

Parameters

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

10.130.2.2 `gazebo::physics::RayShape::RayShape (CollisionPtr _parent) [explicit]`

Constructor.

Parameters

in	<code><i>_parent</i></code>	Collision (p. 213) parent of the shape.
----	-----------------------------	--

10.130.2.3 `virtual gazebo::physics::RayShape::~~RayShape () [virtual]`

Destructor.

10.130.3 Member Function Documentation

10.130.3.1 `void gazebo::physics::RayShape::FillMsg (msgs::Geometry & _msg) [virtual]`

Fill a message with data from this object.

Parameters

out	<code><i>_msg</i></code>	Message to fill. Implement this function.
-----	--------------------------	---

Implements **gazebo::physics::Shape** (p. 777).

10.130.3.2 `int gazebo::physics::RayShape::GetFiducial () const`

Get the fiducial id detected by this ray.

Returns

Fiducial id detected.

10.130.3.3 virtual void gazebo::physics::RayShape::GetGlobalPoints (math::Vector3 & *_posA*, math::Vector3 & *_posB*)
[virtual]

Get the global starting and ending points.

Parameters

out	<i>_posA</i>	Returns the starting point.
out	<i>_posB</i>	Returns the ending point.

10.130.3.4 virtual void gazebo::physics::RayShape::GetIntersection (double & *_dist*, std::string & *_entity*) [pure 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.

Implemented in **gazebo::physics::SimbodyRayShape** (p. 841).

10.130.3.5 double gazebo::physics::RayShape::GetLength () const

Get the length of the ray.

Returns

The ray length.

10.130.3.6 virtual void gazebo::physics::RayShape::GetRelativePoints (math::Vector3 & *_posA*, math::Vector3 & *_posB*)
[virtual]

Get the relative starting and ending points.

Parameters

in	<i>_posA</i>	Returns the starting point.
in	<i>_posB</i>	Returns the ending point.

10.130.3.7 float gazebo::physics::RayShape::GetRetro () const

Get the retro-reflectiveness detected by this ray.

Returns

Retro reflectance value.

10.130.3.8 `virtual void gazebo::physics::RayShape::Init () [virtual]`

In the ray.

Implements **`gazebo::physics::Shape`** (p. 777).

10.130.3.9 `virtual void gazebo::physics::RayShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]`

Update this shape from a message.

Parameters

<code>in</code>	<code>_msg</code>	Message to update from. Implement this function.
-----------------	-------------------	--

Implements **`gazebo::physics::Shape`** (p. 778).

10.130.3.10 `void gazebo::physics::RayShape::SetFiducial (int _fid)`

Set the fiducial id detected by this ray.

Parameters

<code>in</code>	<code>_fid</code>	Fiducial id detected by this ray.
-----------------	-------------------	-----------------------------------

10.130.3.11 `virtual void gazebo::physics::RayShape::SetLength (double _len) [virtual]`

Set the length of the ray.

Parameters

<code>in</code>	<code>_len</code>	Length of the array.
-----------------	-------------------	----------------------

10.130.3.12 `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

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

Reimplemented in **`gazebo::physics::SimbodyRayShape`** (p. 841).

10.130.3.13 `void gazebo::physics::RayShape::SetRetro (float _retro)`

Set the retro-reflectivness detected by this ray.

Parameters

in	<code>_retro</code>	Retro reflectance value.
----	---------------------	--------------------------

10.130.3.14 `virtual void gazebo::physics::RayShape::SetScale (const math::Vector3 & _scale) [virtual]`

Set the scale of the ray.

Implements **gazebo::physics::Shape** (p. 778).

10.130.3.15 `virtual void gazebo::physics::RayShape::Update () [pure virtual]`

Update the ray collision.

Reimplemented from **gazebo::physics::Base** (p. 163).

Implemented in **gazebo::physics::SimbodyRayShape** (p. 842).

10.130.4 Member Data Documentation

10.130.4.1 `int gazebo::physics::RayShape::contactFiducial [protected]`

Fiducial ID value.

10.130.4.2 `double gazebo::physics::RayShape::contactLen [protected]`

Length of the ray.

10.130.4.3 `double gazebo::physics::RayShape::contactRetro [protected]`

Retro reflectance value.

10.130.4.4 `math::Vector3 gazebo::physics::RayShape::globalEndPos [protected]`

End position of the ray in global cs.

10.130.4.5 `math::Vector3 gazebo::physics::RayShape::globalStartPos [protected]`

Start position of the ray in global cs.

10.130.4.6 `math::Vector3 gazebo::physics::RayShape::relativeEndPos [protected]`

End position of the ray, relative to the body.

10.130.4.7 `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:

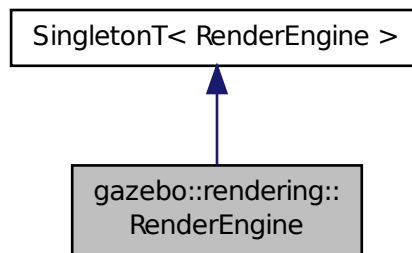
- [RayShape.hh](#)

10.131 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. 123) 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. 123). Load must happen before Init.*
- void **Load ()**
*Load the parameters for **Ogre** (p. 123). 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

10.131.1 Detailed Description

Adaptor to Ogre3d.

Provides the interface to load, initialize the rendering engine.

10.131.2 Member Enumeration Documentation

10.131.2.1 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.

10.131.3 Member Function Documentation

10.131.3.1 `void gazebo::rendering::RenderEngine::AddResourcePath (const std::string & _uri)`

Add a new path for **Ogre** (p. 123) to search for resources.

Parameters

in	_uri	URI of the path. The uri should be of the form <code>file://</code> or <code>model://</code>
----	------	--

10.131.3.2 `ScenePtr gazebo::rendering::RenderEngine::CreateScene (const std::string & _name, bool _enableVisualizations, bool _isServer = false)`

Create a scene.

Parameters

in	_name	The name of the scene.
in	_enable- Visualizations	True enables visualization elements such as laser lines.

10.131.3.3 `void gazebo::rendering::RenderEngine::Fini ()`

Tears down the rendering engine.

10.131.3.4 `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

10.131.3.5 `ScenePtr gazebo::rendering::RenderEngine::GetScene (const std::string & _name = " ")`

Get a scene by name.

Parameters

in	_name	Name of the scene to retrieve.
----	-------	--------------------------------

Returns

A pointer to the **Scene** (p. 728), or NULL if the scene doesn't exist.

10.131.3.6 `ScenePtr gazebo::rendering::RenderEngine::GetScene (unsigned int _index)`

Get a scene by index.

The index should be between 0 and **GetSceneCount()** (p. 709).

Parameters

in	_index	The index of the scene.
----	--------	-------------------------

Returns

A pointer to a **Scene** (p. 728), or NULL if the index was invalid.

10.131.3.7 unsigned int gazebo::rendering::RenderEngine::GetSceneCount () const

Get the number of scenes.

Returns

The number of scenes created by the **RenderEngine** (p. 706).

10.131.3.8 WindowManagerPtr gazebo::rendering::RenderEngine::GetWindowManager () const

Get a pointer to the window manager.

Returns

Pointer to the window manager.

10.131.3.9 void gazebo::rendering::RenderEngine::Init ()

Initialize **Ogre** (p. 123). Load must happen before Init.

10.131.3.10 void gazebo::rendering::RenderEngine::Load ()

Load the parameters for **Ogre** (p. 123). Load must happen before Init.

10.131.3.11 void gazebo::rendering::RenderEngine::RemoveScene (const std::string & _name)

Remove a scene.

Parameters

in	_name	The name of the scene to remove.
----	-------	----------------------------------

10.131.4 Member Data Documentation

10.131.4.1 void* gazebo::rendering::RenderEngine::dummyContext [protected]

GLX context used to render the scenes.Used for gui-less operation.

10.131.4.2 `void* gazebo::rendering::RenderEngine::dummyDisplay` [protected]

Pointer to the dummy display.Used for gui-less operation.

10.131.4.3 `uint64_t gazebo::rendering::RenderEngine::dummyWindowId` [protected]

ID for a dummy window. Used for gui-less operation.

10.131.4.4 `Ogre::Root* gazebo::rendering::RenderEngine::root`

Pointer to the root scene node.

The documentation for this class was generated from the following file:

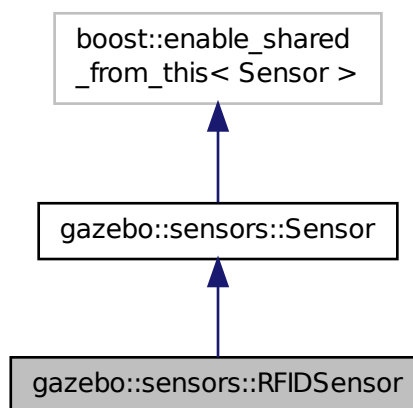
- **RenderEngine.hh**

10.132 gazebo::sensors::RFIDSensor Class Reference

Sensor (p. 751) 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 void **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.132.1 Detailed Description

Sensor (p. 751) class for RFID type of sensor.

10.132.2 Constructor & Destructor Documentation

10.132.2.1 gazebo::sensors::RFIDSensor::RFIDSensor ()

Constructor.

10.132.2.2 virtual gazebo::sensors::RFIDSensor::~~RFIDSensor () [virtual]

Destructor.

10.132.3 Member Function Documentation

10.132.3.1 void gazebo::sensors::RFIDSensor::AddTag (RFIDTag *_tag)

10.132.3.2 virtual void gazebo::sensors::RFIDSensor::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 755).

10.132.3.3 virtual void gazebo::sensors::RFIDSensor::Init () [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 758).

10.132.3.4 `virtual void gazebo::sensors::RFIDSensor::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. 751) parameters.
<code>in</code>	<code>_worldName</code>	Name of world to load from.

Reimplemented from `gazebo::sensors::Sensor` (p. 759).

10.132.3.5 `virtual void gazebo::sensors::RFIDSensor::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. 759).

10.132.3.6 `virtual void 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

<code>in</code>	<code>_force</code>	True if update is forced, false if not
-----------------	---------------------	--

Reimplemented from `gazebo::sensors::Sensor` (p. 760).

The documentation for this class was generated from the following file:

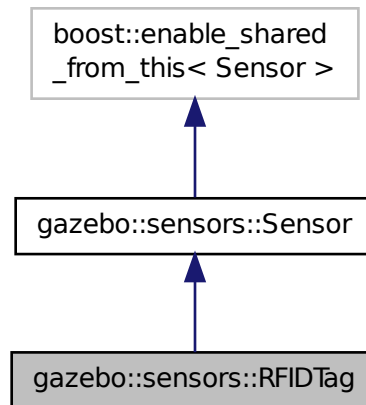
- `RFIDSensor.hh`

10.133 gazebo::sensors::RFIDTag Class Reference

RFIDTag (p. 712) to interact with RFIDTagSensors.

```
#include <sensors/sensors.hh>
```


Inheritance diagram for gazebo::sensors::RFIDTag:



Public Member Functions

- **RFIDTag** ()
Constructor.
- virtual **~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)
- virtual void **Load** (const std::string &_worldName)
Load the sensor with default parameters.

Protected Member Functions

- virtual void **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Additional Inherited Members

10.133.1 Detailed Description

RFIDTag (p. 712) to interact with RFIDTagSensors.

10.133.2 Constructor & Destructor Documentation

10.133.2.1 gazebo::sensors::RFIDTag::RFIDTag ()

Constructor.

10.133.2.2 virtual gazebo::sensors::RFIDTag::~~RFIDTag () [virtual]

Destructor.

10.133.3 Member Function Documentation

10.133.3.1 virtual void gazebo::sensors::RFIDTag::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 755).

10.133.3.2 math::Pose gazebo::sensors::RFIDTag::GetTagPose () const [inline]

Returns pose of tag in world coordinate.

Returns

Pose of object.

10.133.3.3 virtual void gazebo::sensors::RFIDTag::Init () [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 758).

10.133.3.4 virtual void gazebo::sensors::RFIDTag::Load (const std::string & *_worldName*, sdf::ElementPtr & *_sdf*) [virtual]

10.133.3.5 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. 759).

10.133.3.6 virtual void 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

in	<i>_force</i>	True if update is forced, false if not
----	---------------	--

Reimplemented from `gazebo::sensors::Sensor` (p. 760).

The documentation for this class was generated from the following file:

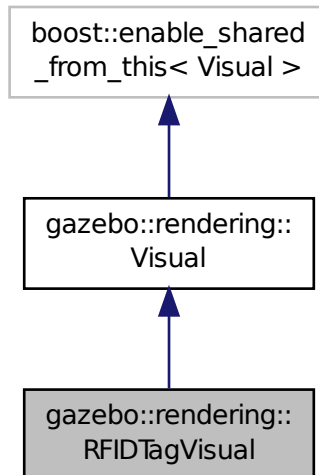
- `RFIDTag.hh`

10.134 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

10.134.1 Detailed Description

Visualization for RFID tags sensor.

10.134.2 Constructor & Destructor Documentation

10.134.2.1 `gazebo::rendering::RFIDTagVisual::RFIDTagVisual (const std::string & _name, VisualIPtr _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>	Parent visual.
<code>in</code>	<code><i>_topicName</i></code>	Name of the topic that publishes RFID data.

See Also

sensors::RFIDSensor (p. 710)

10.134.2.2 `virtual gazebo::rendering::RFIDTagVisual::~RFIDTagVisual () [virtual]`

Destructor.

The documentation for this class was generated from the following file:

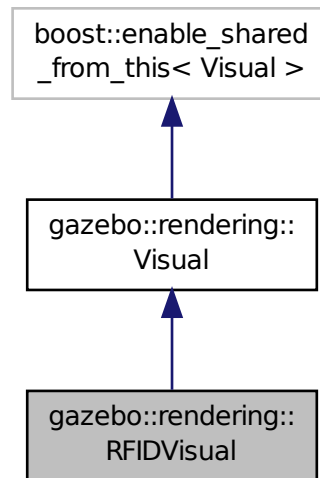
- **RFIDTagVisual.hh**

10.135 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

10.135.1 Detailed Description

Visualization for RFID sensor.

10.135.2 Constructor & Destructor Documentation

10.135.2.1 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. 1034).
in	<i>_vis</i>	Parent Visual (p. 1034).
in	<i>_topicName</i>	Name of the topic which publishes RFID data.

10.135.2.2 virtual gazebo::rendering::RFIDVisual::~~RFIDVisual () [virtual]

Destructor.

The documentation for this class was generated from the following file:

- **RFIDVisual.hh**

10.136 Road Class Reference

Used to render a strip of road.

```
#include <rendering/rendering.hh>
```

10.136.1 Detailed Description

Used to render a strip of road.

The documentation for this class was generated from the following file:

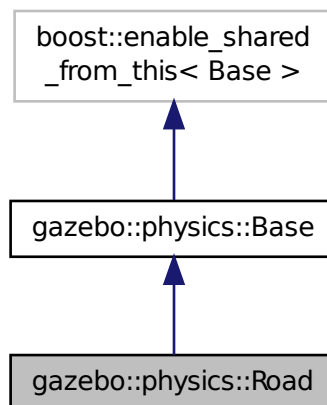
- **Road2d.hh**

10.137 gazebo::physics::Road Class Reference

for building a **Road** (p. 718) 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

10.137.1 Detailed Description

for building a **Road** (p. 718) from SDF

10.137.2 Constructor & Destructor Documentation

10.137.2.1 gazebo::physics::Road::Road (**BasePtr** _parent) [explicit]

Constructor.

Parameters

in	_parent	Parent of this road object.
----	---------	-----------------------------

10.137.2.2 virtual gazebo::physics::Road::~~Road () [virtual]

Destructor.

10.137.3 Member Function Documentation

10.137.3.1 virtual void gazebo::physics::Road::Init () [virtual]

Initialize the road.

Reimplemented from **gazebo::physics::Base** (p. 160).

10.137.3.2 void gazebo::physics::Road::Load (sdf::ElementPtr _sdf) [virtual]

Load the road from SDF.

Parameters

in	_sdf	SDF values to load from.
----	------	--------------------------

Reimplemented from **gazebo::physics::Base** (p. 161).

The documentation for this class was generated from the following file:

- **Road.hh**

10.138 gazebo::rendering::Road2d Class Reference

```
#include <Road2d.hh>
```

Public Member Functions

- **Road2d** ()
Constructor.
- virtual **~Road2d** ()
Destructor.
- void **Load** (**VisualPtr** _parent)
Load the visual using a parent visual.

10.138.1 Constructor & Destructor Documentation

10.138.1.1 gazebo::rendering::Road2d::Road2d ()

Constructor.

10.138.1.2 virtual gazebo::rendering::Road2d::~~Road2d () [virtual]

Destructor.

10.138.2 Member Function Documentation

10.138.2.1 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**

10.139 gazebo::math::RotationSpline Class Reference

Spline (p. 906) 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

10.139.1 Detailed Description

Spline (p. 906) for rotations.

10.139.2 Constructor & Destructor Documentation

10.139.2.1 gazebo::math::RotationSpline::RotationSpline ()

Constructor. Sets the autoCalc to true.

10.139.2.2 gazebo::math::RotationSpline::~~RotationSpline ()

Destructor. Nothing is done.

10.139.3 Member Function Documentation

10.139.3.1 `void gazebo::math::RotationSpline::AddPoint (const Quaternion & _p)`

Adds a control point to the end of the spline.

Parameters

in	_p	control point
----	----	---------------

10.139.3.2 `void gazebo::math::RotationSpline::Clear ()`

Clears all the points in the spline.

10.139.3.3 `unsigned int gazebo::math::RotationSpline::GetNumPoints () const`

Gets the number of control points in the spline.

Returns

the count

10.139.3.4 `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)

10.139.3.5 `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

10.139.3.6 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

10.139.3.7 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.

10.139.3.8 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 recalTangents 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 recalTangents to recalculate them when it best suits.
----	------------------	--

10.139.3.9 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

10.139.4 Member Data Documentation

10.139.4.1 `bool gazebo::math::RotationSpline::autoCalc` [protected]

Automatic recalculation of tangents when control points are updated.

10.139.4.2 `std::vector<Quaternion> gazebo::math::RotationSpline::points` [protected]

the control points

10.139.4.3 `std::vector<Quaternion> gazebo::math::RotationSpline::tangents` [protected]

the tangents

The documentation for this class was generated from the following file:

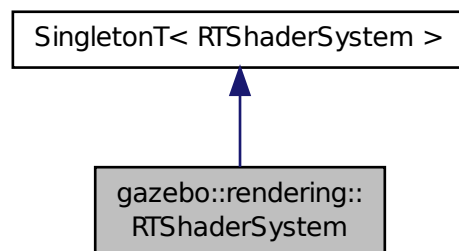
- **RotationSpline.hh**

10.140 gazebo::rendering::RTShaderSystem Class Reference

Implements **Ogre** (p. 123)'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. 123) 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

10.140.1 Detailed Description

Implements **Ogre** (p. 123)'s Run-Time Shader system.

This class allows Gazebo to generate per-pixel shaders for every material at run-time.

10.140.2 Member Enumeration Documentation

10.140.2.1 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.

10.140.3 Member Function Documentation

10.140.3.1 void gazebo::rendering::RTShaderSystem::AddScene (ScenePtr _scene)

Add a scene manager.

Parameters

in	_scene	The scene to process
----	--------	----------------------

10.140.3.2 void gazebo::rendering::RTShaderSystem::ApplyShadows (ScenePtr _scene)

Apply shadows to a scene.

Parameters

in	_scene	The scene to receive shadows.
----	--------	-------------------------------

10.140.3.3 void gazebo::rendering::RTShaderSystem::AttachEntity (Visual * vis)

Set an Ogre::Entity to use RT shaders.

Parameters

in	_vis	Visual (p. 1034) that will use the RTShaderSystem (p. 724).
----	------	---

10.140.3.4 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.

10.140.3.5 void gazebo::rendering::RTShaderSystem::Clear ()

Clear the shader system.

10.140.3.6 void gazebo::rendering::RTShaderSystem::DetachEntity (Visual * _vis)

Remove and entity.

Parameters

in	_vis	Remove this visual.
----	------	---------------------

10.140.3.7 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.

10.140.3.8 void gazebo::rendering::RTShaderSystem::Fini ()

Finalize the shader system.

10.140.3.9 void gazebo::rendering::RTShaderSystem::GenerateShaders (Visual * _vis)

Generate shaders for an entity.

Parameters

in	_vis	The visual to generate shaders for.
----	------	-------------------------------------

10.140.3.10 Ogre::PSSMShadowCameraSetup* gazebo::rendering::RTShaderSystem::GetPSSMShadowCameraSetup () const

Get the **Ogre** (p. 123) PSSM Shadows camera setup.

Returns

The **Ogre** (p. 123) PSSM Shadows camera setup.

10.140.3.11 void gazebo::rendering::RTShaderSystem::Init ()

Init the run time shader system.

10.140.3.12 `void gazebo::rendering::RTShaderSystem::RemoveScene (ScenePtr _scene)`

Remove a scene.

Parameters

<code>in</code>	<i>The</i>	scene to remove
-----------------	------------	-----------------

10.140.3.13 `void gazebo::rendering::RTShaderSystem::RemoveShadows (ScenePtr _scene)`

Remove shadows from a scene.

Parameters

<code>in</code>	<code>_scene</code>	The scene to remove shadows from.
-----------------	---------------------	-----------------------------------

10.140.3.14 `void gazebo::rendering::RTShaderSystem::SetPerPixelLighting (bool _set)`

Set the lighting model to per pixel or per vertex.

Parameters

<code>in</code>	<code>_set</code>	True means to use per-pixel shaders.
-----------------	-------------------	--------------------------------------

10.140.3.15 `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:

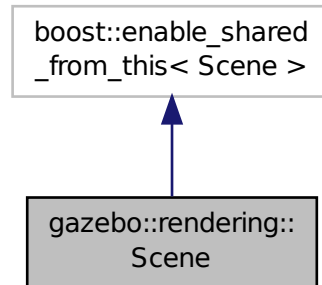
- **RTShaderSystem.hh**

10.141 gazebo::rendering::Scene Class Reference

Representation of an entire scene graph.

```
#include <rendering/rendering.hh>
```


Inheritance diagram for gazebo::rendering::Scene:



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. 728).
- **VisualPtr CloneVisual** (const std::string &_visualName, const std::string &_newName) **GAZEBO_DEPRECATED(1.10)**
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. 728) 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. 728).
- 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 **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. 123) 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.

10.141.1 Detailed Description

Representation of an entire scene graph.

Maintains all the Visuals, Lights, and Cameras for a World.

10.141.2 Member Enumeration Documentation

10.141.2.1 enum gazebo::rendering::Scene::SkyXMode

Enumerator

GZ_SKYX_ALL

GZ_SKYX_CLOUDS

GZ_SKYX_MOON

GZ_SKYX_NONE

10.141.3 Constructor & Destructor Documentation

10.141.3.1 `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>_enableVisualizations</i>	True to enable visualizations, this should be set to true for user interfaces, and false for sensor generation.

10.141.3.2 `virtual gazebo::rendering::Scene::~Scene () [virtual]`

Destructor.

10.141.4 Member Function Documentation

10.141.4.1 `void gazebo::rendering::Scene::AddVisual (VisualPtr _vis)`

Add a visual to the scene.

Parameters

in	<i>_vis</i>	Visual (p. 1034) to add.
----	-------------	---------------------------------

10.141.4.2 `void gazebo::rendering::Scene::Clear ()`

Clear **rendering::Scene** (p. 728).

10.141.4.3 `VisualPtr gazebo::rendering::Scene::CloneVisual (const std::string & _visualName, const std::string & _newName)`

Deprecated.

10.141.4.4 `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.

10.141.4.5 **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.

10.141.4.6 **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.

10.141.4.7 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.

10.141.4.8 **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. 979).
----	--------------	---

Returns

A pointer to the new **UserCamera** (p. 979).

10.141.4.9 `void gazebo::rendering::Scene::DrawLine (const math::Vector3 & _start, const math::Vector3 & _end, const std::string & _name)`

Draw a named line.

Parameters

<code>in</code>	<code>_start</code>	Start position of the line.
<code>in</code>	<code>_end</code>	End position of the line.
<code>in</code>	<code>_name</code>	Name of the line.

10.141.4.10 `common::Color gazebo::rendering::Scene::GetAmbientColor () const`

Get the ambient color.

Returns

The scene's ambient color.

10.141.4.11 `common::Color gazebo::rendering::Scene::GetBackgroundColor () const`

Get the background color.

Returns

The background color.

10.141.4.12 `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. 736).

Parameters

<code>in</code>	<code>_index</code>	Index of the camera to get.
-----------------	---------------------	-----------------------------

Returns

Pointer to the camera. Or NULL if the index is invalid.

10.141.4.13 `CameraPtr gazebo::rendering::Scene::GetCamera (const std::string & _name) const`

Get a camera by name.

Parameters

in	<i>_name</i>	Name of the camera.
----	--------------	---------------------

Returns

Pointer to the camera. Or NULL if the name is invalid.

10.141.4.14 `uint32_t gazebo::rendering::Scene::GetCameraCount () const`

Get the number of cameras in this scene.

Returns

Number of lasers.

10.141.4.15 `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	<i>_camera</i>	Pointer to the camera.
in	<i>_mousePos</i>	2D position of the mouse in pixels.
out	<i>_position</i>	3D position of the first contact point.

Returns

True if a valid object was hit by the raycast.

10.141.4.16 `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. 736).

Parameters

in	<i>_index</i>	Index of the grid.
----	---------------	--------------------

10.141.4.17 `uint32_t gazebo::rendering::Scene::GetGridCount () const`

Get the number of grids.

Returns

The number of grids.

10.141.4.18 `double gazebo::rendering::Scene::GetHeightBelowPoint (const math::Vector3 & _pt)`

Get the Z-value of the first object below the given point.

Parameters

<code>in</code>	<code>_pt</code>	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.

10.141.4.19 `Heightmap* gazebo::rendering::Scene::GetHeightmap () const`

Get a pointer to the heightmap.

Returns

Pointer to the heightmap, NULL if no heightmap.

10.141.4.20 `uint32_t gazebo::rendering::Scene::GetId () const`

Get the scene ID.

Returns

The ID of the scene.

10.141.4.21 `std::string gazebo::rendering::Scene::GetIdString () const`

Get the scene Id as a string.

Returns

The ID as a string.

10.141.4.22 `bool gazebo::rendering::Scene::GetInitialized () const`

Return true if the **Scene** (p. 728) has been initialized.

10.141.4.23 `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.

10.141.4.24 `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. 738).

Parameters

<code>in</code>	<code>_index</code>	Index of the light.
-----------------	---------------------	---------------------

Returns

Pointer to the **Light** (p. 448) or NULL if index was invalid.

10.141.4.25 `uint32_t gazebo::rendering::Scene::GetLightCount () const`

Get the count of the lights.

Returns

The number of lights.

10.141.4.26 `Ogre::SceneManager* gazebo::rendering::Scene::GetManager () const`

Get the OGRE scene manager.

Returns

Pointer to the **Ogre** (p. 123) SceneManager.

10.141.4.27 `VisualPtr gazebo::rendering::Scene::GetModelVisualAt (CameraPtr _camera, const math::Vector2i & _mousePos)`

Get a model's visual at a mouse position.

Parameters

<code>in</code>	<code>_camera</code>	Pointer to the camera used to project the mouse position.
<code>in</code>	<code>_mousePos</code>	The 2d position of the mouse in pixels.

Returns

Pointer to the visual, NULL if none found.

10.141.4.28 `std::string gazebo::rendering::Scene::GetName () const`

Get the name of the scene.

Returns

Name of the scene.

10.141.4.29 `VisualPtr gazebo::rendering::Scene::GetSelectedVisual () const`

Get the currently selected visual.

Returns

Pointer to the currently selected visual, or NULL if nothing is selected.

10.141.4.30 `bool gazebo::rendering::Scene::GetShadowsEnabled () const`

Get whether shadows are on or off.

Returns

True if shadows are enabled.

10.141.4.31 `bool gazebo::rendering::Scene::GetShowClouds () const`

Get whether or not clouds are displayed.

Returns

True if clouds are displayed.

10.141.4.32 `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. 728).

Returns

The current simulation time in **Scene** (p. 728)

10.141.4.33 `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. 740).

Parameters

in	_index	Index of the UserCamera (p. 979) to get.
----	--------	---

Returns

Pointer to the **UserCamera** (p. 979), or NULL if the index was invalid.

10.141.4.34 `uint32_t gazebo::rendering::Scene::GetUserCameraCount () const`

Get the number of user cameras in this scene.

Returns

The number of user cameras.

10.141.4.35 `VisualPtr gazebo::rendering::Scene::GetVisual (const std::string & _name) const`

Get a visual by name.

Parameters

in	_name	Name of the visual to retrieve.
----	-------	---------------------------------

Returns

Pointer to the visual, NULL if not found.

10.141.4.36 `VisualPtr gazebo::rendering::Scene::GetVisual (uint32_t _id) const`

Get a visual by id.

Parameters

in	_id	ID of the visual to retrieve.
----	-----	-------------------------------

Returns

Pointer to the visual, NULL if not found.

10.141.4.37 `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	_camera	The ogre camera, used to do mouse picking
in	_mousePos	The position of the mouse in screen coordinates
out	_mod	Used for object manipulation

Returns

The selected entity, or NULL

10.141.4.38 VisualPtr gazebo::rendering::Scene::GetVisualAt (CameraPtr *_camera*, const math::Vector2i & *_mousePos*)

Get a visual at a mouse position.

Parameters

in	<i>_camera</i>	Pointer to the camera used to project the mouse position.
in	<i>_mousePos</i>	The 2d position of the mouse in pixels.

Returns

Pointer to the visual, NULL if none found.

10.141.4.39 VisualPtr gazebo::rendering::Scene::GetVisualBelow (const std::string & *_visualName*)

Get the closest visual below a given visual.

Parameters

in	<i>_visualName</i>	Name of the visual to search below.
----	--------------------	-------------------------------------

Returns

Pointer to the visual below, or NULL if no visual.

10.141.4.40 uint32_t gazebo::rendering::Scene::GetVisualCount () const

Get the number of visuals.

Returns

The number of visuals in the **Scene** (p. 728).

10.141.4.41 void gazebo::rendering::Scene::GetVisualsBelowPoint (const math::Vector3 & *_pt*, std::vector< VisualPtr > & *_visuals*)

Get a visual directly below a point.

Parameters

in	<i>_pt</i>	3D point to get the visual below.
out	<i>_visuals</i>	The visuals below the point order in proximity.

10.141.4.42 **VisualPtr** gazebo::rendering::Scene::GetWorldVisual () const

Get the top level world visual.

Returns

Pointer to the world visual.

10.141.4.43 void gazebo::rendering::Scene::Init ()

Init **rendering::Scene** (p. 728).

10.141.4.44 void gazebo::rendering::Scene::Load (sdf::ElementPtr *_scene*)

Load the scene from a set of parameters.

Parameters

in	<i>_scene</i>	SDF scene element to load.
----	---------------	----------------------------

10.141.4.45 void gazebo::rendering::Scene::Load ()

Load the scene with default parameters.

10.141.4.46 void gazebo::rendering::Scene::PreRender ()

Process all received messages.

10.141.4.47 void gazebo::rendering::Scene::PrintSceneGraph ()

Print the scene graph to std_out.

10.141.4.48 void gazebo::rendering::Scene::RemoveCamera (const std::string & *_name*)

Remove a camera from the scene.

Parameters

in	<i>_name</i>	Name of the camera.
----	--------------	---------------------

10.141.4.49 void gazebo::rendering::Scene::RemoveVisual (VisualPtr *_vis*)

Remove a visual from the scene.

Parameters

in	<i>_vis</i>	Visual (p. 1034) to remove.
----	-------------	------------------------------------

10.141.4.50 `void gazebo::rendering::Scene::SelectVisual (const std::string & _name, const std::string & _mode)`

Select a visual by name.

Parameters

<code>in</code>	<code>_name</code>	Name of the visual to select.
<code>in</code>	<code>_mode</code>	Selection mode (normal, or move).

10.141.4.51 `void gazebo::rendering::Scene::SetAmbientColor (const common::Color & _color)`

Set the ambient color.

Parameters

<code>in</code>	<code>_color</code>	The ambient color to use.
-----------------	---------------------	---------------------------

10.141.4.52 `void gazebo::rendering::Scene::SetBackgroundColor (const common::Color & _color)`

Set the background color.

Parameters

<code>in</code>	<code>_color</code>	The background color.
-----------------	---------------------	-----------------------

10.141.4.53 `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>_type</code>	Type of fog: "linear", "exp", or "exp2".
<code>in</code>	<code>_color</code>	Color of the fog.
<code>in</code>	<code>_density</code>	Fog density.
<code>in</code>	<code>_start</code>	Distance from camera to start the fog.
<code>in</code>	<code>_end</code>	Distance from camera at which the fog is at max density.

10.141.4.54 `void gazebo::rendering::Scene::SetGrid (bool _enabled)`

Set the grid on or off.

Parameters

<code>in</code>	<code>_enabled</code>	Set to true to turn on the grid
-----------------	-----------------------	---------------------------------

10.141.4.55 `void gazebo::rendering::Scene::SetShadowsEnabled (bool _value)`

Set whether shadows are on or off.

Parameters

<code>in</code>	<code>_value</code>	True to enable shadows, False to disable
-----------------	---------------------	--

10.141.4.56 `void gazebo::rendering::Scene::SetSkyXMode (unsigned int _mode)`

Set **SkyX** (p. 123) mode to enable/disable skyx components such as clouds and moon.

Parameters

<code>in</code>	<code>_mode</code>	SkyX (p. 123) mode bitmask.
-----------------	--------------------	------------------------------------

See Also

Scene::SkyXMode (p. 732)

10.141.4.57 `void gazebo::rendering::Scene::SetTransparent (bool _show)`

Enable or disable transparency for all visuals.

Parameters

<code>in</code>	<code>_show</code>	True to enable transparency for all visuals.
-----------------	--------------------	--

10.141.4.58 `void gazebo::rendering::Scene::SetVisible (const std::string & _name, bool _visible)`

Hide or show a visual.

Parameters

<code>in</code>	<code>_name</code>	Name of the visual to change.
<code>in</code>	<code>_visible</code>	True to make visual visible, False to make it invisible.

10.141.4.59 `void gazebo::rendering::Scene::SetWireframe (bool _show)`

Enable or disable wireframe for all visuals.

Parameters

<code>in</code>	<code>_show</code>	True to enable wireframe for all visuals.
-----------------	--------------------	---

10.141.4.60 `void gazebo::rendering::Scene::ShowClouds (bool _show)`

Display clouds in the sky.

Parameters

in	<code>_show</code>	True to display clouds.
----	--------------------	-------------------------

10.141.4.61 void gazebo::rendering::Scene::ShowCollisions (bool *_show*)

Enable or disable collision visualization.

Parameters

in	<code>_show</code>	True to enable collision visualization.
----	--------------------	---

10.141.4.62 void gazebo::rendering::Scene::ShowCOMs (bool *_show*)

Enable or disable center of mass visualization.

Parameters

in	<code>_show</code>	True to enable center of mass visualization.
----	--------------------	--

10.141.4.63 void gazebo::rendering::Scene::ShowContacts (bool *_show*)

Enable or disable contact visualization.

Parameters

in	<code>_show</code>	True to enable contact visualization.
----	--------------------	---------------------------------------

10.141.4.64 void gazebo::rendering::Scene::ShowJoints (bool *_show*)

Enable or disable joint visualization.

Parameters

in	<code>_show</code>	True to enable joint visualization.
----	--------------------	-------------------------------------

10.141.4.65 void gazebo::rendering::Scene::SnapVisualToNearestBelow (const std::string & *_visualName*)

Move the visual to be ontop of the nearest visual below it.

Parameters

in	<code>_visualName</code>	Name of the visual to move.
----	--------------------------	-----------------------------

10.141.4.66 std::string gazebo::rendering::Scene::StripSceneName (const std::string & *_name*) const

Remove the name of scene from a string.

Parameters

in	<code>_name</code>	Name to string the scene name from.
----	--------------------	-------------------------------------

Returns

The stripped name.

10.141.5 Member Data Documentation

10.141.5.1 SkyX::SkyX* gazebo::rendering::Scene::skyx

Pointer to the sky.

The documentation for this class was generated from the following file:

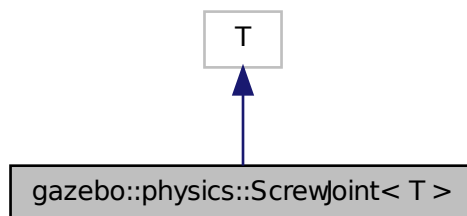
- **Scene.hh**

10.142 gazebo::physics::ScrewJoint< T > Class Template Reference

A screw joint, which has both prismatic and rotational DOFs.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::ScrewJoint< T >:



Public Member Functions

- **ScrewJoint** (**BasePtr** _parent)
Constructor.
- virtual **~ScrewJoint** ()
Destructor.
- virtual **math::Vector3** **GetAnchor** (int _index) const
Get the anchor.
- virtual unsigned int **GetAngleCount** () const

- virtual double **GetThreadPitch** (unsigned int _index)=0
Get screw joint thread pitch.
- virtual void **Load** (sdf::ElementPtr _sdf)
*Load a **ScrewJoint** (p. 746).*
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)
Set the anchor.
- virtual void **SetThreadPitch** (int _index, double _threadPitch)=0
Set screw joint thread pitch.

Protected Attributes

- **math::Vector3 fakeAnchor**
The anchor value is not used internally.
- double **threadPitch**
Pitch of the thread.

10.142.1 Detailed Description

```
template<class T>class gazebo::physics::ScrewJoint< T >
```

A screw joint, which has both prismatic and rotational DOFs.

10.142.2 Constructor & Destructor Documentation

10.142.2.1 `template<class T> gazebo::physics::ScrewJoint< T >::ScrewJoint (BasePtr _parent) [inline], [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Parent of the joint.
----	----------------------	----------------------

10.142.2.2 `template<class T> virtual gazebo::physics::ScrewJoint< T >::~~ScrewJoint () [inline], [virtual]`

Destructor.

10.142.3 Member Function Documentation

10.142.3.1 `template<class T > math::Vector3 gazebo::physics::ScrewJoint< T >::GetAnchor (int _index) const [virtual]`

Get the anchor.

Parameters

in	<code>_index</code>	Index of the axis. Not Used.
----	---------------------	------------------------------

Returns

Anchor for the joint.

10.142.3.2 `template<class T> virtual unsigned int gazebo::physics::ScrewJoint< T >::GetAngleCount () const`
`[inline], [virtual]`

10.142.3.3 `template<class T> virtual double gazebo::physics::ScrewJoint< T >::GetThreadPitch (unsigned int _index)`
`[pure virtual]`

Get screw joint thread pitch.

This must be implemented in a child class

Parameters

in	<code>_index</code>	Index of the axis.
----	---------------------	--------------------

Returns

`_threadPitch` Thread pitch value.

Implemented in `gazebo::physics::SimbodyScrewJoint` (p. 846).

10.142.3.4 `template<class T> virtual void gazebo::physics::ScrewJoint< T >::Load (sdf::ElementPtr _sdf)` `[inline],`
`[virtual]`

Load a **ScrewJoint** (p. 746).

Parameters

in	<code>_sdf</code>	SDF value to load from
----	-------------------	------------------------

Reimplemented in `gazebo::physics::SimbodyScrewJoint` (p. 847).

10.142.3.5 `template<class T > void gazebo::physics::ScrewJoint< T >::SetAnchor (int _index, const math::Vector3 &`
`_anchor)` `[virtual]`

Set the anchor.

Parameters

in	<code>_index</code>	Index of the axis. Not Used.
in	<code>_anchor</code>	Anchor value for the joint.

10.142.3.6 `template<class T> virtual void gazebo::physics::ScrewJoint< T >::SetThreadPitch (int _index, double _threadPitch) [pure virtual]`

Set screw joint thread pitch.

This must be implemented in a child class

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_threadPitch</i>	Thread pitch value.

Implemented in `gazebo::physics::SimbodyScrewJoint` (p. 849).

10.142.4 Member Data Documentation

10.142.4.1 `template<class T> math::Vector3 gazebo::physics::ScrewJoint< T >::fakeAnchor [protected]`

The anchor value is not used internally.

10.142.4.2 `template<class T> double gazebo::physics::ScrewJoint< T >::threadPitch [protected]`

Pitch of the thread.

Referenced by `gazebo::physics::ScrewJoint< SimbodyJoint >::Load()`.

The documentation for this class was generated from the following file:

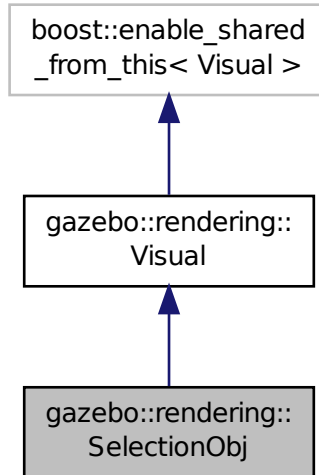
- `ScrewJoint.hh`

10.143 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

10.143.1 Detailed Description

Interactive selection object for models and links.

The documentation for this class was generated from the following file:

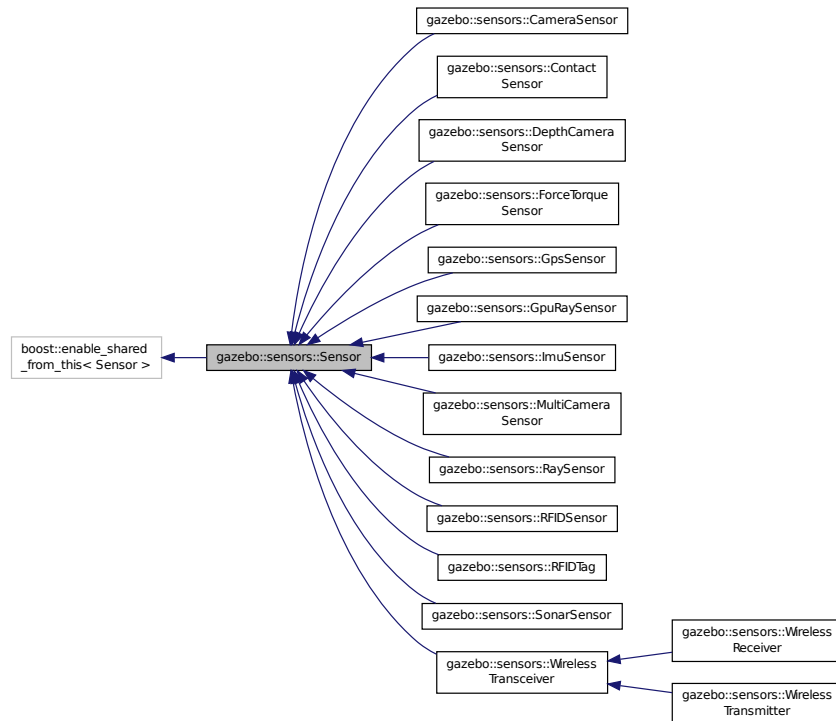
- **SelectionObj.hh**

10.144 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.
- **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(1.10)**
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

- **virtual void 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.
- **boost::mutex mutexLastUpdateTime**
Mutex to protect resetting lastUpdateTime.
- **transport::NodePtr node**
Node for communication.
- **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. 760).*
- **gazebo::physics::WorldPtr world**
Pointer to the world.

10.144.1 Detailed Description

Base class for sensors.

10.144.2 Constructor & Destructor Documentation

10.144.2.1 gazebo::sensors::Sensor::Sensor (SensorCategory _cat) [explicit]

Constructor.

Parameters

in	_class	
----	--------	--

10.144.2.2 `virtual gazebo::sensors::Sensor::~~Sensor() [virtual]`

Destructor.

10.144.3 Member Function Documentation

10.144.3.1 `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. 755)

References `gazebo::event::EventT< T >::Connect()`.

10.144.3.2 `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. 755)

References `gazebo::event::EventT< T >::Disconnect()`.

10.144.3.3 `void gazebo::sensors::Sensor::FillMsg (msgs::Sensor & _msg)`

fills a `msgs::Sensor` message.

Parameters

out	_msg	Message to fill.
-----	------	------------------

10.144.3.4 `virtual void gazebo::sensors::Sensor::Fini () [virtual]`

Finalize the sensor.

Reimplemented in [gazebo::sensors::MultiCameraSensor](#) (p. 575), [gazebo::sensors::ForceTorqueSensor](#) (p. 336), [gazebo::sensors::GpuRaySensor](#) (p. 357), [gazebo::sensors::CameraSensor](#) (p. 208), [gazebo::sensors::ContactSensor](#) (p. 262), [gazebo::sensors::RFIDSensor](#) (p. 711), [gazebo::sensors::RaySensor](#) (p. 695), [gazebo::sensors::DepthCameraSensor](#) (p. 278), [gazebo::sensors::RFIDTag](#) (p. 714), [gazebo::sensors::GpsSensor](#) (p. 342), [gazebo::sensors::SonarSensor](#) (p. 893), [gazebo::sensors::ImuSensor](#) (p. 396), [gazebo::sensors::WirelessTransceiver](#) (p. 1065), and [gazebo::sensors::WirelessReceiver](#) (p. 1062).

10.144.3.5 `SensorCategory` `gazebo::sensors::Sensor::GetCategory () const`

Get the category of the sensor.

Returns

The category of the sensor.

See Also

[SensorCategory](#) (p. 119)

10.144.3.6 `uint32_t` `gazebo::sensors::Sensor::GetId () const`

Get the sensor's ID.

Returns

The sensor's ID.

10.144.3.7 `common::Time` `gazebo::sensors::Sensor::GetLastMeasurementTime ()`

Return last measurement time.

Returns

Time of last measurement.

10.144.3.8 `common::Time` `gazebo::sensors::Sensor::GetLastUpdateTime ()`

Return last update time.

Returns

Time of last update.

10.144.3.9 `std::string` `gazebo::sensors::Sensor::GetName () const`

Get name.

Returns

Name of sensor.

10.144.3.10 `uint32_t gazebo::sensors::Sensor::GetParentId () const`

Get the sensor's parent's ID.

Returns

The sensor's parent's ID.

10.144.3.11 `std::string gazebo::sensors::Sensor::GetParentName () const`

Returns the name of the sensor parent.

The parent name is set by **Sensor::SetParent** (p. 760).

Returns

Name of Parent.

10.144.3.12 `virtual math::Pose gazebo::sensors::Sensor::GetPose () const` [virtual]

Get the current pose.

Returns

Current pose of the sensor.

10.144.3.13 `std::string gazebo::sensors::Sensor::GetScopedName () const`

Get fully scoped name of the sensor.

Returns

`world_name::parent_name::sensor_name`.

10.144.3.14 `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. 361), **gazebo::sensors::RaySensor** (p. 698), **gazebo::sensors::CameraSensor** (p. 208), **gazebo::sensors::SonarSensor** (p. 894), **gazebo::sensors::MultiCameraSensor** (p. 577), **gazebo::sensors::ForceTorqueSensor** (p. 336), and **gazebo::sensors::WirelessTransceiver** (p. 1065).

10.144.3.15 `std::string gazebo::sensors::Sensor::GetType () const`

Get sensor type.

Returns

Type of sensor.

10.144.3.16 `double gazebo::sensors::Sensor::GetUpdateRate ()`

Get the update rate of the sensor.

Returns

_hz update rate of sensor. Returns 0 if unthrottled.

10.144.3.17 `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.

10.144.3.18 `std::string gazebo::sensors::Sensor::GetWorldName () const`

Returns the name of the world the sensor is in.

Returns

Name of the world.

10.144.3.19 `virtual void gazebo::sensors::Sensor::Init () [virtual]`

Initialize the sensor.

Reimplemented in **`gazebo::sensors::GpuRaySensor`** (p. 362), **`gazebo::sensors::ContactSensor`** (p. 264), **`gazebo::sensors::RFIDSensor`** (p. 711), **`gazebo::sensors::RaySensor`** (p. 699), **`gazebo::sensors::CameraSensor`** (p. 209), **`gazebo::sensors::DepthCameraSensor`** (p. 278), **`gazebo::sensors::WirelessTransmitter`** (p. 1069), **`gazebo::sensors::RFIDTag`** (p. 714), **`gazebo::sensors::GpsSensor`** (p. 343), **`gazebo::sensors::SonarSensor`** (p. 894), **`gazebo::sensors::MultiCameraSensor`** (p. 577), **`gazebo::sensors::ImuSensor`** (p. 397), **`gazebo::sensors::WirelessTransceiver`** (p. 1065), **`gazebo::sensors::ForceTorqueSensor`** (p. 336), and **`gazebo::sensors::WirelessReceiver`** (p. 1063).

10.144.3.20 `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. 362), **gazebo::sensors::RaySensor** (p. 699), **gazebo::sensors::ContactSensor** (p. 264), **gazebo::sensors::CameraSensor** (p. 209), **gazebo::sensors::MultiCameraSensor** (p. 577), **gazebo::sensors::SonarSensor** (p. 894), **gazebo::sensors::ImuSensor** (p. 397), and **gazebo::sensors::ForceTorqueSensor** (p. 336).

10.144.3.21 `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. 751) parameters.
in	<code>_worldName</code>	Name of world to load from.

Reimplemented in **gazebo::sensors::ContactSensor** (p. 264), **gazebo::sensors::RFIDSensor** (p. 712), **gazebo::sensors::CameraSensor** (p. 209), and **gazebo::sensors::ImuSensor** (p. 398).

10.144.3.22 `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. 363), **gazebo::sensors::ContactSensor** (p. 264), **gazebo::sensors::RFIDSensor** (p. 712), **gazebo::sensors::RaySensor** (p. 699), **gazebo::sensors::CameraSensor** (p. 209), **gazebo::sensors::DepthCameraSensor** (p. 278), **gazebo::sensors::WirelessTransmitter** (p. 1069), **gazebo::sensors::RFIDTag** (p. 714), **gazebo::sensors::GpsSensor** (p. 343), **gazebo::sensors::SonarSensor** (p. 894), **gazebo::sensors::MultiCameraSensor** (p. 577), **gazebo::sensors::ImuSensor** (p. 398), **gazebo::sensors::WirelessTransceiver** (p. 1065), **gazebo::sensors::ForceTorqueSensor** (p. 337), and **gazebo::sensors::WirelessReceiver** (p. 1063).

10.144.3.23 `void gazebo::sensors::Sensor::ResetLastUpdateTime ()`

Reset the lastUpdateTime to zero.

10.144.3.24 `virtual void gazebo::sensors::Sensor::SetActive (bool _value) [virtual]`

Set whether the sensor is active or not.

Parameters

in	<code>_value</code>	True if active, false if not.
----	---------------------	-------------------------------

Reimplemented in **gazebo::sensors::DepthCameraSensor** (p. 279).

10.144.3.25 `virtual void gazebo::sensors::Sensor::SetParent (const std::string & _name) [virtual]`

Set the parent of the sensor.

Parameters

in	_name	Name of the parent.
----	-------	---------------------

10.144.3.26 `void gazebo::sensors::Sensor::SetParent (const std::string & _name, uint32_t _id)`

Set the sensor's parent.

Parameters

in	_name	The sensor's parent's name.
in	_id	The sensor's parent's ID.

10.144.3.27 `void gazebo::sensors::Sensor::SetUpdateRate (double _hz)`

Set the update rate of the sensor.

Parameters

in	_hz	update rate of sensor.
----	-----	------------------------

10.144.3.28 `void gazebo::sensors::Sensor::Update (bool _force)`

Update the sensor.

Parameters

in	_force	True to force update, false otherwise.
----	--------	--

10.144.3.29 `virtual void 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

in	_force	True if update is forced, false if not
----	--------	--

Reimplemented in `gazebo::sensors::MultiCameraSensor` (p. 578), `gazebo::sensors::ForceTorqueSensor` (p. 337), `gazebo::sensors::GpuRaySensor` (p. 364), `gazebo::sensors::CameraSensor` (p. 210), `gazebo::sensors::ContactSensor` (p. 265), `gazebo::sensors::RFIDSensor` (p. 712), `gazebo::sensors::RaySensor` (p. 699), `gazebo::sensors::DepthCameraSensor` (p. 279), `gazebo::sensors::RFIDTag` (p. 714), `gazebo::sensors::GpsSensor`

(p. 343), **gazebo::sensors::SonarSensor** (p. 894), **gazebo::sensors::WirelessTransmitter** (p. 1069), and **gazebo::sensors::ImuSensor** (p. 398).

10.144.4 Member Data Documentation

10.144.4.1 **bool gazebo::sensors::Sensor::active** [protected]

True if sensor generation is active.

10.144.4.2 **std::vector<event::ConnectionPtr> gazebo::sensors::Sensor::connections** [protected]

All event connections.

10.144.4.3 **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.

10.144.4.4 **common::Time gazebo::sensors::Sensor::lastUpdateTime** [protected]

Time of the last update.

10.144.4.5 **boost::mutex gazebo::sensors::Sensor::mutexLastUpdateTime** [protected]

Mutex to protect resetting lastUpdateTime.

10.144.4.6 **transport::NodePtr gazebo::sensors::Sensor::node** [protected]

Node for communication.

10.144.4.7 **uint32_t gazebo::sensors::Sensor::parentId** [protected]

The sensor's parent ID.

10.144.4.8 **std::string gazebo::sensors::Sensor::parentName** [protected]

Name of the parent.

10.144.4.9 **std::vector<SensorPluginPtr> gazebo::sensors::Sensor::plugins** [protected]

All the plugins for the sensor.

10.144.4.10 **math::Pose gazebo::sensors::Sensor::pose** [protected]

Pose of the sensor.

10.144.4.11 `transport::SubscriberPtr gazebo::sensors::Sensor::poseSub` [protected]

Subscribe to pose updates.

10.144.4.12 `gazebo::rendering::ScenePtr gazebo::sensors::Sensor::scene` [protected]

Pointer to the Scene.

10.144.4.13 `sdf::ElementPtr gazebo::sensors::Sensor::sdf` [protected]

Pointer the the SDF element for the sensor.

10.144.4.14 `common::Time gazebo::sensors::Sensor::updatePeriod` [protected]

Desired time between updates, set indirectly by `Sensor::SetUpdateRate` (p. 760).

10.144.4.15 `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`

10.145 SensorFactor Class Reference

The sensor factory; the class is just for namespacing purposes.

```
#include <sensors/sensors.hh>
```

10.145.1 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`

10.146 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).

10.146.1 Member Function Documentation

10.146.1.1 static void gazebo::sensors::SensorFactory::GetSensorTypes (std::vector< std::string > &_types) [static]

Get all the sensor types.

Parameters

_types	Vector of strings of the sensor types, populated by function
--------	--

10.146.1.2 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

in	_className	Name of sensor class
----	------------	----------------------

Returns

Pointer to **Sensor** (p. 751)

10.146.1.3 static void gazebo::sensors::SensorFactory::RegisterAll () [static]

Register all known sensors.

- **sensors::CameraSensor** (p. 206)
- **sensors::DepthCameraSensor** (p. 276)
- **sensors::GpuRaySensor** (p. 353)
- **sensors::RaySensor** (p. 693)
- **sensors::ContactSensor** (p. 260)
- **sensors::RFIDSensor** (p. 710)
- **sensors::RFIDTag** (p. 712)
- **sensors::WirelessTransmitter** (p. 1066)
- **sensors::WirelessReceiver** (p. 1060)

10.146.1.4 `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:

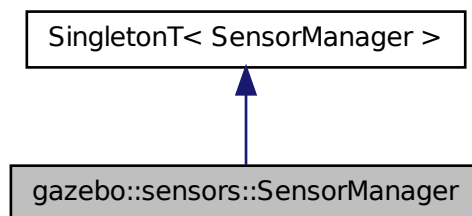
- **SensorFactory.hh**

10.147 gazebo::sensors::SensorManager Class Reference

Class to manage and update all sensors.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::SensorManager:



Public Member Functions

- `std::string CreateSensor (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName) GAZEBO_DEPRECATED(1.10)`
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

10.147.1 Detailed Description

Class to manage and update all sensors.

10.147.2 Member Function Documentation

10.147.2.1 `std::string gazebo::sensors::SensorManager::CreateSensor (sdf::ElementPtr _elem, const std::string & _worldName, const std::string & _parentName)`

Deprecated.

10.147.2.2 `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

in	<code>_elem</code>	The SDF element that describes the sensor
in	<code>_worldName</code>	Name of the world in which to create the sensor
in	<code>_parentName</code>	The name of the parent link which the sensor is attached to.

Returns

The name of the sensor

10.147.2.3 void gazebo::sensors::SensorManager::Fini ()

Finalize all the sensors.

10.147.2.4 **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.

10.147.2.5 **Sensor_V** gazebo::sensors::SensorManager::GetSensors () const

Get all the sensors.

Returns

Vector of all the sensors.

10.147.2.6 void gazebo::sensors::SensorManager::GetSensorTypes (std::vector< std::string > & *_types*) const

Get all the sensor types.

Parameters

out	<i>All</i>	the sensor types.
-----	------------	-------------------

10.147.2.7 void gazebo::sensors::SensorManager::Init ()

Init all the sensors.

10.147.2.8 void gazebo::sensors::SensorManager::RemoveSensor (const std::string & *_name*)

Remove a sensor.

Parameters

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

10.147.2.9 void gazebo::sensors::SensorManager::RemoveSensors ()

Remove all sensors.

10.147.2.10 void gazebo::sensors::SensorManager::ResetLastUpdateTimes ()

Reset last update times in all sensors.

10.147.2.11 void gazebo::sensors::SensorManager::RunThreads ()

Run sensor updates in separate threads.

This will only run non-image based sensor updates.

10.147.2.12 bool gazebo::sensors::SensorManager::SensorsInitialized ()

True if SensorManager::initSensors queue is empty i.e.

all sensors managed by **SensorManager** (p. 764) have been initialized

10.147.2.13 void gazebo::sensors::SensorManager::Stop ()

Stop the run thread.

10.147.2.14 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

in	<i>_force</i>	True force update, false if not
----	---------------	---------------------------------

The documentation for this class was generated from the following file:

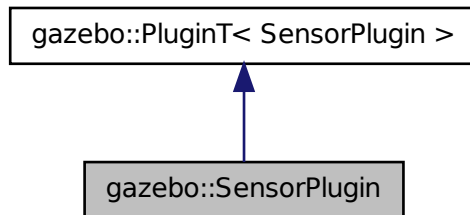
- **SensorManager.hh**

10.148 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

10.148.1 Detailed Description

A plugin with access to physics::Sensor.

See [reference](#).

10.148.2 Constructor & Destructor Documentation

10.148.2.1 gazebo::SensorPlugin::SensorPlugin () [inline]

Constructor.

References [gazebo::SENSOR_PLUGIN](#), and [gazebo::PluginT< SensorPlugin >::type](#).

10.148.2.2 virtual gazebo::SensorPlugin::~~SensorPlugin () [inline],[virtual]

Destructor.

10.148.3 Member Function Documentation

10.148.3.1 virtual void gazebo::SensorPlugin::Init () [inline],[virtual]

Override this method for custom plugin initialization behavior.

10.148.3.2 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

in	<code>_sensor</code>	Pointer the Sensor.
in	<code>_sdf</code>	Pointer the the SDF element of the plugin.

10.148.3.3 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**

10.149 gazebo::Server Class Reference

```
#include <Server.hh>
```

Public Member Functions

- **Server** ()
- virtual **~Server** ()
- void **Fini** ()
- bool **GetInitialized** () const
- void **Init** ()
- 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)
- bool **ParseArgs** (int argc, char **argv)
- bool **PreLoad** ()
Preload the server.
- void **PrintUsage** ()
- void **Run** ()
- void **SetParams** (const common::StrStr_M ¶ms)
- void **Stop** ()

Public Attributes

- int **systemPluginsArgc**
- char ** **systemPluginsArgv**

10.149.1 Constructor & Destructor Documentation

10.149.1.1 gazebo::Server::Server ()

10.149.1.2 virtual gazebo::Server::~Server () [virtual]

10.149.2 Member Function Documentation

10.149.2.1 void gazebo::Server::Fini ()

10.149.2.2 bool gazebo::Server::GetInitialized () const

10.149.2.3 void gazebo::Server::Init ()

10.149.2.4 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>	Type of physics engine to use (ode bullet simbody).

10.149.2.5 bool gazebo::Server::LoadString (const std::string & *_sdfString*)

10.149.2.6 bool gazebo::Server::ParseArgs (int *argc*, char ** *argv*)

10.149.2.7 bool gazebo::Server::PreLoad ()

Preload the server.

Returns

True if load was successful.

10.149.2.8 void gazebo::Server::PrintUsage ()

10.149.2.9 void gazebo::Server::Run ()

10.149.2.10 void gazebo::Server::SetParams (const common::StrStr_M & *params*)

10.149.2.11 void gazebo::Server::Stop ()

10.149.3 Member Data Documentation

10.149.3.1 int gazebo::Server::systemPluginsArgc

10.149.3.2 char** gazebo::Server::systemPluginsArgv

The documentation for this class was generated from the following file:

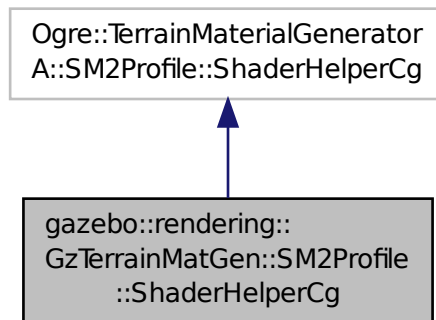
- **Server.hh**

10.150 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg Class Reference

Keeping the CG shader for reference.

```
#include <Heightmap.hh>
```

Inheritance diagram for gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg:



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*)

10.150.1 Detailed Description

Keeping the CG shader for reference.

Utility class to help with generating shaders for Cg / HLSL.

10.150.2 Member Function Documentation

- 10.150.2.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::defaultVpParams (const **SM2Profile** * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, const Ogre::HighLevelGpuProgramPtr & *_prog*)
[protected], [virtual]
- 10.150.2.2 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateFragmentProgram (const **SM2Profile** * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*)
[virtual]
- 10.150.2.3 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgram (const **SM2Profile** * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*)
[virtual]
- 10.150.2.4 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgramSource (const **SM2Profile** * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*)
[protected], [virtual]
- 10.150.2.5 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpDynamicShadows (const **SM2Profile** * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*)
[protected], [virtual]
- 10.150.2.6 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]
- 10.150.2.7 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpFooter (const **SM2Profile** * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*)
[protected], [virtual]
- 10.150.2.8 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:

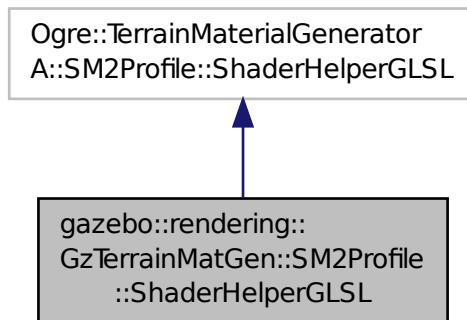
- **Heightmap.hh**

10.151 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL Class Reference

Utility class to help with generating shaders for GLSL.

```
#include <Heightmap.hh>
```

Inheritance diagram for gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL:



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)

10.151.1 Detailed Description

Utility class to help with generating shaders for GLSL.

10.151.2 Member Function Documentation

- 10.151.2.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::defaultVpParams (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, const Ogre::HighLevelGpuProgramPtr & _prog) [protected], [virtual]
- 10.151.2.2 void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadows (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream) [protected]
- 10.151.2.3 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadowsHelpers (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & _outStream) [protected], [virtual]
- 10.151.2.4 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]
- 10.151.2.5 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpFooter (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & _outStream) [protected], [virtual]
- 10.151.2.6 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpHeader (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & _outStream) [protected], [virtual]
- 10.151.2.7 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]

- 10.151.2.8 virtual `Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFragmentProgram (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt)` [virtual]
- 10.151.2.9 virtual void `gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFragmentProgramSource (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.151.2.10 virtual `Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVertexProgram (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt)` [virtual]
- 10.151.2.11 virtual void `gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVertexProgramSource (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.151.2.12 virtual void `gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpDynamicShadows (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.151.2.13 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]
- 10.151.2.14 virtual void `gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpFooter (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.151.2.15 virtual void `gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpHeader (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.151.2.16 virtual void `gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::updateParams (const SM2Profile * _prof, const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain, bool _compositeMap)` [virtual]
- 10.151.2.17 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:

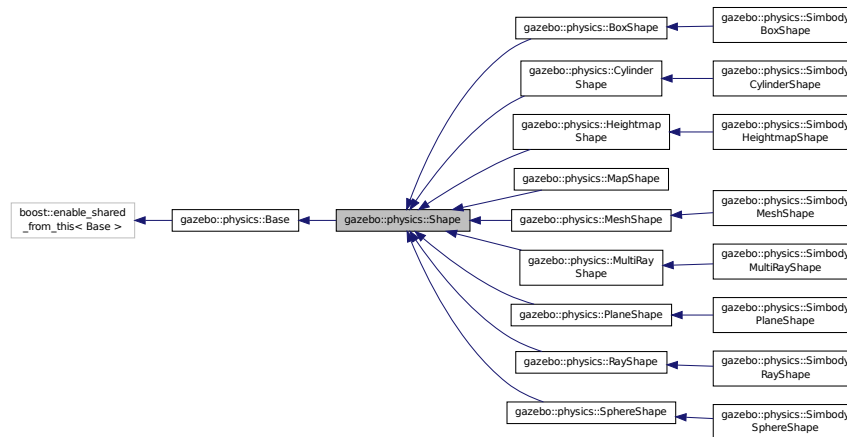
- [Heightmap.hh](#)

10.152 gazebo::physics::Shape Class Reference

Base (p. 153) class for all shapes.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::Shape:



Public Member Functions

- **Shape** (**CollisionPtr** _parent)
Constructor.
- virtual \sim **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

10.152.1 Detailed Description

Base (p. 153) class for all shapes.

10.152.2 Constructor & Destructor Documentation

10.152.2.1 gazebo::physics::Shape::Shape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent of the shape.
----	----------------	----------------------

10.152.2.2 virtual gazebo::physics::Shape::~~Shape () [virtual]

Destructor.

10.152.3 Member Function Documentation

10.152.3.1 virtual void gazebo::physics::Shape::FillMsg (msgs::Geometry & *_msg*) [pure virtual]

Fill in the values for a geometry message.

Parameters

out	<i>_msg</i>	The geometry message to fill.
-----	-------------	-------------------------------

Implemented in **gazebo::physics::MultiRayShape** (p.582), **gazebo::physics::RayShape** (p.702), **gazebo::physics::HeightmapShape** (p.383), **gazebo::physics::PlaneShape** (p.644), **gazebo::physics::MeshShape** (p.535), **gazebo::physics::CylinderShape** (p.270), **gazebo::physics::MapShape** (p.494), **gazebo::physics::SphereShape** (p.900), and **gazebo::physics::BoxShape** (p.171).

10.152.3.2 virtual math::Vector3 gazebo::physics::Shape::GetScale () const [virtual]

Get the scale of the shape.

Returns

Scale of the shape.

Reimplemented in **gazebo::physics::MapShape** (p.495).

10.152.3.3 virtual void gazebo::physics::Shape::Init () [pure virtual]

Initialize the shape.

Reimplemented from **gazebo::physics::Base** (p.160).

Implemented in **gazebo::physics::RayShape** (p.704), **gazebo::physics::MapShape** (p.495), **gazebo::physics::HeightmapShape** (p.384), **gazebo::physics::MeshShape** (p.536), **gazebo::physics::MultiRayShape** (p.585), **gazebo::physics::PlaneShape** (p.645), **gazebo::physics::SphereShape** (p.900), **gazebo::physics::BoxShape** (p.171), **gazebo::physics::CylinderShape** (p.271), **gazebo::physics::SimbodyHeightmapShape** (p.792), and **gazebo::physics::SimbodyMeshShape** (p.824).

10.152.3.4 `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.585), `gazebo::physics::RayShape` (p.704), `gazebo::physics::HeightmapShape` (p.385), `gazebo::physics::PlaneShape` (p.645), `gazebo::physics::MeshShape` (p.536), `gazebo::physics::CylinderShape` (p.271), `gazebo::physics::MapShape` (p.496), `gazebo::physics::SphereShape` (p.900), and `gazebo::physics::BoxShape` (p.171).

10.152.3.5 `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.705), `gazebo::physics::MapShape` (p.496), `gazebo::physics::PlaneShape` (p.646), `gazebo::physics::MeshShape` (p.536), `gazebo::physics::CylinderShape` (p.271), `gazebo::physics::HeightmapShape` (p.385), `gazebo::physics::SphereShape` (p.901), `gazebo::physics::BoxShape` (p.172), and `gazebo::physics::MultiRayShape` (p.585).

10.152.4 Member Data Documentation

10.152.4.1 `CollisionPtr gazebo::physics::Shape::collisionParent [protected]`

This shape's collision parent.

Referenced by `gazebo::physics::SimbodySphereShape::SetRadius()`, `gazebo::physics::SimbodyBoxShape::SetSize()`, and `gazebo::physics::SimbodyCylinderShape::SetSize()`.

10.152.4.2 `math::Vector3 gazebo::physics::Shape::scale [protected]`

This shape's scale;

The documentation for this class was generated from the following file:

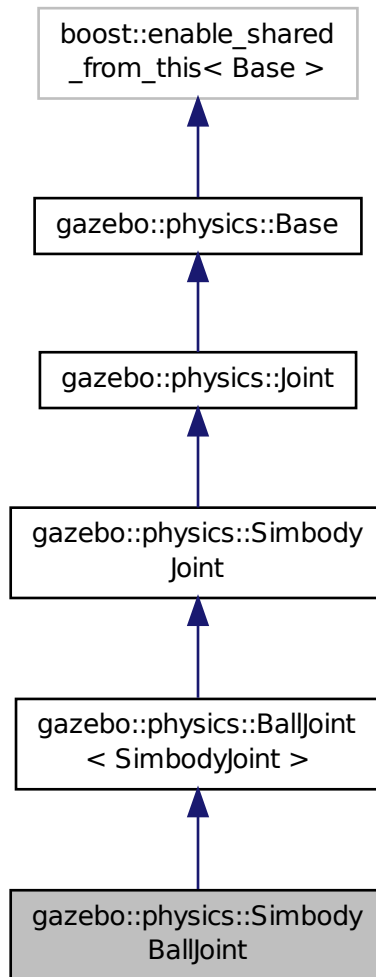
- `Shape.hh`

10.153 gazebo::physics::SimbodyBallJoint Class Reference

`SimbodyBallJoint` (p.778) 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. 411) *Constructor.*
- virtual **~SimbodyBallJoint** ()
Destructor.
- **math::Vector3 GetAnchor** (int _index) const
Get the anchor point.
- virtual **math::Angle GetAngleImpl** (int _index) const
Get the angle of an axis helper function.
- virtual **math::Vector3 GetAxis** (int) const

- virtual **math::Vector3 GetGlobalAxis** (int _index) const
Get the axis of rotation in global coordinate frame.
- virtual double **GetMaxForce** (int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int _index) const
Get the rotation rate of an axis(index)
- virtual void **Init** ()
Initialize a joint.
- virtual void **Load** (sdf::ElementPtr _sdf)
*Load **physics::Joint** (p. 411) from a SDF sdf::Element.*
- virtual void **SetDamping** (int _index, double _damping)
Set joint damping, not yet implemented.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)
Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)
Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _angle)
Set the velocity of an axis(index).

Protected Member Functions

- virtual void **SetForceImpl** (int _index, double _torque)
*Set the force applied to this **physics::Joint** (p. 411).*

Additional Inherited Members

10.153.1 Detailed Description

SimbodyBallJoint (p. 778) class models a ball joint in Simbody.

10.153.2 Constructor & Destructor Documentation

10.153.2.1 **gazebo::physics::SimbodyBallJoint::SimbodyBallJoint** (**SimTK::MultibodySystem** * *_world*, **BasePtr** *_parent*)

Simbody Ball **Joint** (p. 411) Constructor.

10.153.2.2 **virtual gazebo::physics::SimbodyBallJoint::~~SimbodyBallJoint** () [**virtual**]

Destructor.

10.153.3 Member Function Documentation

10.153.3.1 `math::Vector3 gazebo::physics::SimbodyBallJoint::GetAnchor (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. 807).

10.153.3.2 `virtual math::Angle gazebo::physics::SimbodyBallJoint::GetAngleImpl (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. 418).

10.153.3.3 `virtual math::Vector3 gazebo::physics::SimbodyBallJoint::GetAxis (int) const` [inline], [virtual]

10.153.3.4 `virtual math::Vector3 gazebo::physics::SimbodyBallJoint::GetGlobalAxis (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. 420).

10.153.3.5 `virtual double gazebo::physics::SimbodyBallJoint::GetMaxForce (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. 423).

10.153.3.6 virtual double gazebo::physics::SimbodyBallJoint::GetVelocity (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. 423).

10.153.3.7 virtual void gazebo::physics::SimbodyBallJoint::Init () [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Joint** (p. 424).

10.153.3.8 virtual void gazebo::physics::SimbodyBallJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load **physics::Joint** (p. 411) from a SDF sdf::Element.

Parameters

in	<i>_sdf</i>	SDF values to load from.
----	-------------	--------------------------

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 810).

10.153.3.9 virtual void gazebo::physics::SimbodyBallJoint::SetDamping (int *_index*, double *_damping*) [virtual]

Set joint damping, not yet implemented.

See Also

Joint::SetDamping (p. 426)

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 811).

10.153.3.10 `virtual void gazebo::physics::SimbodyBallJoint::SetForceImpl (int _index, double _force)` [protected],
[virtual]

Set the force applied to this **physics::Joint** (p. 411).

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. 411)).

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. 812).

10.153.3.11 `virtual void gazebo::physics::SimbodyBallJoint::SetHighStop (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::Joint** (p. 427).

10.153.3.12 `virtual void gazebo::physics::SimbodyBallJoint::SetLowStop (int _index, const math::Angle & _angle)`
[virtual]

Set the low stop of an axis(index).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_angle</i>	Low stop angle.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.153.3.13 `virtual void gazebo::physics::SimbodyBallJoint::SetMaxForce (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. 427).

10.153.3.14 `virtual void gazebo::physics::SimbodyBallJoint::SetVelocity (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. 428).

The documentation for this class was generated from the following file:

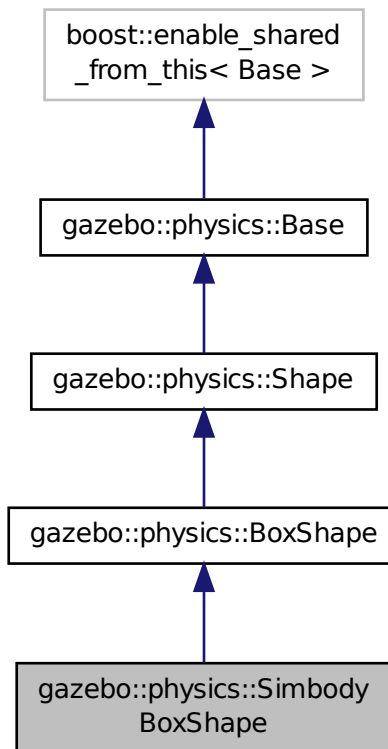
- `SimbodyBallJoint.hh`

10.154 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

10.154.1 Detailed Description

Simbody box collision.

10.154.2 Constructor & Destructor Documentation

10.154.2.1 `gazebo::physics::SimbodyBoxShape::SimbodyBoxShape (CollisionPtr _parent)` `[inline]`

Constructor.

10.154.2.2 `virtual gazebo::physics::SimbodyBoxShape::~~SimbodyBoxShape ()` `[inline],[virtual]`

Destructor.

10.154.3 Member Function Documentation

10.154.3.1 `void gazebo::physics::SimbodyBoxShape::SetSize (const math::Vector3 & _size)` `[inline],[virtual]`

Set the size of the box.

Parameters

in	<code>_size</code>	Size of each side of the box.
----	--------------------	-------------------------------

Reimplemented from `gazebo::physics::BoxShape` (p. 172).

References `gazebo::physics::Shape::collisionParent`, `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:

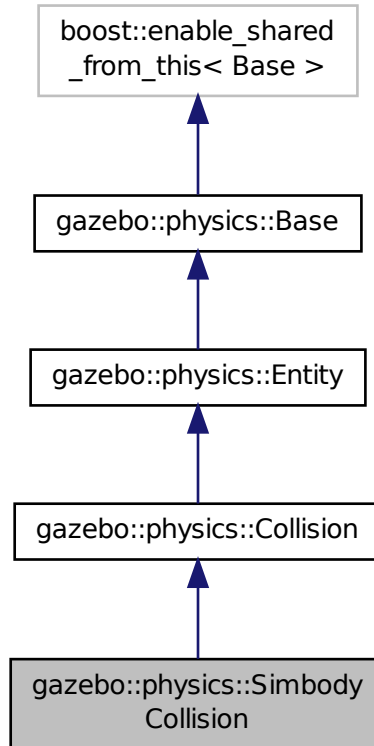
- **SimbodyBoxShape.hh**

10.155 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

10.155.1 Detailed Description

Simbody collisions.

10.155.2 Constructor & Destructor Documentation

10.155.2.1 gazebo::physics::SimbodyCollision::SimbodyCollision (LinkPtr _parent)

Constructor.

10.155.2.2 virtual gazebo::physics::SimbodyCollision::~~SimbodyCollision () [virtual]

Destructor.

10.155.3 Member Function Documentation

10.155.3.1 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. 216).

10.155.3.2 SimTK::ContactGeometry* gazebo::physics::SimbodyCollision::GetCollisionShape () const

Get the simbody collision shape.

Returns

SimTK (p. 123) geometry used as the collision shape.

10.155.3.3 virtual void gazebo::physics::SimbodyCollision::Load (sdf::ElementPtr _sdf) [virtual]

Load the collision.

Parameters

in	<i>_sdf</i>	SDF to load from.
----	-------------	-------------------

Reimplemented from **gazebo::physics::Collision** (p. 220).

10.155.3.4 `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. 301).

10.155.3.5 `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. 220).

10.155.3.6 `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. 220).

10.155.3.7 `void gazebo::physics::SimbodyCollision::SetCollisionShape (SimTK::ContactGeometry * _shape)`

Set the collision shape.

Parameters

<code>in</code>	<code><i>_shape</i></code>	SimTK (p. 123) geometry to use as the collision SimTK (p. 123) geometry to use as the collision shape.
-----------------	----------------------------	--

The documentation for this class was generated from the following file:

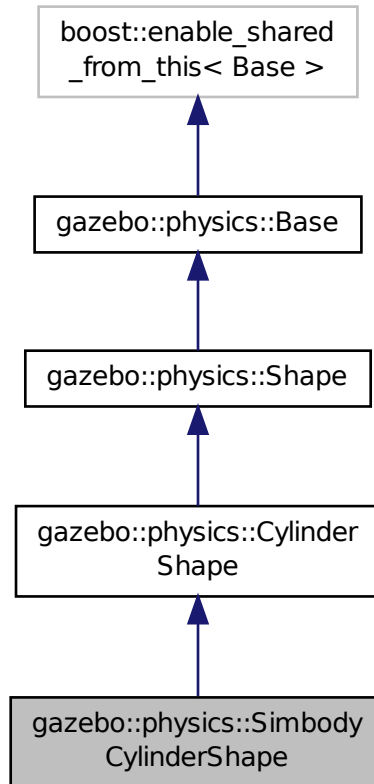
- **SimbodyCollision.hh**

10.156 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

10.156.1 Detailed Description

Cylinder collision.

10.156.2 Constructor & Destructor Documentation

10.156.2.1 `gazebo::physics::SimbodyCylinderShape::SimbodyCylinderShape (CollisionPtr _parent) [inline]`

Constructor.

10.156.2.2 `virtual gazebo::physics::SimbodyCylinderShape::~~SimbodyCylinderShape () [inline],[virtual]`

Destructor.

10.156.3 Member Function Documentation

10.156.3.1 `void gazebo::physics::SimbodyCylinderShape::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. 272).

References `gazebo::physics::Shape::collisionParent`, `gazebo::math::equal()`, `gzerr`, `gzwarn`, and `gazebo::physics::CylinderShape::SetSize()`.

The documentation for this class was generated from the following file:

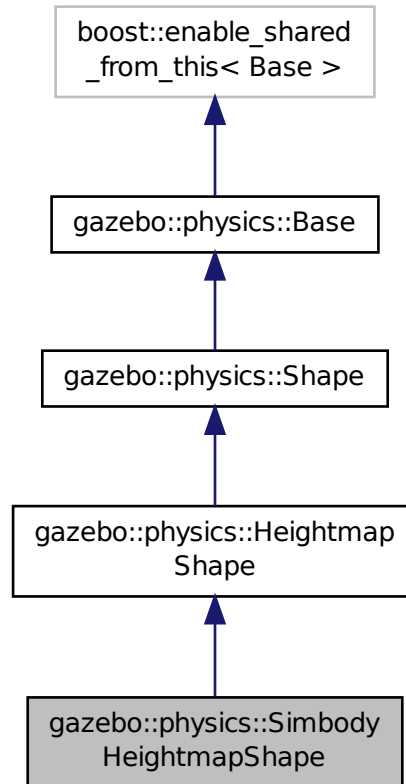
- `SimbodyCylinderShape.hh`

10.157 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

10.157.1 Detailed Description

Height map collision.

10.157.2 Constructor & Destructor Documentation

10.157.2.1 gazebo::physics::SimbodyHeightmapShape::SimbodyHeightmapShape (CollisionPtr _parent)

Constructor.

10.157.2.2 virtual gazebo::physics::SimbodyHeightmapShape::~~SimbodyHeightmapShape () [virtual]

Destructor.

10.157.3 Member Function Documentation

10.157.3.1 virtual void gazebo::physics::SimbodyHeightmapShape::Init () [virtual]

Initialize the heightmap.

Reimplemented from **gazebo::physics::HeightmapShape** (p. 384).

The documentation for this class was generated from the following file:

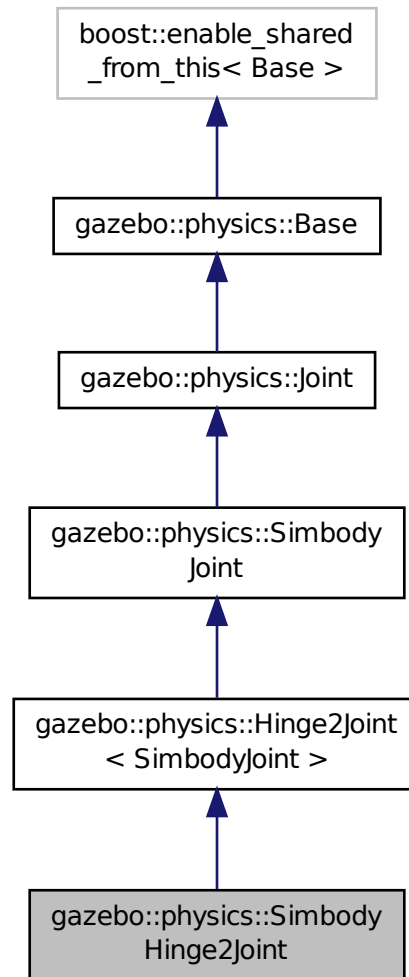
- **SimbodyHeightmapShape.hh**

10.158 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** (int _index) const
Get the anchor point.
- virtual **math::Vector3 GetAxis** (int _index) const
- virtual **math::Vector3 GetGlobalAxis** (int _index) const
Get the axis of rotation in global coordinate frame.

- virtual **math::Angle GetHighStop** (int _index)
Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int _index)
Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int _index) const
Get the rotation rate of an axis(index)
- virtual void **Init** ()
Initialize a joint.
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)
Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetDamping** (int _index, double _damping)
Set joint damping, not yet implemented.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)
Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)
Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _angle)
Set the velocity of an axis(index).

Protected Member Functions

- virtual **math::Angle GetAngleImpl** (int _index) const
Get the angle of an axis helper function.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the joint.
- virtual void **SetForceImpl** (int _index, double _torque)
Set the torque.

Additional Inherited Members

10.158.1 Detailed Description

A two axis hinge joint.

10.158.2 Constructor & Destructor Documentation

10.158.2.1 **gazebo::physics::SimbodyHinge2Joint::SimbodyHinge2Joint** (**SimTK::MultibodySystem** * *world*, **BasePtr** *_parent*)

Constructor.

10.158.2.2 **virtual gazebo::physics::SimbodyHinge2Joint::~~SimbodyHinge2Joint** () [virtual]

Destructor.

10.158.3 Member Function Documentation

10.158.3.1 `virtual math::Vector3 gazebo::physics::SimbodyHinge2Joint::GetAnchor (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. 807).

10.158.3.2 `virtual math::Angle gazebo::physics::SimbodyHinge2Joint::GetAngleImpl (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. 418).

10.158.3.3 `virtual math::Vector3 gazebo::physics::SimbodyHinge2Joint::GetAxis (int _index) const` [virtual]

10.158.3.4 `virtual math::Vector3 gazebo::physics::SimbodyHinge2Joint::GetGlobalAxis (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. 420).

10.158.3.5 `virtual math::Angle gazebo::physics::SimbodyHinge2Joint::GetHighStop (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	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Angle of the high stop value.

Implements **gazebo::physics::Joint** (p. 420).

10.158.3.6 `virtual math::Angle gazebo::physics::SimbodyHinge2Joint::GetLowStop (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	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Angle of the low stop value.

Implements **gazebo::physics::Joint** (p. 422).

10.158.3.7 `virtual double gazebo::physics::SimbodyHinge2Joint::GetMaxForce (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. 423).

10.158.3.8 `virtual double gazebo::physics::SimbodyHinge2Joint::GetVelocity (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. 423).

10.158.3.9 `virtual void gazebo::physics::SimbodyHinge2Joint::Init () [virtual]`

Initialize a joint.

Reimplemented from **gazebo::physics::Joint** (p. 424).

10.158.3.10 `virtual void gazebo::physics::SimbodyHinge2Joint::Load (sdf::ElementPtr _sdf) [protected],[virtual]`

Load the joint.

Parameters

in	_sdf	SDF values to load from.
----	------	--------------------------

Reimplemented from **gazebo::physics::Hinge2Joint< SimbodyJoint >** (p. 387).

10.158.3.11 `virtual void gazebo::physics::SimbodyHinge2Joint::SetAxis (int _index, const math::Vector3 & _axis) [virtual]`

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	_index	Index of the axis to set.
in	_axis	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 811).

10.158.3.12 `virtual void gazebo::physics::SimbodyHinge2Joint::SetDamping (int _index, double _damping) [virtual]`

Set joint damping, not yet implemented.

See Also

Hinge2Joint::SetDamping

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 811).

10.158.3.13 `virtual void gazebo::physics::SimbodyHinge2Joint::SetForceImpl (int _index, double _torque) [protected],[virtual]`

Set the torque.

Implements **gazebo::physics::SimbodyJoint** (p. 812).

10.158.3.14 `virtual void gazebo::physics::SimbodyHinge2Joint::SetHighStop (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::Joint` (p. 427).

10.158.3.15 `virtual void gazebo::physics::SimbodyHinge2Joint::SetLowStop (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::Joint` (p. 427).

10.158.3.16 `virtual void gazebo::physics::SimbodyHinge2Joint::SetMaxForce (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. 427).

10.158.3.17 `virtual void gazebo::physics::SimbodyHinge2Joint::SetVelocity (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. 428).

The documentation for this class was generated from the following file:

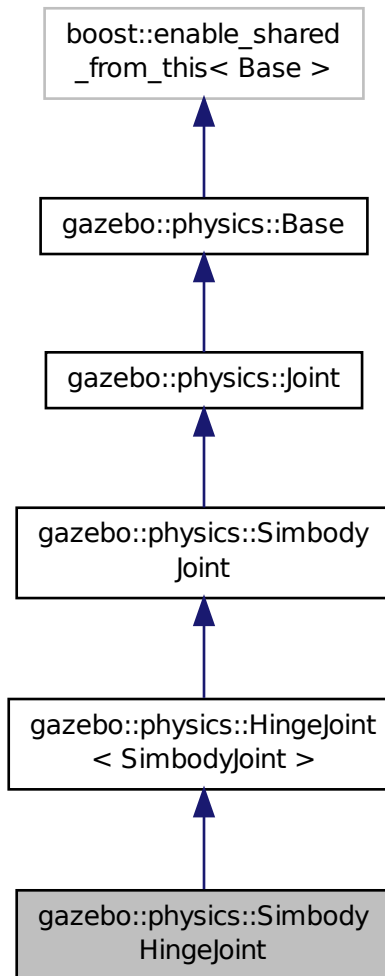
- `SimbodyHinge2Joint.hh`

10.159 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** (int _index) const
Get the axis of rotation in global coordinate frame.
- virtual **math::Angle GetHighStop** (int _index)
Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int _index)
Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (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** (int _index, const **math::Vector3** &_axis)
Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetDamping** (int _index, double _damping)
Set the joint damping.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)
Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)
Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _rate)
Set the velocity of an axis(index).

Protected Member Functions

- virtual **math::Angle GetAngleImpl** (int _index) const
Get the angle of an axis helper function.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load joint.
- virtual void **SetForceImpl** (int _index, double _torque)
*Set the force applied to this **physics::Joint** (p. 411).*

Additional Inherited Members

10.159.1 Detailed Description

A single axis hinge joint.

10.159.2 Constructor & Destructor Documentation

10.159.2.1 gazebo::physics::SimbodyHingeJoint::SimbodyHingeJoint (SimTK::MultibodySystem * world, BasePtr _parent)

Constructor.

10.159.2.2 virtual gazebo::physics::SimbodyHingeJoint::~~SimbodyHingeJoint () [virtual]

Destructor.

10.159.3 Member Function Documentation

10.159.3.1 virtual math::Angle gazebo::physics::SimbodyHingeJoint::GetAngleImpl (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. 418).

10.159.3.2 virtual math::Vector3 gazebo::physics::SimbodyHingeJoint::GetGlobalAxis (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. 420).

10.159.3.3 virtual math::Angle gazebo::physics::SimbodyHingeJoint::GetHighStop (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	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Angle of the high stop value.

Implements gazebo::physics::Joint (p. 420).

10.159.3.4 `virtual math::Angle gazebo::physics::SimbodyHingeJoint::GetLowStop (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.

Implements `gazebo::physics::Joint` (p. 422).

10.159.3.5 `virtual double gazebo::physics::SimbodyHingeJoint::GetMaxForce (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. 423).

10.159.3.6 `virtual double gazebo::physics::SimbodyHingeJoint::GetVelocity (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. 423).

10.159.3.7 `virtual void gazebo::physics::SimbodyHingeJoint::Load (sdf::ElementPtr _sdf) [protected],[virtual]`

Load joint.

Parameters

in	<code>_sdf</code>	Pointer to SDF element
----	-------------------	------------------------

Reimplemented from `gazebo::physics::HingeJoint` < `SimbodyJoint` > (p. 389).

10.159.3.8 virtual void `gazebo::physics::SimbodyHingeJoint::RestoreSimbodyState (SimTK::State & _state)` [virtual]

restore simbody state for spawning

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 810).

10.159.3.9 virtual void `gazebo::physics::SimbodyHingeJoint::SaveSimbodyState (const SimTK::State & _state)` [virtual]

save simbody state for spawning

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 810).

10.159.3.10 void `gazebo::physics::SimbodyHingeJoint::SetAxis (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. 811).

10.159.3.11 virtual void `gazebo::physics::SimbodyHingeJoint::SetDamping (int _index, double _damping)` [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.

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 811).

10.159.3.12 virtual void `gazebo::physics::SimbodyHingeJoint::SetForceImpl (int _index, double _force)` [protected],
[virtual]

Set the force applied to this `physics::Joint` (p. 411).

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. 411)).

Parameters

in	<code>_index</code>	Index of the axis.
in	<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. 812).

10.159.3.13 `virtual void gazebo::physics::SimbodyHingeJoint::SetHighStop (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::Joint** (p. 427).

10.159.3.14 `virtual void gazebo::physics::SimbodyHingeJoint::SetLowStop (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::Joint** (p. 427).

10.159.3.15 `virtual void gazebo::physics::SimbodyHingeJoint::SetMaxForce (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. 427).

10.159.3.16 `virtual void gazebo::physics::SimbodyHingeJoint::SetVelocity (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. 428).

The documentation for this class was generated from the following file:

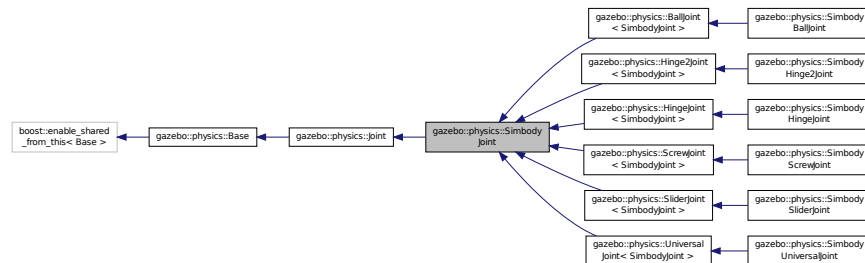
- **SimbodyHingeJoint.hh**

10.160 gazebo::physics::SimbodyJoint Class Reference

Base (p. 153) 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. 411) Force Torque Values if necessary for physics engine.*
- virtual void **Detach** ()
Detach this joint from all links.
- virtual **math::Vector3** **GetAnchor** (int _index) const
Get the anchor point.
- virtual double **GetAttribute** (const std::string &_key, unsigned int _index)
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 **LinkPtr** **GetJointLink** (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. 455) due to the existence of this **Joint** (p. 411).*
- virtual **math::Vector3** **GetLinkTorque** (unsigned int _index) const
*Get the torque applied to the center of mass of a **physics::Link** (p. 455) due to the existence of this **Joint** (p. 411).*
- virtual void **Load** (sdf::ElementPtr _sdf)
*Load **physics::Joint** (p. 411) 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** (int _index, const gazebo::math::Vector3 &_anchor)
Set the anchor point.
- virtual void **SetAttribute** (**Attribute**, int _index, double _value)
Set a parameter for the joint.
- virtual void **SetAttribute** (const std::string &_key, int _index, const boost::any &_value)
Set a non-generic parameter for the joint.
- virtual void **SetAxis** (int _index, const math::Vector3 &_axis)
Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetDamping** (int _index, const double _damping)
Set the joint damping.
- virtual void **SetForce** (int _index, double _force)
*Set the force applied to this **physics::Joint** (p. 411).*

Public Attributes

- SimTK::Constraint **constraint**
: isValid() if we used a constraint to model this joint.
- SimTK::Force::MobilityLinearDamper **damper**
: 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**
: 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::Transform **xCB**
child body frame to mobilizer frame
- SimTK::Transform **xPA**
Normally $A=F$, $B=M$.

Protected Member Functions

- virtual void **SetForceImpl** (int _index, double _force)=0
*Set the force applied to this **physics::Joint** (p. 411).*

Protected Attributes

- **SimbodyPhysicsPtr simbodyPhysics**
keep a pointer to the simbody physics engine for convenience
- SimTK::MultibodySystem * **world**
Simbody Multibody System.

Additional Inherited Members

10.160.1 Detailed Description

Base (p. 153) class for all joints.

10.160.2 Constructor & Destructor Documentation

10.160.2.1 gazebo::physics::SimbodyJoint::SimbodyJoint (BasePtr *_parent*)

Constructor.

10.160.2.2 virtual gazebo::physics::SimbodyJoint::~~SimbodyJoint () [virtual]

Destructor.

10.160.3 Member Function Documentation

10.160.3.1 virtual bool gazebo::physics::SimbodyJoint::AreConnected (LinkPtr *_one*, LinkPtr *_two*) const [virtual]

Determines if the two bodies are connected by a joint.

Parameters

<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. 415).

10.160.3.2 virtual void gazebo::physics::SimbodyJoint::CacheForceTorque () [virtual]

Cache **Joint** (p. 411) Force Torque Values if necessary for physics engine.

Reimplemented from **gazebo::physics::Joint** (p. 416).

10.160.3.3 virtual void gazebo::physics::SimbodyJoint::Detach () [virtual]

Detach this joint from all links.

Reimplemented from **gazebo::physics::Joint** (p. 417).

10.160.3.4 virtual math::Vector3 gazebo::physics::SimbodyJoint::GetAnchor (int *_index*) const [virtual]

Get the anchor point.

Parameters

in	<code>_index</code>	Index of the axis.
----	---------------------	--------------------

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 417).

Reimplemented in **gazebo::physics::ScrewJoint**< **SimbodyJoint** > (p. 747), **gazebo::physics::SliderJoint**< **SimbodyJoint** > (p. 887), **gazebo::physics::SimbodyHinge2Joint** (p. 795), **gazebo::physics::SimbodyUniversalJoint** (p. 860), and **gazebo::physics::SimbodyBallJoint** (p. 781).

10.160.3.5 `virtual double gazebo::physics::SimbodyJoint::GetAttribute (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.
in	<code>_value</code>	Value of the attribute.

Implements **gazebo::physics::Joint** (p. 418).

10.160.3.6 `virtual double gazebo::physics::SimbodyJoint::GetForce (unsigned int _index)` [virtual]

Todo : not yet implemented. Get external forces applied at this **Joint** (p. 411). 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 force applied to an axis.

Reimplemented from **gazebo::physics::Joint** (p. 419).

10.160.3.7 `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. 442) data structure. Where **JointWrench.body1Force** (p. 444) contains the force applied by the parent **Link** (p. 455) on the **Joint** (p. 411) specified in the parent **Link** (p. 455) frame, and **JointWrench.body2Force** (p. 444) contains the force applied by the child **Link** (p. 455) on the **Joint** (p. 411) specified in the child **Link** (p. 455) frame. Note that this sign convention is opposite of the reaction forces of the **Joint** (p. 411) 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. 455) 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>_index</code>	Not used right now
-----------------	---------------------	--------------------

Returns

The force and torque at the joint, see above for details on conventions.

Implements **gazebo::physics::Joint** (p. 420).

10.160.3.8 `virtual LinkPtr gazebo::physics::SimbodyJoint::GetJointLink (int _index) const` [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.

Implements **gazebo::physics::Joint** (p. 421).

10.160.3.9 `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. 455) due to the existence of this **Joint** (p. 411).

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.

Implements **gazebo::physics::Joint** (p. 421).

10.160.3.10 `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. 455) due to the existence of this **Joint** (p. 411).

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. 422).

10.160.3.11 `virtual void gazebo::physics::SimbodyJoint::Load (sdf::ElementPtr _sdf) [virtual]`

Load **physics::Joint** (p. 411) from a SDF `sdf::Element`.

Parameters

<code>in</code>	<code>_sdf</code>	SDF values to load from.
-----------------	-------------------	--------------------------

Reimplemented from **gazebo::physics::Joint** (p. 424).

Reimplemented in **gazebo::physics::SimbodySliderJoint** (p. 853), **gazebo::physics::Hinge2Joint< SimbodyJoint >** (p. 387), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 152), **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 748), **gazebo::physics::UniversalJoint< SimbodyJoint >** (p. 978), **gazebo::physics::HingeJoint< SimbodyJoint >** (p. 389), **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 888), **gazebo::physics::SimbodyHingeJoint** (p. 802), **gazebo::physics::SimbodyHinge2Joint** (p. 797), **gazebo::physics::SimbodyUniversalJoint** (p. 862), **gazebo::physics::SimbodyScrewJoint** (p. 847), and **gazebo::physics::SimbodyBallJoint** (p. 782).

10.160.3.12 `virtual void gazebo::physics::SimbodyJoint::Reset () [virtual]`

Reset the joint.

Reimplemented from **gazebo::physics::Joint** (p. 425).

10.160.3.13 `virtual void gazebo::physics::SimbodyJoint::RestoreSimbodyState (SimTK::State & _state) [virtual]`

Reimplemented in **gazebo::physics::SimbodyHingeJoint** (p. 803).

10.160.3.14 `virtual void gazebo::physics::SimbodyJoint::SaveSimbodyState (const SimTK::State & _state) [virtual]`

Reimplemented in **gazebo::physics::SimbodyHingeJoint** (p. 803).

10.160.3.15 `virtual void gazebo::physics::SimbodyJoint::SetAnchor (int _index, const gazebo::math::Vector3 & _anchor) [virtual]`

Set the anchor point.

Parameters

<code>in</code>	<code>_index</code>	Indx of the axis.
<code>in</code>	<code>_anchor</code>	Anchor value.

Implements **gazebo::physics::Joint** (p. 425).

Reimplemented in **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 748), and **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 888).

10.160.3.16 `virtual void gazebo::physics::SimbodyJoint::SetAttribute (Attribute , int _index, double _value) [virtual]`

Set a parameter for the joint.

10.160.3.17 `virtual void gazebo::physics::SimbodyJoint::SetAttribute (const std::string & _key, int _index, const boost::any & _value) [virtual]`

Set a non-generic parameter for the joint.

replaces **SetAttribute(Attribute, int, double)** (p. 811)

Parameters

<i>in</i>	<i>_key</i>	String key.
<i>in</i>	<i>_index</i>	Index of the axis.
<i>in</i>	<i>_value</i>	Value of the attribute.

Implements **gazebo::physics::Joint** (p. 425).

10.160.3.18 `virtual void gazebo::physics::SimbodyJoint::SetAxis (int _index, const math::Vector3 & _axis) [virtual]`

Set the axis of rotation where axis is specified in local joint frame.

Parameters

<i>in</i>	<i>_index</i>	Index of the axis to set.
<i>in</i>	<i>_axis</i>	Vector in local joint frame of axis direction (must have length greater than zero).

Implements **gazebo::physics::Joint** (p. 426).

Reimplemented in **gazebo::physics::BallJoint**< **SimbodyJoint** > (p. 153), **gazebo::physics::SimbodyHinge2Joint** (p. 797), **gazebo::physics::SimbodyUniversalJoint** (p. 862), **gazebo::physics::SimbodyScrewJoint** (p. 847), **gazebo::physics::SimbodyHingeJoint** (p. 803), and **gazebo::physics::SimbodySliderJoint** (p. 854).

10.160.3.19 `virtual void gazebo::physics::SimbodyJoint::SetDamping (int _index, const double _damping) [virtual]`

Set the joint damping.

Parameters

<i>in</i>	<i>_index</i>	Index of the axis to set, currently ignored, to be implemented.
<i>in</i>	<i>_damping</i>	Damping value for the axis.

Implements **gazebo::physics::Joint** (p. 426).

Reimplemented in **gazebo::physics::SimbodySliderJoint** (p. 854), **gazebo::physics::SimbodyUniversalJoint** (p. 862), **gazebo::physics::SimbodyHinge2Joint** (p. 797), **gazebo::physics::SimbodyHingeJoint** (p. 803), **gazebo::physics::SimbodyScrewJoint** (p. 847), and **gazebo::physics::SimbodyBallJoint** (p. 782).

10.160.3.20 `virtual void gazebo::physics::SimbodyJoint::SetForce (int _index, double _effort) [virtual]`

Set the force applied to this **physics::Joint** (p. 411).

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. 411)). 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. 426).

```
10.160.3.21 virtual void gazebo::physics::SimbodyJoint::SetForceImpl ( int _index, double _force ) [protected], [pure virtual]
```

Set the force applied to this **physics::Joint** (p. 411).

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. 411)).

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::SimbodyHinge2Joint** (p. 797), **gazebo::physics::SimbodyUniversalJoint** (p. 863), **gazebo::physics::SimbodyScrewJoint** (p. 848), **gazebo::physics::SimbodyHingeJoint** (p. 803), **gazebo::physics::SimbodySliderJoint** (p. 854), and **gazebo::physics::SimbodyBallJoint** (p. 783).

10.160.4 Member Data Documentation

10.160.4.1 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.

10.160.4.2 SimTK::Force::MobilityLinearDamper gazebo::physics::SimbodyJoint::damper

: for enforcing joint damping forces.

Set when we build the Simbody model. : Make these arrays for multi-axis joints. : Also, consider moving this into individual joint type subclass so we can specify custom dampers for special joints like ball joints.

10.160.4.3 SimTK::Transform gazebo::physics::SimbodyJoint::defxAB

default mobilizer pose

10.160.4.4 bool gazebo::physics::SimbodyJoint::isReversed

: if mobilizer, did it reverse parent&child? Set when we build the Simbody model.

10.160.4.5 SimTK::Force::MobilityLinearStop gazebo::physics::SimbodyJoint::limitForce

: for enforcing joint stops Set when we build the Simbody model.

: Make these arrays for multi-axis joints. : Also, consider moving this into individual joint type subclass so we can specify custom dampers for special joints like ball joints.

10.160.4.6 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.

10.160.4.7 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.

10.160.4.8 bool gazebo::physics::SimbodyJoint::physicsInitialized

10.160.4.9 SimbodyPhysicsPtr gazebo::physics::SimbodyJoint::simbodyPhysics [protected]

keep a pointer to the simbody physics engine for convenience

10.160.4.10 SimTK::MultibodySystem* gazebo::physics::SimbodyJoint::world [protected]

Simbody Multibody System.

10.160.4.11 SimTK::Transform gazebo::physics::SimbodyJoint::xCB

child body frame to mobilizer frame

10.160.4.12 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:

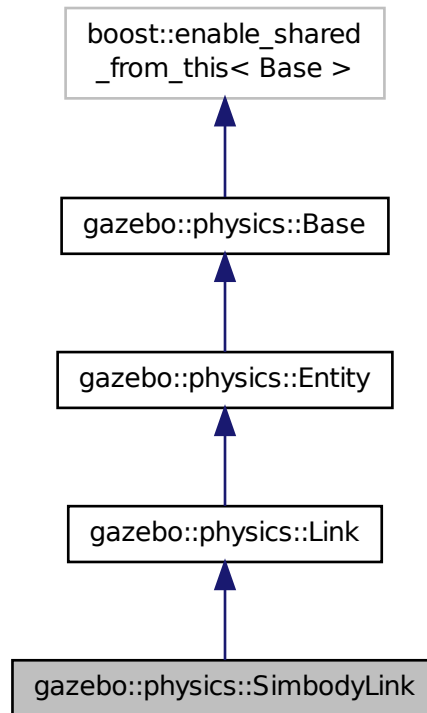
- **SimbodyJoint.hh**

10.161 gazebo::physics::SimbodyLink Class Reference

Simbody **Link** (p. 455) class.

```
#include <SimbodyLink.hh>
```

Inheritance diagram for gazebo::physics::SimbodyLink:



Public Member Functions

- **SimbodyLink** (EntityPtr _parent)
Constructor.
- virtual \sim **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

10.161.1 Detailed Description

Simbody **Link** (p. 455) class.

10.161.2 Constructor & Destructor Documentation

10.161.2.1 gazebo::physics::SimbodyLink::SimbodyLink (EntityPtr *_parent*)

Constructor.

10.161.2.2 virtual gazebo::physics::SimbodyLink::~~SimbodyLink () [virtual]

Destructor.

10.161.3 Member Function Documentation

10.161.3.1 virtual void gazebo::physics::SimbodyLink::AddForce (const **math::Vector3** & *_force*) [virtual]

Add a force to the body.

Parameters

<i>in</i>	<i>_force</i>	Force to add.
-----------	---------------	---------------

Implements **gazebo::physics::Link** (p. 460).

10.161.3.2 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. 461).

10.161.3.3 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. 461).

10.161.3.4 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. 461).

10.161.3.5 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. 461).

10.161.3.6 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. 462).

10.161.3.7 `virtual void gazebo::physics::SimbodyLink::Fini() [virtual]`

Finalize the body.

Reimplemented from **gazebo::physics::Link** (p. 463).

10.161.3.8 `SimTK::MassProperties gazebo::physics::SimbodyLink::GetEffectiveMassProps(int _numFragments) const`

10.161.3.9 `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. 464).

10.161.3.10 `virtual bool gazebo::physics::SimbodyLink::GetGravityMode() const [virtual]`

Get the gravity mode.

Returns

True if gravity is enabled.

Implements **gazebo::physics::Link** (p. 464).

10.161.3.11 `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.

10.161.3.12 `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. 1004) for the velocity.

Reimplemented from **gazebo::physics::Entity** (p. 300).

10.161.3.13 `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. 468).

10.161.3.14 `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. 468).

10.161.3.15 `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. 455) frame will be returned.

Parameters

<code>in</code>	<code>_offset</code>	Offset of the point from the origin of the Link (p. 455) frame, expressed in the body-fixed frame.
-----------------	----------------------	---

Returns

Linear velocity of the point on the body

Implements **gazebo::physics::Link** (p. 468).

10.161.3.16 `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. 469).

10.161.3.17 `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. 469).

10.161.3.18 `virtual void gazebo::physics::SimbodyLink::Init () [virtual]`

Initialize the body.

Reimplemented from **gazebo::physics::Link** (p. 469).

10.161.3.19 `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. 469).

10.161.3.20 `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. 470).

10.161.3.21 `virtual void gazebo::physics::SimbodyLink::RestoreSimbodyState (SimTK::State & _state) [virtual]`

10.161.3.22 `virtual void gazebo::physics::SimbodyLink::SaveSimbodyState (const SimTK::State & _state) [virtual]`

10.161.3.23 `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. 471).

10.161.3.24 `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. 471).

10.161.3.25 `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. 471).

10.161.3.26 `void gazebo::physics::SimbodyLink::SetDirtyPose (const math::Pose & _pose)`

10.161.3.27 `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. 472).

10.161.3.28 `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. 472).

10.161.3.29 `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. 472).

10.161.3.30 `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. 473).

10.161.3.31 `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. 473).

10.161.3.32 `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. 473).

10.161.3.33 `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. 474).

10.161.3.34 `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. 474).

10.161.4 Member Data Documentation

10.161.4.1 `SimTK::MobilizedBody gazebo::physics::SimbodyLink::masterMobod`

10.161.4.2 `bool gazebo::physics::SimbodyLink::mustBeBaseLink`

: Force this link to be a base body, where its inboard body is the world with 6DOF.

10.161.4.3 `bool gazebo::physics::SimbodyLink::physicsInitialized`

10.161.4.4 `std::vector<SimTK::MobilizedBody> gazebo::physics::SimbodyLink::slaveMobods`

10.161.4.5 `std::vector<SimTK::Constraint::Weld>` gazebo::physics::SimbodyLink::slaveWelds

The documentation for this class was generated from the following file:

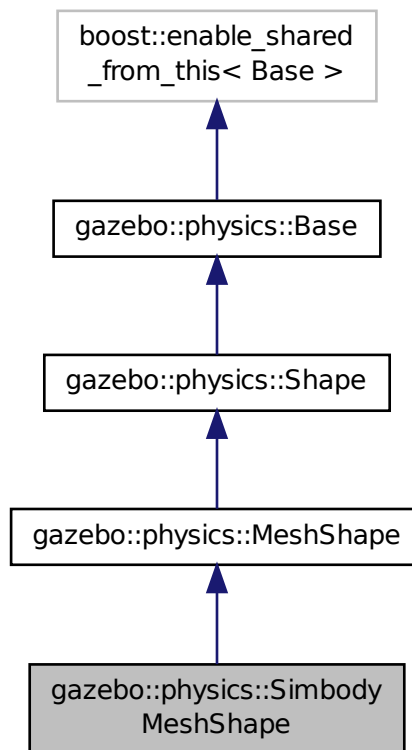
- **SimbodyLink.hh**

10.162 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

10.162.1 Detailed Description

Triangle mesh collision.

10.162.2 Constructor & Destructor Documentation

10.162.2.1 gazebo::physics::SimbodyMeshShape::SimbodyMeshShape (CollisionPtr _parent)

Constructor.

10.162.2.2 virtual gazebo::physics::SimbodyMeshShape::~~SimbodyMeshShape () [virtual]

Destructor.

10.162.3 Member Function Documentation

10.162.3.1 virtual void gazebo::physics::SimbodyMeshShape::Init () [protected],[virtual]

Initialize the shape.

Reimplemented from **gazebo::physics::MeshShape** (p. 536).

10.162.3.2 virtual void gazebo::physics::SimbodyMeshShape::Load (sdf::ElementPtr _sdf) [virtual]

Load.

Parameters

<i>in</i>	<i>node</i>	Pointer to an SDF parameters
-----------	-------------	------------------------------

Reimplemented from **gazebo::physics::Base** (p. 161).

The documentation for this class was generated from the following file:

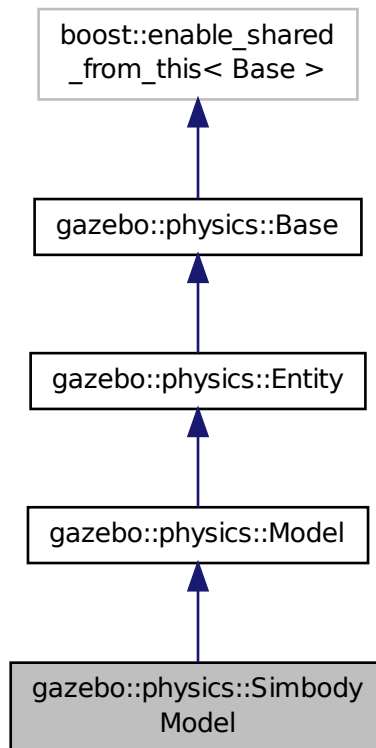
- **SimbodyMeshShape.hh**

10.163 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

10.163.1 Detailed Description

A model is a collection of links, joints, and plugins.

10.163.2 Constructor & Destructor Documentation

10.163.2.1 gazebo::physics::SimbodyModel::SimbodyModel (BasePtr *_parent*) [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent object.
----	----------------	----------------

10.163.2.2 virtual gazebo::physics::SimbodyModel::~~SimbodyModel () [virtual]

Destructor.

10.163.3 Member Function Documentation

10.163.3.1 virtual void gazebo::physics::SimbodyModel::Init () [virtual]

Initialize the model.

Reimplemented from **gazebo::physics::Model** (p. 546).

10.163.3.2 virtual void gazebo::physics::SimbodyModel::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the model.

Parameters

in	<i>_sdf</i>	SDF parameters to load from.
----	-------------	------------------------------

Reimplemented from **gazebo::physics::Model** (p. 546).

The documentation for this class was generated from the following file:

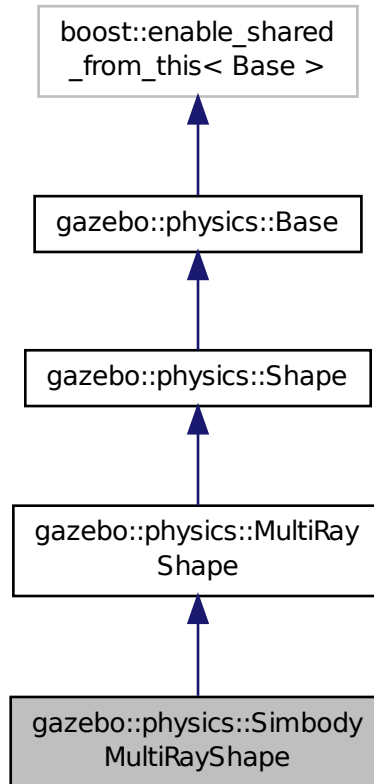
- **SimbodyModel.hh**

10.164 gazebo::physics::SimbodyMultiRayShape Class Reference

Simbody specific version of **MultiRayShape** (p. 578).

```
#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

10.164.1 Detailed Description

Simbody specific version of **MultiRayShape** (p. 578).

10.164.2 Constructor & Destructor Documentation

10.164.2.1 gazebo::physics::SimbodyMultiRayShape::SimbodyMultiRayShape (CollisionPtr parent)

Constructor.

10.164.2.2 virtual gazebo::physics::SimbodyMultiRayShape::~~SimbodyMultiRayShape () [virtual]

Destructor.

10.164.3 Member Function Documentation

10.164.3.1 virtual void gazebo::physics::SimbodyMultiRayShape::AddRay (const math::Vector3 & _start, const math::Vector3 & _end) [protected],[virtual]

Add a ray to the collision.

Parameters

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

Reimplemented from **gazebo::physics::MultiRayShape** (p. 581).

10.164.3.2 virtual void gazebo::physics::SimbodyMultiRayShape::UpdateRays () [virtual]

Physics engine specific method for updating the rays.

Implements **gazebo::physics::MultiRayShape** (p. 586).

The documentation for this class was generated from the following file:

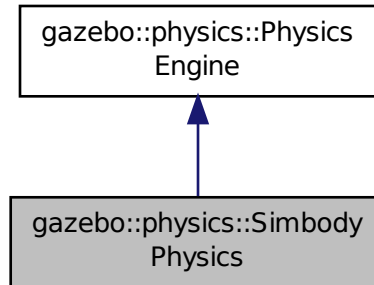
- **SimbodyMultiRayShape.hh**

10.165 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 **~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. 775) 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 std::string **GetType** () const
 - Return the type of the physics engine (ode|bullet|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. 537) 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 gravity vector.*
- 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. 160) to string, this is needed by the MultibodyGraphMaker.*
- static std::string **GetTypeString** (physics::Base::EntityType _type)
 - Convert **Base::GetType()** (p. 160) 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. 675) to SimTK::Quaternion.*
- static math::Quaternion **QuadToQuad** (const SimTK::Quaternion &_q)
 - Convert SimTK::Quaternion to **gazebo::math::Quaternion** (p. 675).*
- 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. 1004).*
- static SimTK::Vec3 **Vector3ToVec3** (const math::Vector3 &_v)
 - Convert **gazebo::math::Vector3** (p. 1004) 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

10.165.1 Detailed Description

Simbody physics engine.

10.165.2 Constructor & Destructor Documentation

10.165.2.1 gazebo::physics::SimbodyPhysics::SimbodyPhysics (WorldPtr _world)

Constructor.

10.165.2.2 virtual gazebo::physics::SimbodyPhysics::~~SimbodyPhysics () [virtual]

Destructor.

10.165.3 Member Function Documentation

10.165.3.1 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. 624).

10.165.3.2 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. 537) parent.

Implements **gazebo::physics::PhysicsEngine** (p. 625).

10.165.3.3 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. 625).

10.165.3.4 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. 625).

10.165.3.5 virtual `ShapePtr gazebo::physics::SimbodyPhysics::CreateShape (const std::string & _shapeType, CollisionPtr _collision)` [virtual]

Create a `physics::Shape` (p. 775) object.

Parameters

in	<code>_shapeType</code>	Type of shape to create.
in	<code>_collision</code>	<code>Collision</code> (p. 213) parent.

Implements `gazebo::physics::PhysicsEngine` (p. 625).

10.165.3.6 virtual void `gazebo::physics::SimbodyPhysics::DebugPrint () const` [virtual]

Debug print out of the physic engine state.

Implements `gazebo::physics::PhysicsEngine` (p. 626).

10.165.3.7 virtual void `gazebo::physics::SimbodyPhysics::Fini ()` [virtual]

Finilize the physics engine.

Reimplemented from `gazebo::physics::PhysicsEngine` (p. 626).

10.165.3.8 `SimTK::MultibodySystem* gazebo::physics::SimbodyPhysics::GetDynamicsWorld () const`

Register a joint with the dynamics world.

10.165.3.9 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.

10.165.3.10 `virtual std::string gazebo::physics::SimbodyPhysics::GetType () const [virtual]`

Return the type of the physics engine (ode|bullet|simbody).

Returns

Type of the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 628).

10.165.3.11 `static std::string gazebo::physics::SimbodyPhysics::GetTypeString (unsigned int _type) [static]`

Convert **Base::GetType()** (p. 160) to string, this is needed by the MultibodyGraphMaker.

Parameters

in	_type	Joint (p. 411) type returned by Joint::GetType() (p. 160).
----	-------	--

Returns

a hard-coded string needed by the MultibodyGraphMaker.

10.165.3.12 `static std::string gazebo::physics::SimbodyPhysics::GetTypeString (physics::Base::EntityType _type) [static]`

Convert **Base::GetType()** (p. 160) to string, this is needed by the MultibodyGraphMaker.

Parameters

in	_type	Joint (p. 411) type returned by Joint::GetType() (p. 160).
----	-------	--

Returns

a hard-coded string needed by the MultibodyGraphMaker.

10.165.3.13 `virtual void gazebo::physics::SimbodyPhysics::Init () [virtual]`

Initialize the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 629).

10.165.3.14 `virtual void gazebo::physics::SimbodyPhysics::InitForThread () [virtual]`

Init the engine for threads.

Implements **gazebo::physics::PhysicsEngine** (p. 629).

10.165.3.15 `void gazebo::physics::SimbodyPhysics::InitModel (const physics::ModelPtr _model)`

Add a **Model** (p. 537) to the Simbody system.

Parameters

in	<code>_model</code>	Pointer to the model to add into Simbody.
----	---------------------	---

10.165.3.16 `virtual void gazebo::physics::SimbodyPhysics::Load (sdf::ElementPtr _sdf) [virtual]`

Load the physics engine.

Parameters

in	<code>_sdf</code>	Pointer to the SDF parameters.
----	-------------------	--------------------------------

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 629).

10.165.3.17 `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. 630).

10.165.3.18 `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. 630).

10.165.3.19 `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. 648) object
----	--------------------	--

Returns

Simbody's SimTK::Transform object

10.165.3.20 `static SimTK::Quaternion gazebo::physics::SimbodyPhysics::QuadToQuad (const math::Quaternion & _q)`
`[static]`

Convert **gazebo::math::Quaternion** (p. 675) to SimTK::Quaternion.

Parameters

in	_q	Gazeb's math::Quaternion (p. 675) object
----	----	---

Returns

Simbody's SimTK::Quaternion object

10.165.3.21 `static math::Quaternion gazebo::physics::SimbodyPhysics::QuadToQuad (const SimTK::Quaternion & _q)`
`[static]`

Convert SimTK::Quaternion to **gazebo::math::Quaternion** (p. 675).

Parameters

in	_q	Simbody's SimTK::Quaternion object
----	----	------------------------------------

Returns

Gazeb's **math::Quaternion** (p. 675) object

10.165.3.22 `virtual void gazebo::physics::SimbodyPhysics::Reset ()` `[virtual]`

Rest the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 630).

10.165.3.23 `virtual void gazebo::physics::SimbodyPhysics::SetGravity (const gazebo::math::Vector3 & _gravity)`
`[virtual]`

Set the gavity vector.

Parameters

in	_gravity	New gravity vector.
----	----------	---------------------

Implements **gazebo::physics::PhysicsEngine** (p. 631).

10.165.3.24 `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. 632).

10.165.3.25 `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. 648) object

10.165.3.26 `virtual void gazebo::physics::SimbodyPhysics::UpdateCollision ()` [virtual]

Update the physics engine collision.

Implements **gazebo::physics::PhysicsEngine** (p. 633).

10.165.3.27 `virtual void gazebo::physics::SimbodyPhysics::UpdatePhysics ()` [virtual]

Update the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 633).

10.165.3.28 `static math::Vector3 gazebo::physics::SimbodyPhysics::Vec3ToVector3 (const SimTK::Vec3 & _v)` [static]

Convert SimTK::Vec3 to **gazebo::math::Vector3** (p. 1004).

Parameters

in	_v	Simbody's SimTK::Vec3 object
----	----	------------------------------

Returns

Gazebo's **math::Vector3** (p. 1004) object

10.165.3.29 `static SimTK::Vec3 gazebo::physics::SimbodyPhysics::Vector3ToVec3 (const math::Vector3 & _v)` [static]

Convert **gazebo::math::Vector3** (p. 1004) to SimTK::Vec3.

Parameters

in	_v	Gazebo's math::Vector3 (p. 1004) object
----	----	--

Returns

Simbody's SimTK::Vec3 object

10.165.4 Member Data Documentation

10.165.4.1 SimTK::CompliantContactSubsystem gazebo::physics::SimbodyPhysics::contact

10.165.4.2 SimTK::Force::DiscreteForces gazebo::physics::SimbodyPhysics::discreteForces

10.165.4.3 SimTK::GeneralForceSubsystem gazebo::physics::SimbodyPhysics::forces

10.165.4.4 SimTK::Force::Gravity gazebo::physics::SimbodyPhysics::gravity

10.165.4.5 SimTK:: Integrator* gazebo::physics::SimbodyPhysics::integ

10.165.4.6 SimTK::SimbodyMatterSubsystem gazebo::physics::SimbodyPhysics::matter

10.165.4.7 bool gazebo::physics::SimbodyPhysics::simbodyPhysicsInitialized

true if initialized

10.165.4.8 bool gazebo::physics::SimbodyPhysics::simbodyPhysicsStepped

10.165.4.9 SimTK::MultibodySystem gazebo::physics::SimbodyPhysics::system

10.165.4.10 SimTK::ContactTrackerSubsystem gazebo::physics::SimbodyPhysics::tracker

The documentation for this class was generated from the following file:

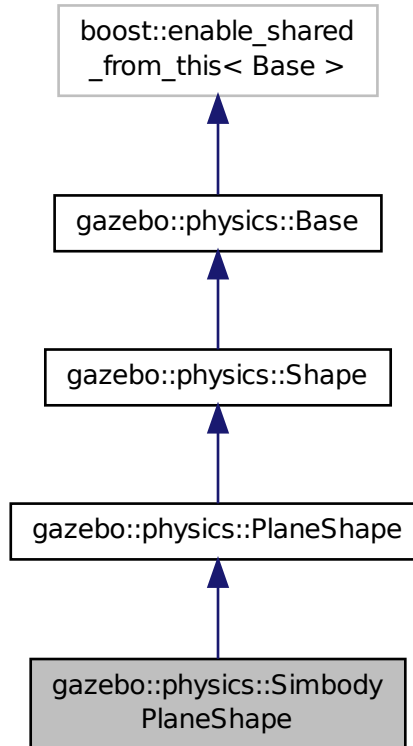
- **SimbodyPhysics.hh**

10.166 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

10.166.1 Detailed Description

Simbody collision for an infinite plane.

10.166.2 Constructor & Destructor Documentation

10.166.2.1 gazebo::physics::SimbodyPlaneShape::SimbodyPlaneShape (CollisionPtr *_parent*)

Constructor.

10.166.2.2 virtual gazebo::physics::SimbodyPlaneShape::~~SimbodyPlaneShape () [virtual]

Destructor.

10.166.3 Member Function Documentation

10.166.3.1 virtual void gazebo::physics::SimbodyPlaneShape::CreatePlane () [virtual]

Create the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 644).

10.166.3.2 virtual void gazebo::physics::SimbodyPlaneShape::SetAltitude (const math::Vector3 & *_pos*) [virtual]

Set the altitude of the plane.

Parameters

in	<i>_pos</i>	Position of the plane.
----	-------------	------------------------

Reimplemented from **gazebo::physics::PlaneShape** (p. 645).

The documentation for this class was generated from the following file:

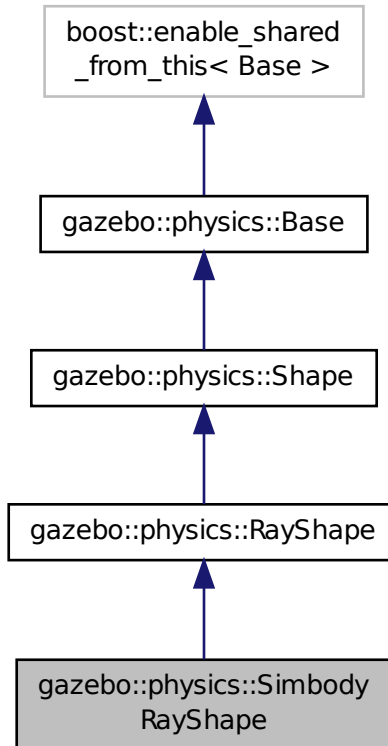
- **SimbodyPlaneShape.hh**

10.167 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

10.167.1 Detailed Description

Ray shape for simbody.

10.167.2 Constructor & Destructor Documentation

10.167.2.1 gazebo::physics::SimbodyRayShape::SimbodyRayShape (*PhysicsEnginePtr* *_physicsEngine*)

Constructor.

Parameters

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

10.167.2.2 gazebo::physics::SimbodyRayShape::SimbodyRayShape (*CollisionPtr* *_collision*)

Constructor.

Parameters

in	<i>_collision</i>	Collision (p. 213) the ray is attached to.
----	-------------------	---

10.167.2.3 virtual gazebo::physics::SimbodyRayShape::~~SimbodyRayShape () [virtual]

Destructor.

10.167.3 Member Function Documentation

10.167.3.1 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. 703).

10.167.3.2 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. 704).

10.167.3.3 virtual void gazebo::physics::SimbodyRayShape::Update () [virtual]

Update the ray collision.

Implements **gazebo::physics::RayShape** (p. 705).

The documentation for this class was generated from the following file:

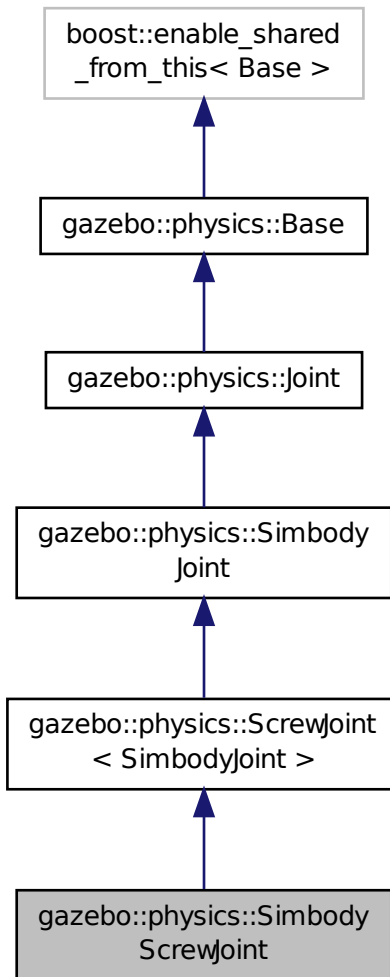
- **SimbodyRayShape.hh**

10.168 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** (int _index) const
Get the angle of an axis helper function.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const
Get the axis of rotation in global coordinate frame.
- virtual **math::Angle GetHighStop** (int _index)

- Get the high stop of an axis(index).*

 - virtual **math::Angle GetLowStop** (int _index)

Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)

Get the max allowed force of an axis(index).
- virtual double **GetThreadPitch** (unsigned int)

Get screw joint thread pitch.
- virtual double **GetVelocity** (int _index) const

Get the rotation rate of an axis(index)
- virtual void **Init** ()

Initialize a joint.
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)

Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetDamping** (int _index, double _damping)

Set the joint damping.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)

Set the max allowed force of an axis(index).
- virtual void **SetThreadPitch** (int _index, double _threadPitch)

Set screw joint thread pitch.
- virtual void **SetVelocity** (int _index, double _angle)

Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

*Load a **ScrewJoint** (p. 746).*
- virtual void **SetForceImpl** (int _index, double _force)

*Set the force applied to this **physics::Joint** (p. 411).*

Additional Inherited Members

10.168.1 Detailed Description

A screw joint.

10.168.2 Constructor & Destructor Documentation

10.168.2.1 gazebo::physics::SimbodyScrewJoint::SimbodyScrewJoint (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.

10.168.2.2 virtual gazebo::physics::SimbodyScrewJoint::~~SimbodyScrewJoint () [virtual]

Destructor.

10.168.3 Member Function Documentation

10.168.3.1 virtual math::Angle gazebo::physics::SimbodyScrewJoint::GetAngleImpl (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. 418).

10.168.3.2 virtual math::Vector3 gazebo::physics::SimbodyScrewJoint::GetGlobalAxis (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. 420).

10.168.3.3 virtual math::Angle gazebo::physics::SimbodyScrewJoint::GetHighStop (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	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Angle of the high stop value.

Implements gazebo::physics::Joint (p. 420).

10.168.3.4 `virtual math::Angle gazebo::physics::SimbodyScrewJoint::GetLowStop (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. 422).

10.168.3.5 `virtual double gazebo::physics::SimbodyScrewJoint::GetMaxForce (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. 423).

10.168.3.6 `virtual double gazebo::physics::SimbodyScrewJoint::GetThreadPitch (unsigned int _index) [virtual]`

Get screw joint thread pitch.

This must be implemented in a child class

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

`_threadPitch` Thread pitch value.

Implements `gazebo::physics::ScrewJoint< SimbodyJoint >` (p. 748).

10.168.3.7 `virtual double gazebo::physics::SimbodyScrewJoint::GetVelocity (int _index) const [virtual]`

Get the rotation rate of an axis(index)

Parameters

in	<code>_index</code>	Index of the axis.
----	---------------------	--------------------

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 423).

10.168.3.8 `virtual void gazebo::physics::SimbodyScrewJoint::Init () [virtual]`

Initialize a joint.

Reimplemented from **gazebo::physics::Joint** (p. 424).

10.168.3.9 `virtual void gazebo::physics::SimbodyScrewJoint::Load (sdf::ElementPtr _sdf) [protected],[virtual]`

Load a **ScrewJoint** (p. 746).

Parameters

in	<code>_sdf</code>	SDF value to load from
----	-------------------	------------------------

Reimplemented from **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 748).

10.168.3.10 `virtual void gazebo::physics::SimbodyScrewJoint::SetAxis (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. 811).

10.168.3.11 `virtual void gazebo::physics::SimbodyScrewJoint::SetDamping (int _index, double _damping) [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.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 811).

10.168.3.12 `virtual void gazebo::physics::SimbodyScrewJoint::SetForceImpl (int _index, double _force) [protected], [virtual]`

Set the force applied to this **physics::Joint** (p. 411).

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. 411)).

Parameters

in	<code><i>_index</i></code>	Index of the axis.
in	<code><i>_force</i></code>	Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only.

Implements **gazebo::physics::SimbodyJoint** (p. 812).

10.168.3.13 `virtual void gazebo::physics::SimbodyScrewJoint::SetHighStop (int _index, const math::Angle & _angle) [virtual]`

Set the high stop of an axis(index).

Parameters

in	<code><i>_index</i></code>	Index of the axis.
in	<code><i>_angle</i></code>	High stop angle.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.168.3.14 `virtual void gazebo::physics::SimbodyScrewJoint::SetLowStop (int _index, const math::Angle & _angle) [virtual]`

Set the low stop of an axis(index).

Parameters

in	<code><i>_index</i></code>	Index of the axis.
in	<code><i>_angle</i></code>	Low stop angle.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.168.3.15 `virtual void gazebo::physics::SimbodyScrewJoint::SetMaxForce (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><i>_index</i></code>	Index of the axis.
in	<code><i>_force</i></code>	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 427).

10.168.3.16 `virtual void gazebo::physics::SimbodyScrewJoint::SetThreadPitch (int _index, double _threadPitch) [virtual]`

Set screw joint thread pitch.

This must be implemented in a child class

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_threadPitch</i></code>	Thread pitch value.

Implements `gazebo::physics::ScrewJoint` < `SimbodyJoint` > (p. 749).

10.168.3.17 `virtual void gazebo::physics::SimbodyScrewJoint::SetVelocity (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. 428).

The documentation for this class was generated from the following file:

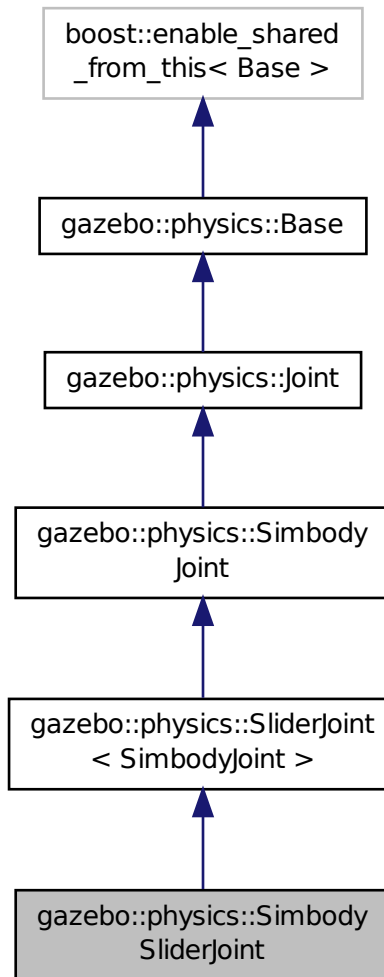
- `SimbodyScrewJoint.hh`

10.169 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** (int _index) const
Get the angle of an axis helper function.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const
Get the axis of rotation in global coordinate frame.
- virtual **math::Angle GetHighStop** (int _index)

- Get the high stop of an axis(index).*

 - virtual **math::Angle GetLowStop** (int _index)

Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)

Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int _index) const

Get the rotation rate of an axis(index)
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)

Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetDamping** (int _index, const double _damping)

Set the joint damping.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)

Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _rate)

Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

*Load a **SliderJoint** (p. 886).*
- virtual void **SetForceImpl** (int _index, double _force)

*Set the force applied to this **physics::Joint** (p. 411).*

Additional Inherited Members

10.169.1 Detailed Description

A slider joint.

10.169.2 Constructor & Destructor Documentation

10.169.2.1 gazebo::physics::SimbodySliderJoint::SimbodySliderJoint (**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.

10.169.2.2 virtual gazebo::physics::SimbodySliderJoint::~~SimbodySliderJoint () [virtual]

Destructor.

10.169.3 Member Function Documentation

10.169.3.1 `virtual math::Angle gazebo::physics::SimbodySliderJoint::GetAngleImpl (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. 418).

10.169.3.2 `virtual math::Vector3 gazebo::physics::SimbodySliderJoint::GetGlobalAxis (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. 420).

10.169.3.3 `virtual math::Angle gazebo::physics::SimbodySliderJoint::GetHighStop (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. 420).

10.169.3.4 `virtual math::Angle gazebo::physics::SimbodySliderJoint::GetLowStop (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	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Angle of the low stop value.

Implements **gazebo::physics::Joint** (p. 422).

10.169.3.5 virtual double gazebo::physics::SimbodySliderJoint::GetMaxForce (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. 423).

10.169.3.6 virtual double gazebo::physics::SimbodySliderJoint::GetVelocity (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. 423).

10.169.3.7 virtual void gazebo::physics::SimbodySliderJoint::Load (sdf::ElementPtr *_sdf*) [protected],[virtual]

Load a **SliderJoint** (p. 886).

Parameters

in	<i>_sdf</i>	SDF values to load from
----	-------------	-------------------------

Reimplemented from **gazebo::physics::SliderJoint** < **SimbodyJoint** > (p. 888).

10.169.3.8 `virtual void gazebo::physics::SimbodySliderJoint::SetAxis (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><i>_index</i></code>	Index of the axis to set.
<code>in</code>	<code><i>_axis</i></code>	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 811).

10.169.3.9 `virtual void gazebo::physics::SimbodySliderJoint::SetDamping (int _index, const double _damping)` `[virtual]`

Set the joint damping.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis to set, currently ignored, to be implemented.
<code>in</code>	<code><i>_damping</i></code>	Damping value for the axis.

Reimplemented from `gazebo::physics::SimbodyJoint` (p. 811).

10.169.3.10 `virtual void gazebo::physics::SimbodySliderJoint::SetForceImpl (int _index, double _force)` `[protected]`,
`[virtual]`

Set the force applied to this `physics::Joint` (p. 411).

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. 411)).

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_force</i></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. 812).

10.169.3.11 `virtual void gazebo::physics::SimbodySliderJoint::SetHighStop (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::Joint` (p. 427).

10.169.3.12 `virtual void gazebo::physics::SimbodySliderJoint::SetLowStop (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::Joint` (p. 427).

10.169.3.13 `virtual void gazebo::physics::SimbodySliderJoint::SetMaxForce (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. 427).

10.169.3.14 `virtual void gazebo::physics::SimbodySliderJoint::SetVelocity (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. 428).

The documentation for this class was generated from the following file:

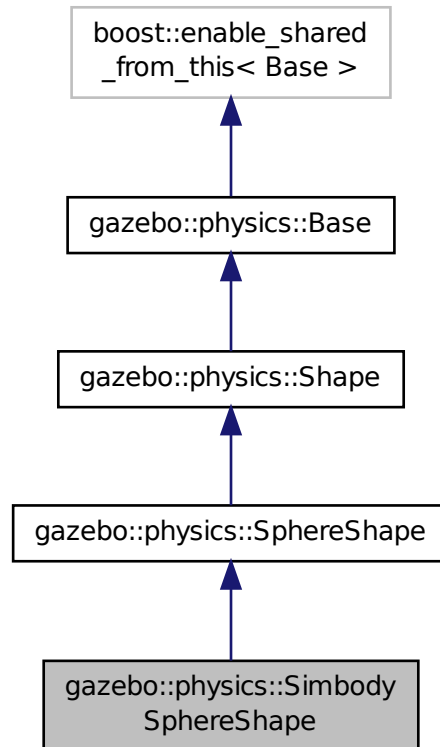
- `SimbodySliderJoint.hh`

10.170 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

10.170.1 Detailed Description

Simbody sphere collision.

10.170.2 Constructor & Destructor Documentation

10.170.2.1 gazebo::physics::SimbodySphereShape::SimbodySphereShape (CollisionPtr *_parent*) [inline]

Constructor.

Parameters

in	<i>_parent</i>	Collision (p. 213) parent pointer
----	----------------	-----------------------------------

10.170.2.2 virtual gazebo::physics::SimbodySphereShape::~~SimbodySphereShape () [inline],[virtual]

Destructor.

10.170.3 Member Function Documentation

10.170.3.1 virtual void gazebo::physics::SimbodySphereShape::SetRadius (double *_radius*) [inline],[virtual]

Set the size.

Parameters

in	<i>_radius</i>	Radius of the sphere.
----	----------------	-----------------------

Reimplemented from gazebo::physics::SphereShape (p. 901).

References gazebo::physics::Shape::collisionParent, gazebo::math::equal(), gzerr, gzwarn, and gazebo::physics::SphereShape::SetRadius().

The documentation for this class was generated from the following file:

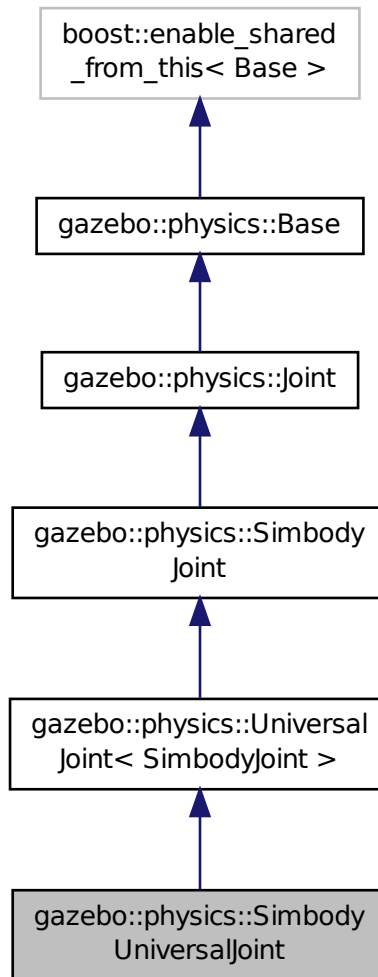
- SimbodySphereShape.hh

10.171 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** (int _index) const
Get the anchor point.
- virtual **math::Vector3 GetAxis** (int _index) const
- virtual **math::Vector3 GetGlobalAxis** (int _index) const
Get the axis of rotation in global coordinate frame.

- virtual **math::Angle GetHighStop** (int _index)
Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int _index)
Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (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 **UniversalJoint** (p. 976).*
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)
Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetDamping** (int _index, double _damping)
Set the joint damping.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)
Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)
Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _angle)
Set the velocity of an axis(index).

Protected Member Functions

- virtual **math::Angle GetAngleImpl** (int _index) const
Get the angle of an axis helper function.
- virtual void **SetForceImpl** (int _index, double _torque)
*Set the force applied to this **physics::Joint** (p. 411).*

Additional Inherited Members

10.171.1 Detailed Description

A simbody universal joint class.

10.171.2 Constructor & Destructor Documentation

10.171.2.1 **gazebo::physics::SimbodyUniversalJoint::SimbodyUniversalJoint** (**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.

10.171.2.2 `virtual gazebo::physics::SimbodyUniversalJoint::~~SimbodyUniversalJoint () [virtual]`

Destuctor.

10.171.3 Member Function Documentation

10.171.3.1 `virtual math::Vector3 gazebo::physics::SimbodyUniversalJoint::GetAnchor (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. 807).

10.171.3.2 `virtual math::Angle gazebo::physics::SimbodyUniversalJoint::GetAngleImpl (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. 418).

10.171.3.3 `virtual math::Vector3 gazebo::physics::SimbodyUniversalJoint::GetAxis (int _index) const [virtual]`

10.171.3.4 `virtual math::Vector3 gazebo::physics::SimbodyUniversalJoint::GetGlobalAxis (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. 420).

10.171.3.5 `virtual math::Angle gazebo::physics::SimbodyUniversalJoint::GetHighStop (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.

Implements `gazebo::physics::Joint` (p. 420).

10.171.3.6 `virtual math::Angle gazebo::physics::SimbodyUniversalJoint::GetLowStop (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. 422).

10.171.3.7 `virtual double gazebo::physics::SimbodyUniversalJoint::GetMaxForce (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. 423).

10.171.3.8 `virtual double gazebo::physics::SimbodyUniversalJoint::GetVelocity (int _index) const [virtual]`

Get the rotation rate of an axis(index)

Parameters

in	<code>_index</code>	Index of the axis.
----	---------------------	--------------------

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 423).

10.171.3.9 `virtual void gazebo::physics::SimbodyUniversalJoint::Init () [virtual]`

Initialize a joint.

Reimplemented from **gazebo::physics::Joint** (p. 424).

10.171.3.10 `virtual void gazebo::physics::SimbodyUniversalJoint::Load (sdf::ElementPtr _sdf) [virtual]`

Load a **UniversalJoint** (p. 976).

Parameters

in	<code>_sdf</code>	SDF values to load from.
----	-------------------	--------------------------

Reimplemented from **gazebo::physics::UniversalJoint< SimbodyJoint >** (p. 978).

10.171.3.11 `virtual void gazebo::physics::SimbodyUniversalJoint::SetAxis (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. 811).

10.171.3.12 `virtual void gazebo::physics::SimbodyUniversalJoint::SetDamping (int _index, double _damping) [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.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 811).

10.171.3.13 `virtual void gazebo::physics::SimbodyUniversalJoint::SetForceImpl (int _index, double _force)` [protected],
[virtual]

Set the force applied to this **physics::Joint** (p. 411).

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. 411)).

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. 812).

10.171.3.14 `virtual void gazebo::physics::SimbodyUniversalJoint::SetHighStop (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::Joint** (p. 427).

10.171.3.15 `virtual void gazebo::physics::SimbodyUniversalJoint::SetLowStop (int _index, const math::Angle & _angle)`
[virtual]

Set the low stop of an axis(index).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_angle</i>	Low stop angle.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.171.3.16 `virtual void gazebo::physics::SimbodyUniversalJoint::SetMaxForce (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. 427).

10.171.3.17 `virtual void gazebo::physics::SimbodyUniversalJoint::SetVelocity (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. 428).

The documentation for this class was generated from the following file:

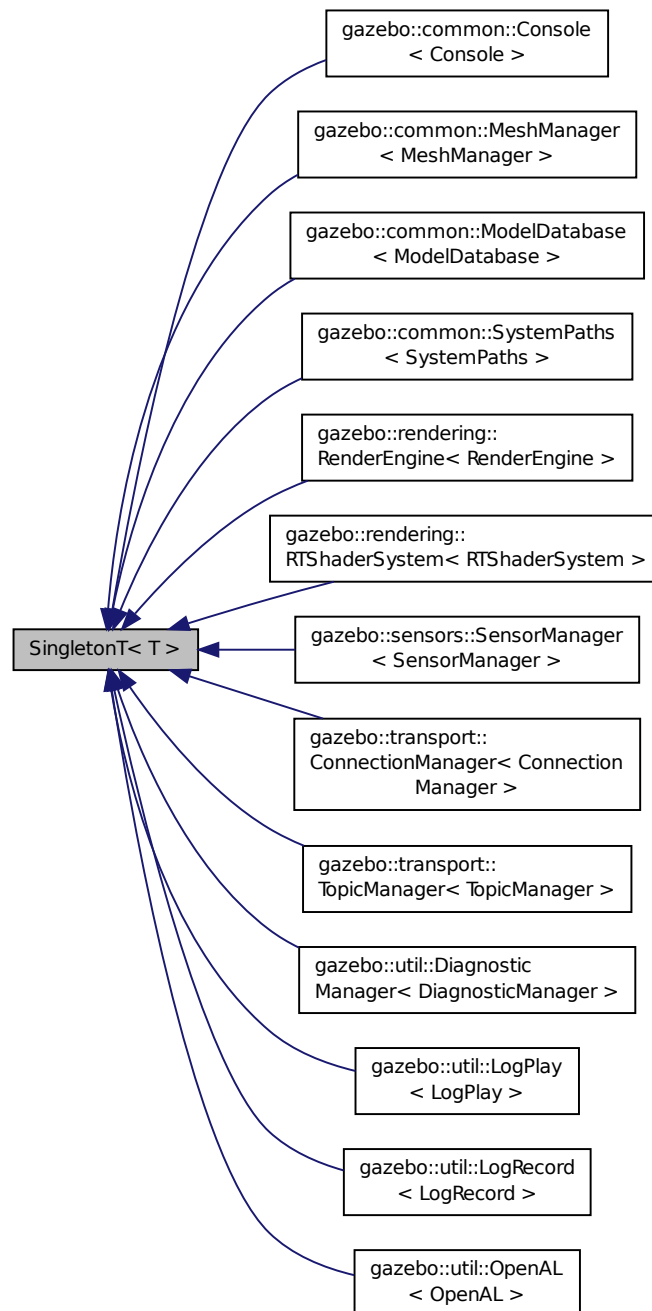
- `SimbodyUniversalJoint.hh`

10.172 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.

10.172.1 Detailed Description

```
template<class T>class SingletonT< T >
```

Singleton template class.

10.172.2 Constructor & Destructor Documentation

10.172.2.1 `template<class T> SingletonT< T >::SingletonT () [inline],[protected]`

Constructor.

10.172.2.2 `template<class T> virtual SingletonT< T >::~~SingletonT () [inline],[protected],[virtual]`

Destructor.

10.172.3 Member Function Documentation

10.172.3.1 `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**

10.173 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

10.173.1 Detailed Description

A skeleton.

10.173.2 Constructor & Destructor Documentation

10.173.2.1 gazebo::common::Skeleton::Skeleton ()

Constructor.

10.173.2.2 gazebo::common::Skeleton::Skeleton (**SkeletonNode** * *_root*)

Constructor.

Parameters

in	<i>_root</i>	node
----	--------------	------

10.173.2.3 virtual gazebo::common::Skeleton::~Skeleton () [virtual]

Destructor.

10.173.3 Member Function Documentation

10.173.3.1 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
----	--------------	----------------------

10.173.3.2 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)

10.173.3.3 void gazebo::common::Skeleton::BuildNodeMap () [protected]

Initializes the handle numbers for each node in the map using breadth first traversal.

10.173.3.4 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

10.173.3.5 math::Matrix4 gazebo::common::Skeleton::GetBindShapeTransform ()

Return bind pose skeletal transform.

Returns

a matrix

10.173.3.6 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

10.173.3.7 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

10.173.3.8 `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

10.173.3.9 `NodeMap` `gazebo::common::Skeleton::GetNodes ()`

Get a copy of the node dictionary.

10.173.3.10 `unsigned int` `gazebo::common::Skeleton::GetNumAnimations ()`

Returns the number of animations.

Returns

the count

10.173.3.11 `unsigned int` `gazebo::common::Skeleton::GetNumJoints ()`

Returns the number of joints.

Returns

the count

10.173.3.12 `unsigned int` `gazebo::common::Skeleton::GetNumNodes ()`

Returns the node count.

Returns

the count

10.173.3.13 `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

10.173.3.14 `SkeletonNode* gazebo::common::Skeleton::GetRootNode ()`

Return the root.

Returns

the root

10.173.3.15 `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

10.173.3.16 `void gazebo::common::Skeleton::PrintTransforms ()`

Outputs the transforms to `std::err` stream.

10.173.3.17 `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
-----------------	-------------------------	----------------

10.173.3.18 `void gazebo::common::Skeleton::SetBindShapeTransform (math::Matrix4 _trans)`

Set the bind pose skeletal transform.

Parameters

in	<i>_trans</i>	the transform
----	---------------	---------------

10.173.3.19 void gazebo::common::Skeleton::SetNumVertAttached (unsigned int *_vertices*)

Resizes the raw node weight array.

Parameters

in	<i>_vertices</i>	the new size
----	------------------	--------------

10.173.3.20 void gazebo::common::Skeleton::SetRootNode (SkeletonNode * *_node*)

Change the root node.

Parameters

in	<i>_node</i>	the new node
----	--------------	--------------

10.173.4 Member Data Documentation

10.173.4.1 std::vector<SkeletonAnimation*> gazebo::common::Skeleton::anim [protected]

the array of animations

10.173.4.2 math::Matrix4 gazebo::common::Skeleton::bindShapeTransform [protected]

the bind pose skeletal transform

10.173.4.3 NodeMap gazebo::common::Skeleton::nodes [protected]

The dictionary of nodes, indexed by name.

10.173.4.4 RawNodeWeights gazebo::common::Skeleton::rawNW [protected]

the node weight table

10.173.4.5 SkeletonNode* gazebo::common::Skeleton::root [protected]

the root node

The documentation for this class was generated from the following file:

- **Skeleton.hh**

10.174 gazebo::common::SkeletonAnimation Class Reference

Skeleton (p. 866) 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

10.174.1 Detailed Description

Skeleton (p. 866) animation.

10.174.2 Constructor & Destructor Documentation

10.174.2.1 gazebo::common::SkeletonAnimation::SkeletonAnimation (const std::string & *_name*)

The Constructor.

Parameters

in	<i>_name</i>	the name of the animation
----	--------------	---------------------------

10.174.2.2 gazebo::common::SkeletonAnimation::~~SkeletonAnimation ()

The destructor.

Clears the list without destroying the animations

10.174.3 Member Function Documentation

10.174.3.1 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

10.174.3.2 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. 648)

10.174.3.3 double gazebo::common::SkeletonAnimation::GetLength () const

Returns the duration of the animations.

Returns

the duration in seconds

10.174.3.4 `std::string gazebo::common::SkeletonAnimation::GetName () const`

Returns the name.

Returns

the name

10.174.3.5 `unsigned int gazebo::common::SkeletonAnimation::GetNodeCount () const`

Returns the number of animation nodes.

Returns

the count

10.174.3.6 `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

10.174.3.7 `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

10.174.3.8 `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)

10.174.3.9 `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

10.174.3.10 `void gazebo::common::SkeletonAnimation::Scale (const double _scale)`

Scales every animation in the animations list.

Parameters

in	<i>_scale</i>	the scaling factor
----	---------------	--------------------

10.174.3.11 `void gazebo::common::SkeletonAnimation::SetName (const std::string & _name)`

Changes the name.

Parameters

in	<i>_name</i>	the new name
----	--------------	--------------

10.174.4 Member Data Documentation

10.174.4.1 `std::map<std::string, NodeAnimation*> gazebo::common::SkeletonAnimation::animations` `[protected]`

a dictionary of node animations

10.174.4.2 double gazebo::common::SkeletonAnimation::length [protected]

the duration of the longest animation

10.174.4.3 std::string gazebo::common::SkeletonAnimation::name [protected]

the node name

The documentation for this class was generated from the following file:

- **SkeletonAnimation.hh**

10.175 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 whether 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

10.175.1 Detailed Description

A skeleton node.

10.175.2 Member Enumeration Documentation

10.175.2.1 enum gazebo::common::SkeletonNode::SkeletonNodeType

enumeration of node types

Enumerator

NODE

JOINT

10.175.3 Constructor & Destructor Documentation

10.175.3.1 gazebo::common::SkeletonNode::SkeletonNode (**SkeletonNode** * *_parent*)

Constructor.

Parameters

<code>in</code>	<code><i>_parent</i></code>	The parent node
-----------------	-----------------------------	-----------------

10.175.3.2 `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

10.175.3.3 `virtual gazebo::common::SkeletonNode::~~SkeletonNode () [virtual]`

Destructor.

10.175.4 Member Function Documentation

10.175.4.1 `void gazebo::common::SkeletonNode::AddChild (SkeletonNode * _child)`

Add a new child.

Parameters

in	<code>_child</code>	a child
----	---------------------	---------

10.175.4.2 `void gazebo::common::SkeletonNode::AddRawTransform (NodeTransform _t)`

Add a raw transform.

Parameters

in	<code>_t</code>	the transform
----	-----------------	---------------

10.175.4.3 `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

10.175.4.4 `SkeletonNode* gazebo::common::SkeletonNode::GetChildById (std::string _id)`

Get child by string id.

Parameters

<code>in</code>	<code>_id</code>	the string id
-----------------	------------------	---------------

Returns

the child skeleton or NULL if not found

10.175.4.5 `SkeletonNode*` gazebo::common::SkeletonNode::GetChildByName (`std::string _name`)

Get child by name.

Parameters

<code>in</code>	<code>_name</code>	the name of the child skeleton
-----------------	--------------------	--------------------------------

Returns

the skeleton, or NULL if not found

10.175.4.6 `unsigned int` gazebo::common::SkeletonNode::GetChildCount ()

Returns the children count.

Returns

the count

10.175.4.7 `unsigned int` gazebo::common::SkeletonNode::GetHandle ()

Get the handle index.

Returns

the handle index

10.175.4.8 `std::string` gazebo::common::SkeletonNode::GetId ()

Returns the index.

Returns

the id string

10.175.4.9 `math::Matrix4` gazebo::common::SkeletonNode::GetInverseBindTransform ()

Retrieve the inverse of the bind pose skeletal transform.

Returns

the transform

10.175.4.10 `math::Matrix4 gazebo::common::SkeletonNode::GetModelTransform ()`

Retrieve the model transform.

Returns

the transform

10.175.4.11 `std::string gazebo::common::SkeletonNode::GetName ()`

Returns the name.

Returns

the name

10.175.4.12 `unsigned int gazebo::common::SkeletonNode::GetNumRawTrans ()`

Return the raw transformations count.

Returns

the count

10.175.4.13 `SkeletonNode* gazebo::common::SkeletonNode::GetParent ()`

Returns the parent node.

Returns

the parent

10.175.4.14 `NodeTransform gazebo::common::SkeletonNode::GetRawTransform (unsigned int i)`

Find a raw transformation.

Parameters

<code>in</code>	<code><i>i</i></code>	the index of the transformation
-----------------	-----------------------	---------------------------------

Returns

the node transform. NO BOUNDS CHECKING PERFORMED

10.175.4.15 `std::vector<NodeTransform> gazebo::common::SkeletonNode::GetRawTransforms ()`

Retrieve the raw transformations.

Returns

an array of transformations

10.175.4.16 `math::Matrix4 gazebo::common::SkeletonNode::GetTransform ()`

Get transform relative to parent.

10.175.4.17 `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)

10.175.4.18 `bool gazebo::common::SkeletonNode::IsJoint ()`

Is a joint query.

Returns

true if the skeleton type is a joint, false otherwise

10.175.4.19 `bool gazebo::common::SkeletonNode::IsRootNode ()`

Queries whether a node has no parent parent.

Returns

true if the node has no parent, false otherwise

10.175.4.20 `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
-----------------	------------------------------------	--

10.175.4.21 `void gazebo::common::SkeletonNode::SetHandle (unsigned int _h)`

Assign a handle number.

Parameters

<code>in</code>	<code><i>_h</i></code>	the handle
-----------------	------------------------	------------

10.175.4.22 `void gazebo::common::SkeletonNode::SetId (std::string _id)`

Change the id string.

Parameters

<code>in</code>	<code><i>_id</i></code>	the new id string
-----------------	-------------------------	-------------------

10.175.4.23 `void gazebo::common::SkeletonNode::SetInitialTransform (math::Matrix4 _tras)`

Sets the initial transformation.

Parameters

<code>in</code>	<code><i>_tras</i></code>	the transformation matrix
-----------------	---------------------------	---------------------------

10.175.4.24 `void gazebo::common::SkeletonNode::SetInverseBindTransform (math::Matrix4 _invBM)`

Assign the inverse of the bind pose skeletal transform.

Parameters

<code>in</code>	<code><i>_invBM</i></code>	the transform
-----------------	----------------------------	---------------

10.175.4.25 `void gazebo::common::SkeletonNode::SetModelTransform (math::Matrix4 _trans, bool _updateChildren = true)`

Set the model transformation.

Parameters

<code>in</code>	<code><i>_trans</i></code>	the transformation
<code>in</code>	<code><i>_updateChildren</i></code>	when true the UpdateChildrenTransforms operation is performed

10.175.4.26 `void gazebo::common::SkeletonNode::SetName (std::string _name)`

Change the name.

Parameters

<code>in</code>	<code><i>_name</i></code>	the new name
-----------------	---------------------------	--------------

10.175.4.27 `void gazebo::common::SkeletonNode::SetParent (SkeletonNode * _parent)`

Set the parent node.

Parameters

<code>in</code>	<code><i>_parent</i></code>	the new parent
-----------------	-----------------------------	----------------

10.175.4.28 `void gazebo::common::SkeletonNode::SetTransform (math::Matrix4 _trans, bool _updateChildren = true)`

Set a transformation.

Parameters

in	<code>_trans</code>	the transformation
in	<code>_updateChildren</code>	when true the UpdateChildrenTransforms operation is performed

10.175.4.29 `void gazebo::common::SkeletonNode::SetType (SkeletonNodeType _type)`

Change the skeleton node type.

Parameters

in	<code>_type</code>	the new type
----	--------------------	--------------

10.175.4.30 `void gazebo::common::SkeletonNode::UpdateChildrenTransforms ()`

Apply model transformations in order for each node in the tree.

10.175.5 Member Data Documentation

10.175.5.1 `std::vector<SkeletonNode*> gazebo::common::SkeletonNode::children` [protected]

the children nodes

10.175.5.2 `unsigned int gazebo::common::SkeletonNode::handle` [protected]

handle index number

10.175.5.3 `std::string gazebo::common::SkeletonNode::id` [protected]

a string identifier

10.175.5.4 `math::Matrix4 gazebo::common::SkeletonNode::initialTransform` [protected]

the initial transformation

10.175.5.5 `math::Matrix4 gazebo::common::SkeletonNode::invBindTransform` [protected]

the inverse of the bind pose skeletal transform

10.175.5.6 `math::Matrix4 gazebo::common::SkeletonNode::modelTransform` [protected]

the model transformation

10.175.5.7 `std::string gazebo::common::SkeletonNode::name` [protected]

the name of the skeletal node

10.175.5.8 `SkeletonNode* gazebo::common::SkeletonNode::parent` [protected]

the parent node

10.175.5.9 `std::vector<NodeTransform> gazebo::common::SkeletonNode::rawTransforms` [protected]

the raw transformation

10.175.5.10 `math::Matrix4 gazebo::common::SkeletonNode::transform` [protected]

the transform

10.175.5.11 `SkeletonNodeType gazebo::common::SkeletonNode::type` [protected]

the type fo node

The documentation for this class was generated from the following file:

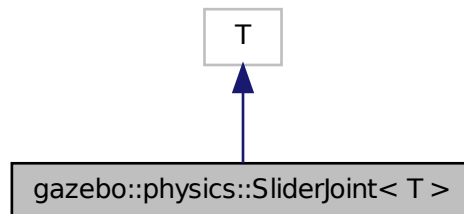
- **Skeleton.hh**

10.176 gazebo::physics::SliderJoint< T > Class Template Reference

A slider joint.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::SliderJoint< T >:



Public Member Functions

- **SliderJoint** (**BasePtr** _parent)

Constructor.

- virtual `~SliderJoint ()`

Destructor.

- virtual `math::Vector3 GetAnchor (int _index) const`

Get the anchor.

- virtual unsigned int `GetAngleCount () const`

- virtual void `Load (sdf::ElementPtr _sdf)`

Load a `SliderJoint` (p. 886).

- virtual void `SetAnchor (int _index, const math::Vector3 &_anchor)`

Set the anchor.

Protected Attributes

- `math::Vector3 fakeAnchor`

The anchor value is not used internally.

10.176.1 Detailed Description

```
template<class T>class gazebo::physics::SliderJoint< T >
```

A slider joint.

10.176.2 Constructor & Destructor Documentation

10.176.2.1 `template<class T> gazebo::physics::SliderJoint< T >::SliderJoint (BasePtr _parent) [inline], [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Parent of the joint.
----	----------------------	----------------------

10.176.2.2 `template<class T> virtual gazebo::physics::SliderJoint< T >::~SliderJoint () [inline], [virtual]`

Destructor.

10.176.3 Member Function Documentation

10.176.3.1 `template<class T > math::Vector3 gazebo::physics::SliderJoint< T >::GetAnchor (int _index) const [virtual]`

Get the anchor.

Parameters

in	<code>_index</code>	Index of the axis. Not used.
----	---------------------	------------------------------

Returns

Anchor for the joint.

10.176.3.2 `template<class T> virtual unsigned int gazebo::physics::SliderJoint< T >::GetAngleCount () const` [inline], [virtual]

10.176.3.3 `template<class T> virtual void gazebo::physics::SliderJoint< T >::Load (sdf::ElementPtr _sdf)` [inline], [virtual]

Load a **SliderJoint** (p. 886).

Parameters

in	<code>_sdf</code>	SDF values to load from
----	-------------------	-------------------------

Reimplemented in **gazebo::physics::SimbodySliderJoint** (p. 853).

10.176.3.4 `template<class T > void gazebo::physics::SliderJoint< T >::SetAnchor (int _index, const math::Vector3 & _anchor)` [virtual]

Set the anchor.

Parameters

in	<code>_index</code>	Index of the axis. Not used.
in	<code>_anchor</code>	Anchor for the axis.

10.176.4 Member Data Documentation

10.176.4.1 `template<class T> math::Vector3 gazebo::physics::SliderJoint< T >::fakeAnchor` [protected]

The anchor value is not used internally.

The documentation for this class was generated from the following file:

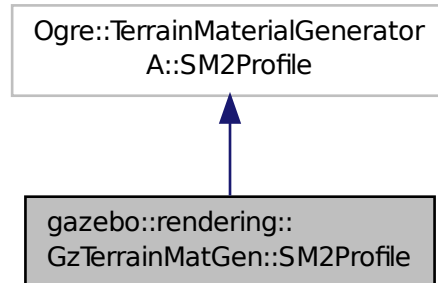
- **SliderJoint.hh**

10.177 gazebo::rendering::GzTerrainMatGen::SM2Profile Class Reference

Shader model 2 profile target.

```
#include <Heightmap.hh>
```


Inheritance diagram for gazebo::rendering::GzTerrainMatGen::SM2Profile:



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)

10.177.1 Detailed Description

Shader model 2 profile target.

10.177.2 Constructor & Destructor Documentation

10.177.2.1 `gazebo::rendering::GzTerrainMatGen::SM2Profile::SM2Profile (Ogre::TerrainMaterialGenerator * _parent, const Ogre::String & _name, const Ogre::String & _desc)`

Constructor.

10.177.2.2 `virtual gazebo::rendering::GzTerrainMatGen::SM2Profile::~SM2Profile () [virtual]`

Destructor.

10.177.3 Member Function Documentation

10.177.3.1 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::addTechnique (const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain, TechniqueType _tt) [protected],[virtual]`

10.177.3.2 `Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generate (const Ogre::Terrain * _terrain)`

10.177.3.3 `Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generateForCompositeMap (const Ogre::Terrain * _terrain)`

10.177.3.4 `void gazebo::rendering::GzTerrainMatGen::SM2Profile::UpdateParams (const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain)`

10.177.3.5 `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:

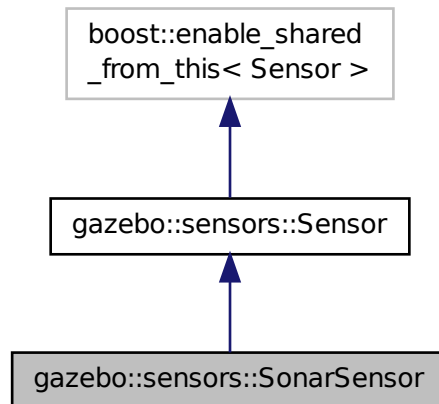
- **Heightmap.hh**

10.178 gazebo::sensors::SonarSensor Class Reference

Sensor (p. 751) 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 void **UpdateImpl** (bool _force)
This gets overwritten by derived sensor types.

Protected Attributes

- **event::EventT**< void(msgs::SonarStamped)> **update**
Update event.

10.178.1 Detailed Description

Sensor (p. 751) with sonar cone.

This sensor uses a cone .

10.178.2 Constructor & Destructor Documentation

10.178.2.1 gazebo::sensors::SonarSensor::SonarSensor ()

Constructor.

10.178.2.2 virtual gazebo::sensors::SonarSensor::~~SonarSensor () [virtual]

Destructor.

10.178.3 Member Function Documentation

10.178.3.1 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.

References gazebo::event::EventT< T >::Connect(), and update.

10.178.3.2 void gazebo::sensors::SonarSensor::DisconnectUpdate (event::ConnectionPtr & _conn) [inline]

Disconnect from the update signal.

Parameters

in	<code>_conn</code>	Connection to remove.
----	--------------------	-----------------------

References `gazebo::event::EventT< T >::Disconnect()`, and `update`.

10.178.3.3 virtual void gazebo::sensors::SonarSensor::Fini () [protected],[virtual]

Finalize the sensor.

Reimplemented from `gazebo::sensors::Sensor` (p. 755).

10.178.3.4 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.

10.178.3.5 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.

10.178.3.6 double gazebo::sensors::SonarSensor::GetRangeMax () const

Get the minimum range of the sonar.

Returns

The sonar's maximum range.

10.178.3.7 double gazebo::sensors::SonarSensor::GetRangeMin () const

Get the minimum range of the sonar.

Returns

The sonar's minimum range.

10.178.3.8 `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. 757).

10.178.3.9 `virtual void gazebo::sensors::SonarSensor::Init () [virtual]`

Initialize the sensor.

Reimplemented from `gazebo::sensors::Sensor` (p. 758).

10.178.3.10 `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. 758).

10.178.3.11 `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. 759).

10.178.3.12 `virtual void 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
----	---------------------	--

Reimplemented from `gazebo::sensors::Sensor` (p. 760).

10.178.4 Member Data Documentation

10.178.4.1 event::EventT<void(msgs::SonarStamped)> gazebo::sensors::SonarSensor::update [protected]

Update event.

Referenced by ConnectUpdate(), and DisconnectUpdate().

The documentation for this class was generated from the following file:

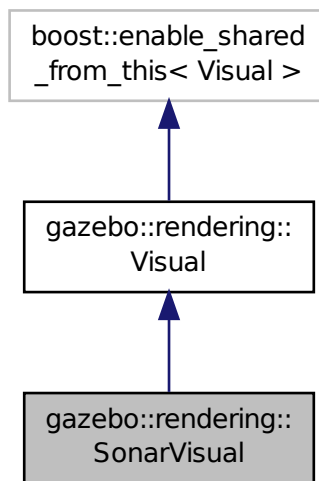
- **SonarSensor.hh**

10.179 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

10.179.1 Detailed Description

Visualization for sonar data.

10.179.2 Constructor & Destructor Documentation

10.179.2.1 `gazebo::rendering::SonarVisual::SonarVisual (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. 1034).
in	<code>_topicName</code>	Name of the topic that has sonar data.

10.179.2.2 `virtual gazebo::rendering::SonarVisual::~SonarVisual () [virtual]`

Destructor.

10.179.3 Member Function Documentation

10.179.3.1 `virtual void gazebo::rendering::SonarVisual::Load () [virtual]`

Load the visual with default parameters.

Reimplemented from `gazebo::rendering::Visual` (p. 1048).

The documentation for this class was generated from the following file:

- **SonarVisual.hh**

10.180 Joint_TEST::SpawnJointOptions Class Reference

Class to hold parameters for spawning joints.

```
#include <Joint_TEST.hh>
```

Public Member Functions

- **SpawnJointOptions ()**
Constructor.
- **~SpawnJointOptions ()**
Destructor.

Public Attributes

- **math::Vector3 axis**
Axis value for spawned joint.
- **math::Pose childLinkPose**
Child link pose for spawned model.
- **math::Pose jointPose**
Joint pose for spawned joint.
- **math::Pose modelPose**
Model pose for spawned model.
- **math::Pose parentLinkPose**
Parent link pose for spawned model.
- **std::string type**
Type of joint to create.
- **common::Time wait**
Length of time to wait for model to spawn in order to return Joint pointer.
- **bool worldChild**
Flag to set child link to the world.
- **bool worldParent**
Flag to set parent link to the world.

10.180.1 Detailed Description

Class to hold parameters for spawning joints.

10.180.2 Constructor & Destructor Documentation

10.180.2.1 `Joint_TEST::SpawnJointOptions::SpawnJointOptions () [inline]`

Constructor.

10.180.2.2 `Joint_TEST::SpawnJointOptions::~~SpawnJointOptions () [inline]`

Destructor.

10.180.3 Member Data Documentation

10.180.3.1 `math::Vector3 Joint_TEST::SpawnJointOptions::axis`

Axis value for spawned joint.

Referenced by `Joint_TEST::SpawnJoint()`.

10.180.3.2 `math::Pose Joint_TEST::SpawnJointOptions::childLinkPose`

Child link pose for spawned model.

Referenced by `Joint_TEST::SpawnJoint()`.

10.180.3.3 `math::Pose Joint_TEST::SpawnJointOptions::jointPose`

Joint pose for spawned joint.

Referenced by `Joint_TEST::SpawnJoint()`.

10.180.3.4 `math::Pose Joint_TEST::SpawnJointOptions::modelPose`

Model pose for spawned model.

Referenced by `Joint_TEST::SpawnJoint()`.

10.180.3.5 `math::Pose Joint_TEST::SpawnJointOptions::parentLinkPose`

Parent link pose for spawned model.

Referenced by `Joint_TEST::SpawnJoint()`.

10.180.3.6 `std::string Joint_TEST::SpawnJointOptions::type`

Type of joint to create.

Referenced by `Joint_TEST::SpawnJoint()`.

10.180.3.7 `common::Time Joint_TEST::SpawnJointOptions::wait`

Length of time to wait for model to spawn in order to return Joint pointer.

Referenced by `Joint_TEST::SpawnJoint()`.

10.180.3.8 `bool Joint_TEST::SpawnJointOptions::worldChild`

Flag to set child link to the world.

Referenced by `Joint_TEST::SpawnJoint()`.

10.180.3.9 `bool Joint_TEST::SpawnJointOptions::worldParent`

Flag to set parent link to the world.

Referenced by `Joint_TEST::SpawnJoint()`.

The documentation for this class was generated from the following file:

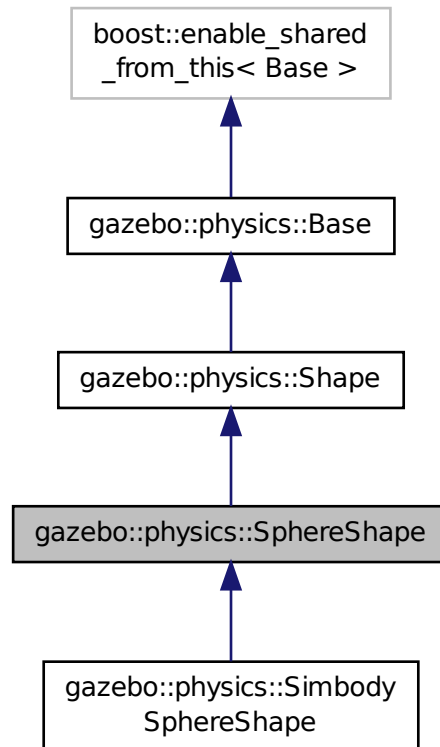
- `Joint_TEST.hh`

10.181 `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 **~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

10.181.1 Detailed Description

Sphere collision shape.

10.181.2 Constructor & Destructor Documentation

10.181.2.1 `gazebo::physics::SphereShape::SphereShape (CollisionPtr _parent)` [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent collision object.
----	----------------	--------------------------

10.181.2.2 `virtual gazebo::physics::SphereShape::~~SphereShape ()` [virtual]

Destructor.

10.181.3 Member Function Documentation

10.181.3.1 `virtual void gazebo::physics::SphereShape::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. 777).

10.181.3.2 `double gazebo::physics::SphereShape::GetRadius () const`

Get the sphere's radius.

Returns

Radius of the sphere.

10.181.3.3 `virtual void gazebo::physics::SphereShape::Init ()` [virtual]

Initialize the sphere.

Implements `gazebo::physics::Shape` (p. 777).

10.181.3.4 `virtual void gazebo::physics::SphereShape::ProcessMsg (const msgs::Geometry & _msg)` [virtual]

Process a geometry message.

Parameters

in	<code>_msg</code>	The message to set values from.
----	-------------------	---------------------------------

Implements **gazebo::physics::Shape** (p. 778).

10.181.3.5 virtual void gazebo::physics::SphereShape::SetRadius (double *.radius*) [virtual]

Set the size.

Parameters

in	<code>_radius</code>	Radius of the sphere.
----	----------------------	-----------------------

Reimplemented in **gazebo::physics::SimbodySphereShape** (p. 857).

Referenced by gazebo::physics::SimbodySphereShape::SetRadius().

10.181.3.6 virtual void gazebo::physics::SphereShape::SetScale (const math::Vector3 & *.scale*) [virtual]

Set the scale of the sphere.

Parameters

in	<code>_scale</code>	Scale to set the sphere to.
----	---------------------	-----------------------------

Implements **gazebo::physics::Shape** (p. 778).

The documentation for this class was generated from the following file:

- **SphereShape.hh**

10.182 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.

10.182.1 Detailed Description

Convert spherical coordinates for planetary surfaces.

10.182.2 Member Enumeration Documentation

10.182.2.1 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

10.182.3 Constructor & Destructor Documentation

10.182.3.1 gazebo::common::SphericalCoordinates::SphericalCoordinates ()

Constructor.

10.182.3.2 gazebo::common::SphericalCoordinates::SphericalCoordinates (const SurfaceType *_type*)

Constructor with surface type input.

Parameters

<i>in</i>	<i>_type</i>	SurfaceType specification.
-----------	--------------	----------------------------

10.182.3.3 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

<i>in</i>	<i>_type</i>	SurfaceType specification.
<i>in</i>	<i>_latitude</i>	Reference latitude.
<i>in</i>	<i>_longitude</i>	Reference longitude.
<i>in</i>	<i>_elevation</i>	Reference elevation.
<i>in</i>	<i>_heading</i>	Heading offset.

10.182.3.4 gazebo::common::SphericalCoordinates::~~SphericalCoordinates ()

Destructor.

10.182.4 Member Function Documentation

10.182.4.1 static SurfaceType gazebo::common::SphericalCoordinates::Convert (const std::string & *_str*) [static]

Convert a string to a SurfaceType.

Parameters

<i>in</i>	<i>_str</i>	String to convert.
-----------	-------------	--------------------

Returns

Conversion to SurfaceType.

10.182.4.2 double gazebo::common::SphericalCoordinates::GetElevationReference () const

Get reference elevation in meters.

Returns

Reference elevation.

10.182.4.3 `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.

10.182.4.4 `math::Angle gazebo::common::SphericalCoordinates::GetLatitudeReference () const`

Get reference geodetic latitude.

Returns

Reference geodetic latitude.

10.182.4.5 `math::Angle gazebo::common::SphericalCoordinates::GetLongitudeReference () const`

Get reference longitude.

Returns

Reference longitude.

10.182.4.6 `SurfaceType gazebo::common::SphericalCoordinates::GetSurfaceType () const`

Get SurfaceType currently in use.

Returns

Current SurfaceType value.

10.182.4.7 `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

<code>in</code>	<code>_xyz</code>	Cartesian vector in gazebo's world frame.
-----------------	-------------------	---

Returns

Rotated vector with components (x,y,z): (East, North, Up).

10.182.4.8 `void gazebo::common::SphericalCoordinates::SetElevationReference (double _elevation)`

Set reference elevation above sea level in meters.

Parameters

<code>in</code>	<code><i>_elevation</i></code>	Reference elevation.
-----------------	--------------------------------	----------------------

10.182.4.9 `void gazebo::common::SphericalCoordinates::SetHeadingOffset (const math::Angle & _angle)`

Set heading angle offset for gazebo frame.

Parameters

<code>in</code>	<code><i>_angle</i></code>	Heading offset for gazebo frame.
-----------------	----------------------------	----------------------------------

10.182.4.10 `void gazebo::common::SphericalCoordinates::SetLatitudeReference (const math::Angle & _angle)`

Set reference geodetic latitude.

Parameters

<code>in</code>	<code><i>_angle</i></code>	Reference geodetic latitude.
-----------------	----------------------------	------------------------------

10.182.4.11 `void gazebo::common::SphericalCoordinates::SetLongitudeReference (const math::Angle & _angle)`

Set reference longitude.

Parameters

<code>in</code>	<code><i>_angle</i></code>	Reference longitude.
-----------------	----------------------------	----------------------

10.182.4.12 `void gazebo::common::SphericalCoordinates::SetSurfaceType (const SurfaceType & _type)`

Set SurfaceType for planetary surface model.

Parameters

<code>in</code>	<code><i>_type</i></code>	SurfaceType value.
-----------------	---------------------------	--------------------

10.182.4.13 `math::Vector3 gazebo::common::SphericalCoordinates::SphericalFromLocal (const math::Vector3 & _xyz) const`

Convert a Cartesian position vector to geodetic coordinates.

Parameters

in	<code>_xyz</code>	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**

10.183 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.

10.183.1 Detailed Description

Splines.

10.183.2 Constructor & Destructor Documentation

10.183.2.1 gazebo::math::Spline::Spline ()

constructor

10.183.2.2 gazebo::math::Spline::~~Spline ()

destructor

10.183.3 Member Function Documentation

10.183.3.1 void gazebo::math::Spline::AddPoint (const Vector3 & _pt)

Adds a control point to the end of the spline.

Parameters

in	_pt	point to add
----	-----	--------------

10.183.3.2 void gazebo::math::Spline::Clear ()

Clears all the points in the spline.

10.183.3.3 Vector3 gazebo::math::Spline::GetPoint (unsigned int _index) const

Gets the detail of one of the control points of the spline.

Parameters

<code>in</code>	<code>_index</code>	the control point index
-----------------	---------------------	-------------------------

Returns

the control point, or [0,0,0] and a message on the error stream

10.183.3.4 `unsigned int gazebo::math::Spline::GetPointCount () const`

Gets the number of control points in the spline.

Returns

the count

10.183.3.5 `Vector3 gazebo::math::Spline::GetTangent (unsigned int _index) const`

Get the tangent value for a point.

Parameters

<code>in</code>	<code>_index</code>	the control point index
-----------------	---------------------	-------------------------

10.183.3.6 `double gazebo::math::Spline::GetTension () const`

Get the tension value.

Returns

The value of the tension, which is between 0.0 and 1.0

10.183.3.7 `Vector3 gazebo::math::Spline::Interpolate (double _t) const`

Returns an interpolated point based on a parametric value over the whole series.

Parameters

<code>in</code>	<code>_t</code>	parameter (range 0 to 1)
-----------------	-----------------	--------------------------

10.183.3.8 `Vector3 gazebo::math::Spline::Interpolate (unsigned int _fromIndex, double _t) const`

Interpolates a single segment of the spline given a parametric value.

Parameters

<code>in</code>	<code>_fromIndex</code>	The point index to treat as t = 0. fromIndex + 1 is deemed to be t = 1
<code>in</code>	<code>_t</code>	Parametric value

10.183.3.9 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.

10.183.3.10 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

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.
----	------------------	--

10.183.3.11 void gazebo::math::Spline::SetTension (double *_t*)

Set the tension parameter.

A value of 0 = Catmull-Rom spline.

Parameters

in	<i>_t</i>	Tension value between 0.0 and 1.0
----	-----------	-----------------------------------

10.183.3.12 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

in	<i>_index</i>	the control point index
in	<i>_value</i>	the new position

10.183.4 Member Data Documentation

10.183.4.1 `bool gazebo::math::Spline::autoCalc` [protected]

when true, the tangents are recalculated when the control point change

10.183.4.2 `Matrix4 gazebo::math::Spline::coeffs` [protected]

Matrix of coefficients.

10.183.4.3 `std::vector<Vector3> gazebo::math::Spline::points` [protected]

control points

10.183.4.4 `std::vector<Vector3> gazebo::math::Spline::tangents` [protected]

tangents

10.183.4.5 `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:

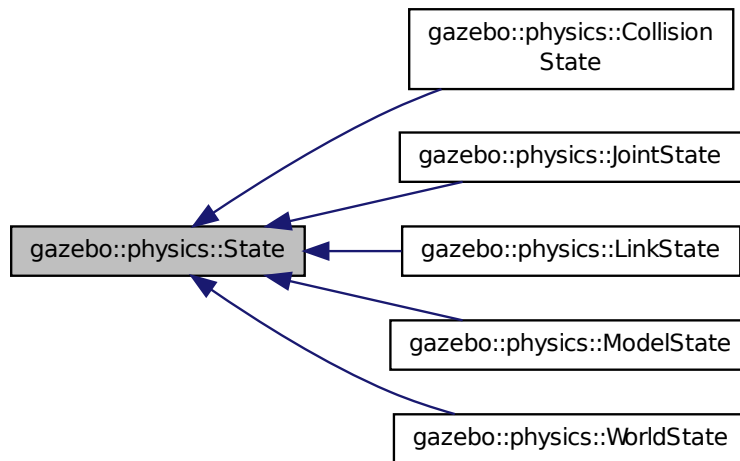
- `Spline.hh`

10.184 `gazebo::physics::State` Class Reference

State (p. 910) 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. 910).*
- **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. 910).*
- 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. 910).*
- **common::Time** **realTime**
- **common::Time** **simTime**
- **common::Time** **wallTime**
Times for the state data.

10.184.1 Detailed Description

State (p. 910) of an entity.

This is the base class for all **State** (p. 910) information.

10.184.2 Constructor & Destructor Documentation

10.184.2.1 gazebo::physics::State::State ()

Default constructor.

10.184.2.2 gazebo::physics::State::State (const std::string & _name, const common::Time & _realTime, const common::Time & _simTime)

Constructor.

Construct a **State** (p. 910) object using some basic information.

Parameters

<code>_name</code>	Name associated with the State (p. 910) information. This is typically the name of an Entity (p. 293). <code>_realTime</code> Clock time since simulation started.
<code>_simTime</code>	Simulation time associated with this State (p. 910) info.

10.184.2.3 virtual gazebo::physics::State::~~State () [virtual]

Destructor.

10.184.3 Member Function Documentation

10.184.3.1 std::string gazebo::physics::State::GetName () const

Get the name associated with this **State** (p. 910).

Returns

Name associated with this state information. Typically a name of an **Entity** (p. 293).

10.184.3.2 **common::Time gazebo::physics::State::GetRealTime () const**

Get the real time when this state was generated.

Returns

Clock time since simulation was stated.

10.184.3.3 **common::Time gazebo::physics::State::GetSimTime () const**

Get the sim time when this state was generated.

Returns

Simulation time when the data was recorded.

10.184.3.4 **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. 910) data was recorded.

10.184.3.5 **virtual void gazebo::physics::State::Load (const sdf::ElementPtr *_elem*) [virtual]**

Load state from SDF element.

Populates the **State** (p. 910) information from data stored in an SDF::Element

Parameters

<i>_elem</i>	Pointer to the SDF::Element
--------------	-----------------------------

Reimplemented in **gazebo::physics::ModelState** (p. 561), **gazebo::physics::LinkState** (p. 481), **gazebo::physics::WorldState** (p. 1087), **gazebo::physics::JointState** (p. 439), and **gazebo::physics::CollisionState** (p. 225).

10.184.3.6 **State gazebo::physics::State::operator- (const State & *_state*) const**

Subtraction operator.

Parameters

<i>in</i>	<i>_pt</i>	A state to subtract.
-----------	------------	----------------------

Returns

The resulting state.

10.184.3.7 State& gazebo::physics::State::operator= (const State & _state)

Assignment operator.

Parameters

in	_state	State (p. 910) value
----	--------	-----------------------------

Returns

this

10.184.3.8 void gazebo::physics::State::SetName (const std::string & _name)

Set the name associated with this **State** (p. 910).

Parameters

in	_name	Name associated with this state information. Typically the name of an Entity (p. 293).
----	-------	---

10.184.3.9 virtual void gazebo::physics::State::SetRealTime (const common::Time & _time) [virtual]

Set the real time when this state was generated.

Parameters

in	_time	Clock time since simulation was started.
----	-------	--

Reimplemented in **gazebo::physics::ModelState** (p. 562), **gazebo::physics::LinkState** (p. 482), and **gazebo::physics::WorldState** (p. 1088).

10.184.3.10 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. 562), **gazebo::physics::LinkState** (p. 482), and **gazebo::physics::WorldState** (p. 1088).

10.184.3.11 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. 910) data was recorded.
----	-------	---

Reimplemented in **gazebo::physics::ModelState** (p. 562), **gazebo::physics::LinkState** (p. 482), and **gazebo::physics::WorldState** (p. 1088).

10.184.4 Member Data Documentation

10.184.4.1 `std::string gazebo::physics::State::name` [protected]

Name associated with this **State** (p. 910).

10.184.4.2 `common::Time gazebo::physics::State::realTime` [protected]

10.184.4.3 `common::Time gazebo::physics::State::simTime` [protected]

10.184.4.4 `common::Time gazebo::physics::State::wallTime` [protected]

Times for the state data.

The documentation for this class was generated from the following file:

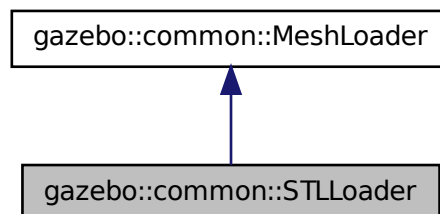
- **State.hh**

10.185 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.

10.185.1 Detailed Description

Class used to load STL mesh files.

10.185.2 Constructor & Destructor Documentation

10.185.2.1 gazebo::common::STLLoader::STLLoader ()

Constructor.

10.185.2.2 virtual gazebo::common::STLLoader::~~STLLoader () [virtual]

Destructor.

10.185.3 Member Function Documentation

10.185.3.1 virtual Mesh* gazebo::common::STLLoader::Load (const std::string & _filename) [virtual]

Creates a new mesh and loads the data from a file.

Parameters

<code>in</code>	<code>_filename</code>	the mesh file
-----------------	------------------------	---------------

Implements `gazebo::common::MeshLoader` (p. 528).

The documentation for this class was generated from the following file:

- `STLLoader.hh`

10.186 gazebo::common::SubMesh Class Reference

A child mesh.

```
#include <Mesh.hh>
```

Public Types

- enum `PrimitiveType` {
POINTS, LINES, LINESTRIPS, 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*.*

10.186.1 Detailed Description

A child mesh.

10.186.2 Member Enumeration Documentation

10.186.2.1 enum gazebo::common::SubMesh::PrimitiveType

An enumeration of the geometric mesh primitives.

Enumerator

POINTS

LINES

LINESTRIPS

TRIANGLES

TRIFANS

TRISTRIPS

10.186.3 Constructor & Destructor Documentation

10.186.3.1 gazebo::common::SubMesh::SubMesh ()

Constructor.

10.186.3.2 gazebo::common::SubMesh::SubMesh (const SubMesh * *_mesh*)

Copy Constructor.

10.186.3.3 virtual gazebo::common::SubMesh::~~SubMesh () [virtual]

Destructor.

10.186.4 Member Function Documentation

10.186.4.1 void gazebo::common::SubMesh::AddIndex (unsigned int *_i*)

Add an index to the mesh.

Parameters

<i>in</i>	<i>_i</i>	the new vertex index
-----------	-----------	----------------------

10.186.4.2 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)

10.186.4.3 void gazebo::common::SubMesh::AddNormal (const math::Vector3 & *_n*)

Add a normal to the mesh.

Parameters

in	<i>_n</i>	the normal
----	-----------	------------

10.186.4.4 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

10.186.4.5 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

10.186.4.6 void gazebo::common::SubMesh::AddVertex (const math::Vector3 & *_v*)

Add a vertex to the mesh.

Parameters

in	<i>_v</i>	the new position
----	-----------	------------------

10.186.4.7 void gazebo::common::SubMesh::AddVertex (double *_x*, double *_y*, double *_z*)

Add a vertex to the mesh.

Parameters

in	<code>_x</code>	position along x
in	<code>_y</code>	position along y
in	<code>_z</code>	position along z

10.186.4.8 `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	<code>_center</code>	Location of the mesh center.
----	----------------------	------------------------------

10.186.4.9 `void gazebo::common::SubMesh::CopyNormals (const std::vector< math::Vector3 > & _norms)`

Copy normals from a vector.

Parameters

in	<code>_norms</code>	to copy from
----	---------------------	--------------

10.186.4.10 `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
----	---------------------	---------------------------

10.186.4.11 `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>	

10.186.4.12 `void gazebo::common::SubMesh::GenSphericalTexCoord (const math::Vector3 & _center)`

Generate texture coordinates using spherical projection from center.

Parameters

in	<code>_center</code>	
----	----------------------	--

10.186.4.13 `unsigned int gazebo::common::SubMesh::GetIndex (unsigned int _i) const`

Get an index.

Parameters

<code>in</code>	<code><i>_i</i></code>
-----------------	------------------------

10.186.4.14 `unsigned int gazebo::common::SubMesh::GetIndexCount () const`

Return the number of indices.

10.186.4.15 `unsigned int gazebo::common::SubMesh::GetMaterialIndex () const`

Get the material index.

10.186.4.16 `math::Vector3 gazebo::common::SubMesh::GetMax () const`

Get the maximum X, Y, Z values.

Returns

10.186.4.17 `unsigned int gazebo::common::SubMesh::GetMaxIndex () const`

Get the highest index value.

10.186.4.18 `math::Vector3 gazebo::common::SubMesh::GetMin () const`

Get the minimum X, Y, Z values.

Returns

10.186.4.19 `std::string gazebo::common::SubMesh::GetName () const`

Get the name of this mesh.

Returns

the name

10.186.4.20 **NodeAssignment** gazebo::common::SubMesh::GetNodeAssignment (unsigned int *_i*) const

Get a vertex - skeleton node assignment.

Parameters

in	<i>_i</i>	the index of the assignment
----	-----------	-----------------------------

10.186.4.21 unsigned int gazebo::common::SubMesh::GetNodeAssignmentsCount () const

Return the number of vertex - skeleton node assignments.

10.186.4.22 **math::Vector3** gazebo::common::SubMesh::GetNormal (unsigned int *_i*) const

Get a normal.

Parameters

in	<i>_i</i>	the normal index
----	-----------	------------------

Returns

the orientation of the normal, or throws an exception

10.186.4.23 unsigned int gazebo::common::SubMesh::GetNormalCount () const

Return the number of normals.

10.186.4.24 **PrimitiveType** gazebo::common::SubMesh::GetPrimitiveType () const

Get the primitive type.

Returns

the primitive type

10.186.4.25 **math::Vector2d** gazebo::common::SubMesh::GetTexCoord (unsigned int *_i*) const

Get a tex coord.

Parameters

in	<i>_i</i>	the texture index
----	-----------	-------------------

Returns

the texture coordinates

10.186.4.26 `unsigned int gazebo::common::SubMesh::GetTexCoordCount () const`

Return the number of texture coordinates.

10.186.4.27 `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

10.186.4.28 `unsigned int gazebo::common::SubMesh::GetVertexCount () const`

Return the number of vertices.

10.186.4.29 `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>	
-----------------	-----------------------	--

10.186.4.30 `bool gazebo::common::SubMesh::HasVertex (const math::Vector3 & v) const`

Return true if this submesh has the vertex.

Parameters

<code>in</code>	<code><i>v</i></code>	
-----------------	-----------------------	--

10.186.4.31 `void gazebo::common::SubMesh::RecalculateNormals ()`

Recalculate all the normals.

10.186.4.32 `void gazebo::common::SubMesh::Scale (double factor)`

Scale all vertices by `factor`.

Parameters

<code>in</code>	<code><i>factor</i></code>	Scaling factor
-----------------	----------------------------	----------------

10.186.4.33 void gazebo::common::SubMesh::SetIndexCount (unsigned int *_count*)

Resize the index array.

Parameters

in	<i>_count</i>	the new size of the array
----	---------------	---------------------------

10.186.4.34 void gazebo::common::SubMesh::SetMaterialIndex (unsigned int *_index*)

Set the material index.

Relates to the parent mesh material list

Parameters

in	<i>_index</i>	
----	---------------	--

10.186.4.35 void gazebo::common::SubMesh::SetName (const std::string & *_n*)

Set the name of this mesh.

Parameters

in	<i>_n</i>	the name to set
----	-----------	-----------------

10.186.4.36 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

10.186.4.37 void gazebo::common::SubMesh::SetNormalCount (unsigned int *_count*)

Resize the normal array.

Parameters

in	<i>_count</i>	the new size of the array
----	---------------	---------------------------

10.186.4.38 void gazebo::common::SubMesh::SetPrimitiveType (PrimitiveType *_type*)

Set the primitive type.

Parameters

in	<i>_type</i>	the type
----	--------------	----------

10.186.4.39 void gazebo::common::SubMesh::SetScale (const math::Vector3 & *_factor*)

Scale all vertices by the *_factor* vector.

Parameters

in	<i>_factor</i>	Scaling vector
----	----------------	----------------

10.186.4.40 void gazebo::common::SubMesh::SetSubMeshCenter (math::Vector3 *_center*)

Reset mesh center to geometric center.

Parameters

in	<i>_center</i>	
----	----------------	--

10.186.4.41 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>	

10.186.4.42 void gazebo::common::SubMesh::SetTexCoordCount (unsigned int *_count*)

Resize the texture coordinate array.

Parameters

in	<i>_count</i>	
----	---------------	--

10.186.4.43 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

10.186.4.44 void gazebo::common::SubMesh::SetVertexCount (unsigned int *_count*)

Resize the vertex array.

Parameters

in	<i>_count</i>	the new size of the array
----	---------------	---------------------------

10.186.4.45 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**

10.187 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.

10.187.1 Detailed Description

Options for a subscription.

10.187.2 Constructor & Destructor Documentation

10.187.2.1 `gazebo::transport::SubscribeOptions::SubscribeOptions ()` `[inline]`

Constructor.

10.187.3 Member Function Documentation

10.187.3.1 `bool gazebo::transport::SubscribeOptions::GetLatching () const` `[inline]`

Are we latching?

Returns

true if we're latching the latest message, false otherwise

10.187.3.2 `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

10.187.3.3 `NodePtr gazebo::transport::SubscribeOptions::GetNode () const` `[inline]`

Get the node we're subscribed to.

Returns

The associated node

10.187.3.4 `std::string gazebo::transport::SubscribeOptions::GetTopic () const` `[inline]`

Get the topic we're subscribed to.

Returns

The topic we're subscribed to

10.187.3.5 `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><i>_topic</i></code>	Topic we're subscribing to
<code>in, out</code>	<code><i>_node</i></code>	The associated node
<code>in</code>	<code><i>_latching</i></code>	If true, latch the latest message; if false, don't latch

References gzthrow, and NULL.

Referenced by gazebo::transport::Node::Subscribe().

10.187.3.6 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

in	<i>_topic</i>	Topic we're subscribing to
in, out	<i>_node</i>	The associated node
in	<i>_latching</i>	If true, latch the latest message; if false, don't latch

The documentation for this class was generated from the following file:

- **SubscribeOptions.hh**

10.188 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.

10.188.1 Detailed Description

A subscriber to a topic.

10.188.2 Constructor & Destructor Documentation

10.188.2.1 gazebo::transport::Subscriber::Subscriber (const std::string & *_topic*, NodePtr *_node*)

Constructor.

Parameters

in	<code>_topic</code>	The topic we're subscribing to
in	<code>_node</code>	The associated node

10.188.2.2 `virtual gazebo::transport::Subscriber::~~Subscriber () [virtual]`

Destructor.

10.188.3 Member Function Documentation

10.188.3.1 `unsigned int gazebo::transport::Subscriber::GetCallbackId () const`

10.188.3.2 `std::string gazebo::transport::Subscriber::GetTopic () const`

Get the topic name.

Returns

The topic name

10.188.3.3 `void gazebo::transport::Subscriber::SetCallbackId (unsigned int _id)`

10.188.3.4 `void gazebo::transport::Subscriber::Unsubscribe () const`

Unsubscribe from the topic.

The documentation for this class was generated from the following file:

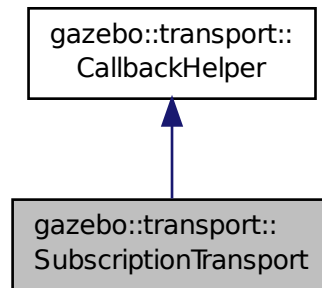
- **Subscriber.hh**

10.189 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

10.189.1 Detailed Description

transport/transport.hh

Handles sending data over the wire to remote subscribers

10.189.2 Constructor & Destructor Documentation

10.189.2.1 gazebo::transport::SubscriptionTransport::SubscriptionTransport ()

Constructor.

10.189.2.2 `virtual gazebo::transport::SubscriptionTransport::~~SubscriptionTransport () [virtual]`

Destructor.

10.189.3 Member Function Documentation

10.189.3.1 `const ConnectionPtr& gazebo::transport::SubscriptionTransport::GetConnection () const`

Get the connection we're using.

Returns

Pointer to the connection we're using

10.189.3.2 `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	_newdata	The message to be handled
----	----------	---------------------------

Returns

true if the message was handled successfully, false otherwise

Parameters

in	_cb	If non-null, callback to be invoked after transmission is complete.
in	_id	ID associated with the message data.

Implements `gazebo::transport::CallbackHelper` (p. 175).

10.189.3.3 `virtual bool gazebo::transport::SubscriptionTransport::HandleMessage (MessagePtr _newMsg) [virtual]`

Process new incoming message.

Parameters

in	_newMsg	Incoming message to be processed
----	---------	----------------------------------

Returns

true if successfully processed; false otherwise

Implements `gazebo::transport::CallbackHelper` (p. 176).

10.189.3.4 `void gazebo::transport::SubscriptionTransport::Init (ConnectionPtr _conn, bool _latching)`

Initialize the publication link.

Parameters

in	<code>_conn</code>	The connection to use
in	<code>_latching</code>	If true, latch the latest message; if false, don't latch

10.189.3.5 `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. 176).

The documentation for this class was generated from the following file:

- `SubscriptionTransport.hh`

10.190 gazebo::physics::SurfaceParams Class Reference

`SurfaceParams` (p. 933) defines various Surface contact parameters.

```
#include <physics/physics.hh>
```

Public Member Functions

- `SurfaceParams ()`
Constructor.
- `virtual ~SurfaceParams ()`
Destructor.
- `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)`

Public Attributes

- double `bounce`
bounce restitution coefficient [0,1], with 0 being inelastic, and 1 being perfectly elastic.
- double `bounceThreshold`
minimum contact velocity for bounce to take effect, otherwise the collision is treated as an inelastic collision.
- double `cfm`
Constraint Force Mixing parameter.
- bool `collideWithoutContact`
Allow collision checking without generating a contact joint.
- unsigned int `collideWithoutContactBitmask`
Custom collision filtering used when collideWithoutContact is true.

- double **erp**
Error Reduction Parameter.
- **math::Vector3 fdir1**
*Primary friction direction for dry friction coefficient (**SurfaceParams::mu1** (p. 937)) of the friction pyramid.*
- double **kd**
*spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 935) and **SurfaceParams::erp** (p. 935).*
- double **kp**
*spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 935) and **SurfaceParams::erp** (p. 935).*
- double **maxVel**
Maximum interpenetration error correction velocity.
- double **minDepth**
Minimum depth before ERP takes effect.
- double **mu1**
Dry friction coefficient in the primary friction direction as defined by the friction pyramid.
- double **mu2**
Dry friction coefficient in the second friction direction as defined by the friction pyramid.
- double **slip1**
Artificial contact slip in the primary friction direction.
- double **slip2**
Artificial contact slip in the secondary friction dirction.

10.190.1 Detailed Description

SurfaceParams (p. 933) defines various Surface contact parameters.

These parameters defines the properties of a **physics::Contact** (p. 253) constraint.

10.190.2 Constructor & Destructor Documentation

10.190.2.1 gazebo::physics::SurfaceParams::SurfaceParams ()

Constructor.

10.190.2.2 virtual gazebo::physics::SurfaceParams::~~SurfaceParams () [virtual]

Destructor.

10.190.3 Member Function Documentation

10.190.3.1 void gazebo::physics::SurfaceParams::FillMsg (msgs::Surface & _msg)

Fill in a surface message.

Parameters

in	_msg	Message to fill with this object's values.
----	------	--

10.190.3.2 `virtual void gazebo::physics::SurfaceParams::Load (sdf::ElementPtr _sdf) [virtual]`

Load the contact params.

Parameters

<code>in</code>	<code>_sdf</code>	SDF values to load from.
-----------------	-------------------	--------------------------

10.190.3.3 `virtual void gazebo::physics::SurfaceParams::ProcessMsg (const msgs::Surface & _msg) [virtual]`

10.190.4 Member Data Documentation

10.190.4.1 `double gazebo::physics::SurfaceParams::bounce`

bounce restitution coefficient [0,1], with 0 being inelastic, and 1 being perfectly elastic.

See Also

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.190.4.2 `double gazebo::physics::SurfaceParams::bounceThreshold`

minimum contact velocity for bounce to take effect, otherwise the collision is treated as an inelastic collision.

See Also

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.190.4.3 `double gazebo::physics::SurfaceParams::cfm`

Constraint Force Mixing parameter.

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_0 for more details.

10.190.4.4 `bool gazebo::physics::SurfaceParams::collideWithoutContact`

Allow collision checking without generating a contact joint.

10.190.4.5 `unsigned int gazebo::physics::SurfaceParams::collideWithoutContactBitmask`

Custom collision filtering used when `collideWithoutContact` is true.

10.190.4.6 `double gazebo::physics::SurfaceParams::erp`

Error Reduction Parameter.

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_0 for more details.

10.190.4.7 `math::Vector3 gazebo::physics::SurfaceParams::fdir1`

Primary friction direction for dry friction coefficient (**SurfaceParams::mu1** (p. 937)) of the friction pyramid.

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

10.190.4.8 `double gazebo::physics::SurfaceParams::kd`

spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 935) and **SurfaceParams::erp** (p. 935).

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_2 for more details.

10.190.4.9 `double gazebo::physics::SurfaceParams::kp`

spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 935) and **SurfaceParams::erp** (p. 935).

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_2 for more details.

10.190.4.10 `double gazebo::physics::SurfaceParams::maxVel`

Maximum interpenetration error correction velocity.

If set to 0, two objects interpenetrating each other will not be pushed apart.

See Also

See `dWroldSetContactMaxCorrectingVel` (http://www.ode.org/ode-latest-userguide.html#sec_5_2_0)

10.190.4.11 `double gazebo::physics::SurfaceParams::minDepth`

Minimum depth before ERP takes effect.

See Also

See `dWorldSetContactSurfaceLayer` (http://www.ode.org/ode-latest-userguide.html#sec_5_2_0)

10.190.4.12 double gazebo::physics::SurfaceParams::mu1

Dry friction coefficient in the primary friction direction as defined by the friction pyramid.

This is fdir1 if defined, otherwise, 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

10.190.4.13 double gazebo::physics::SurfaceParams::mu2

Dry friction coefficient in the second friction direction as defined by the friction pyramid.

This is fdir1 if defined, otherwise, 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

10.190.4.14 double gazebo::physics::SurfaceParams::slip1

Artificial contact slip in the primary friction direction.

See Also

See dContactSlip1 in http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.190.4.15 double gazebo::physics::SurfaceParams::slip2

Artificial contact slip in the secondary friction direction.

See Also

See dContactSlip2 in http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

The documentation for this class was generated from the following file:

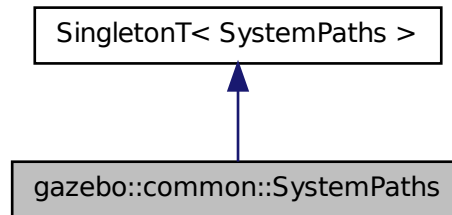
- **SurfaceParams.hh**

10.191 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.
- 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 **GetWorldPathExtension** ()

Returns the world path extension.

Public Attributes

- bool **gazeboPathsFromEnv**
*if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 941)*
- bool **modelPathsFromEnv**
*if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 941)*
- bool **ogrePathsFromEnv**
*if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 941)*
- bool **pluginPathsFromEnv**
*if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 942)*

Additional Inherited Members

10.191.1 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. 123) RenderSystem_GL.so et. al.
- SystemPaths::pluginPaths - plugin library paths for common::WorldPlugin

10.191.2 Member Function Documentation

10.191.2.1 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
----	--------------	----------------------

10.191.2.2 void gazebo::common::SystemPaths::AddModelPaths (const std::string & *_path*)

Add colon delimited paths to modelPaths.

Parameters

in	<i>_path</i>	the directory to add
----	--------------	----------------------

10.191.2.3 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
----	--------------	----------------------

10.191.2.4 void gazebo::common::SystemPaths::AddPluginPaths (const std::string & *_path*)

Add colon delimited paths to plugins.

Parameters

in	<i>_path</i>	the directory to add
----	--------------	----------------------

10.191.2.5 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
----	----------------	-------------------

10.191.2.6 void gazebo::common::SystemPaths::ClearGazeboPaths ()

clear out SystemPaths::gazeboPaths

10.191.2.7 void gazebo::common::SystemPaths::ClearModelPaths ()

clear out SystemPaths::modelPaths

10.191.2.8 void gazebo::common::SystemPaths::ClearOgrePaths ()

clear out SystemPaths::ogrePaths

10.191.2.9 void gazebo::common::SystemPaths::ClearPluginPaths ()

clear out SystemPaths::pluginPaths

10.191.2.10 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

10.191.2.11 `std::string gazebo::common::SystemPaths::FindFileURI (const std::string & _uri)`

Find a file or path using a URI.

Parameters

<code>in</code>	<code>_uri</code>	the uniform resource identifier
-----------------	-------------------	---------------------------------

Returns

Returns full path name to file

10.191.2.12 `const std::list<std::string>& gazebo::common::SystemPaths::GetGazeboPaths ()`

Get the gazebo install paths.

Returns

a list of paths

10.191.2.13 `std::string gazebo::common::SystemPaths::GetLogPath () const`

Get the log path.

Returns

the path

10.191.2.14 `const std::list<std::string>& gazebo::common::SystemPaths::GetModelPaths ()`

Get the model paths.

Returns

a list of paths

10.191.2.15 `const std::list<std::string>& gazebo::common::SystemPaths::GetOgrePaths ()`

Get the ogre install paths.

Returns

a list of paths

10.191.2.16 `const std::list<std::string>& gazebo::common::SystemPaths::GetPluginPaths ()`

Get the plugin paths.

Returns

a list of paths

10.191.2.17 `std::string gazebo::common::SystemPaths::GetWorldPathExtension ()`

Returns the world path extension.

Returns

Right now, it just returns "/worlds"

10.191.3 Member Data Documentation

10.191.3.1 `bool gazebo::common::SystemPaths::gazeboPathsFromEnv`

if true, call `UpdateGazeboPaths()` within **GetGazeboPaths()** (p. 941)

10.191.3.2 `bool gazebo::common::SystemPaths::modelPathsFromEnv`

if true, call `UpdateGazeboPaths()` within **GetGazeboPaths()** (p. 941)

10.191.3.3 `bool gazebo::common::SystemPaths::ogrePathsFromEnv`

if true, call `UpdateOgrePaths()` within **GetOgrePaths()** (p. 941)

10.191.3.4 `bool gazebo::common::SystemPaths::pluginPathsFromEnv`

if true, call `UpdatePluginPaths()` within **GetPluginPaths()** (p. 942)

The documentation for this class was generated from the following file:

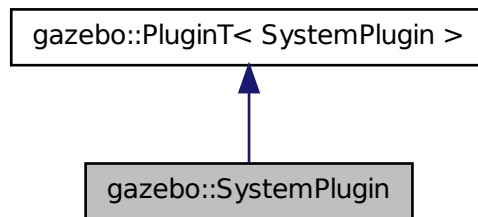
- **SystemPaths.hh**

10.192 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

10.192.1 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?

10.192.2 Constructor & Destructor Documentation

10.192.2.1 gazebo::SystemPlugin::SystemPlugin () [inline]

Constructor.

References gazebo::SYSTEM_PLUGIN, and gazebo::PluginT< SystemPlugin >::type.

10.192.2.2 virtual gazebo::SystemPlugin::~~SystemPlugin () [inline],[virtual]

Destructor.

10.192.3 Member Function Documentation

10.192.3.1 `virtual void gazebo::SystemPlugin::Init () [inline],[virtual]`

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.192.3.2 `virtual void gazebo::SystemPlugin::Load (int _argc = 0, char **_argv = NULL) [pure virtual]`

Load function.

Called before Gazebo is loaded. Must not block.

Parameters

<code>_argc</code>	Number of command line arguments.
<code>_argv</code>	Array of command line arguments.

10.192.3.3 `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**

10.193 gazebo::common::Time Class Reference

A **Time** (p. 944) 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 **~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==** (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.

10.193.1 Detailed Description

A **Time** (p. 944) class, can be used to hold wall- or sim-time. stored as sec and nano-sec.

10.193.2 Constructor & Destructor Documentation

10.193.2.1 `gazebo::common::Time::Time ()`

Constructors.

10.193.2.2 `gazebo::common::Time::Time (const Time & _time)`

Copy constructor.

Parameters

<code>in</code>	<code>time</code>	Time (p. 944) to copy
-----------------	-------------------	------------------------------

10.193.2.3 `gazebo::common::Time::Time (const struct timeval & _tv)`

Constructor.

Parameters

<code>in</code>	<code>_tv</code>	Time (p. 944) to initialize to
-----------------	------------------	---------------------------------------

10.193.2.4 `gazebo::common::Time::Time (const struct timespec & _tv)`

Constructor.

Parameters

<code>in</code>	<code>_tv</code>	Time (p. 944) to initialize to
-----------------	------------------	---------------------------------------

10.193.2.5 gazebo::common::Time::Time (int32_t _sec, int32_t _nsec)

Constructor.

Parameters

in	_sec	Seconds
in	_nsec	Nanoseconds

10.193.2.6 gazebo::common::Time::Time (double _time)

Constructor.

Parameters

in	_time	Time (p. 944) in double format sec.nsec
----	-------	--

10.193.2.7 virtual gazebo::common::Time::~~Time () [virtual]

Destructor.

10.193.3 Member Function Documentation

10.193.3.1 double gazebo::common::Time::Double () const

Get the time as a double.

Returns

Time (p. 944) as a double in seconds

10.193.3.2 float gazebo::common::Time::Float () const

Get the time as a float.

Returns

Time (p. 944) as a float in seconds

10.193.3.3 static const Time& gazebo::common::Time::GetWallTime () [static]

Get the wall time.

Returns

the current time

Referenced by Joint_TEST::SpawnJoint().

10.193.3.4 `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.

10.193.3.5 `static double gazebo::common::Time::MicToNano (double _ms) [inline],[static]`

Convert microseconds to nanoseconds.

Parameters

<code><i>_ms</i></code>	microseconds
-------------------------	--------------

Returns

nanoseconds

10.193.3.6 `static double gazebo::common::Time::MilToNano (double _ms) [inline],[static]`

Convert milliseconds to nanoseconds.

Parameters

<code>in</code>	<code><i>_ms</i></code>	milliseconds
-----------------	-------------------------	--------------

Returns

nanoseconds

10.193.3.7 `static Time gazebo::common::Time::MSleep (unsigned int _ms) [static]`

Millisecond sleep.

Parameters

<code>in</code>	<code><i>_ms</i></code>	milliseconds
-----------------	-------------------------	--------------

Returns

Time (p. 944) actually slept

Referenced by Joint_TEST::SpawnJoint().

10.193.3.8 `static Time gazebo::common::Time::NSleep (unsigned int _ns) [static]`

Nano sleep.

Parameters

in	<code>_ns</code>	nanoseconds
----	------------------	-------------

Returns

Time (p. 944) actually slept

10.193.3.9 `bool gazebo::common::Time::operator!=(const struct timeval & _tv) const`

Equal to operator.

Parameters

in	<code>_tv</code>	the time to compare to
----	------------------	------------------------

Returns

true if values are the same, false otherwise

10.193.3.10 `bool gazebo::common::Time::operator!=(const struct timespec & _tv) const`

Equal to operator.

Parameters

in	<code>_tv</code>	the time to compare to
----	------------------	------------------------

Returns

true if values are the same, false otherwise

10.193.3.11 `bool gazebo::common::Time::operator!=(const Time & _time) const`

Equal to operator.

Parameters

in	<code>_time</code>	the time to compare to
----	--------------------	------------------------

Returns

true if values are the same, false otherwise

10.193.3.12 `bool gazebo::common::Time::operator!=(double _time) const`

Equal to operator.

Parameters

in	<i>_time</i>	the time to compare to
----	--------------	------------------------

Returns

true if values are the same, false otherwise

10.193.3.13 **Time** gazebo::common::Time::operator* (const struct timeval & *_tv*) const

Multiplication operator.

Parameters

in	<i>_tv</i>	The scaling duration
----	------------	----------------------

Returns

Time (p. 944) instance

10.193.3.14 **Time** gazebo::common::Time::operator* (const struct timespec & *_tv*) const

Multiplication operator.

Parameters

in	<i>_tv</i>	the scaling duration
----	------------	----------------------

Returns

Time (p. 944) instance

10.193.3.15 **Time** gazebo::common::Time::operator* (const Time & *_time*) const

Multiplication operators.

Parameters

in	<i>_time</i>	the scaling factor
----	--------------	--------------------

Returns

a scaled **Time** (p. 944) instance

10.193.3.16 **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

10.193.3.17 `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

10.193.3.18 `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. 944) instance

10.193.3.19 `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. 944) instance

10.193.3.20 `Time gazebo::common::Time::operator+(const struct timespec & _tv) const`

Addition operators.

Parameters

in	<code>_tv</code>	the time to add
----	------------------	-----------------

Returns

a **Time** (p. 944) instance

10.193.3.21 **Time** gazebo::common::Time::operator+ (const Time & *_time*) const

Addition operators.

Parameters

in	<code>_time</code>	The time to add
----	--------------------	-----------------

Returns

a **Time** (p. 944) instance

10.193.3.22 **const Time&** gazebo::common::Time::operator+= (const struct timeval & *_tv*)

Addition assignment operator.

Parameters

in	<code>_tv</code>	the time to add
----	------------------	-----------------

Returns

a reference to this instance

10.193.3.23 **const Time&** gazebo::common::Time::operator+= (const struct timespec & *_tv*)

Addition assignment operator.

Parameters

in	<code>_tv</code>	the time to add
----	------------------	-----------------

Returns

a reference to this instance

10.193.3.24 **const Time&** gazebo::common::Time::operator+= (const Time & *_time*)

Addition assignemtn operator.

Parameters

in	<i>_time</i>	The time to add
----	--------------	-----------------

Returns

a **Time** (p. 944) instance

10.193.3.25 Time gazebo::common::Time::operator- (const struct timeval & *_tv*) const

Subtraction operator.

Parameters

in	<i>_tv</i>	The time to subtract
----	------------	----------------------

Returns

a **Time** (p. 944) instance

10.193.3.26 Time gazebo::common::Time::operator- (const struct timespec & *_tv*) const

Subtraction operator.

Parameters

in	<i>_tv</i>	The time to subtract
----	------------	----------------------

Returns

a **Time** (p. 944) instance

10.193.3.27 Time gazebo::common::Time::operator- (const Time & *_time*) const

Subtraction operator.

Parameters

in	<i>_time</i>	The time to subtract
----	--------------	----------------------

Returns

a **Time** (p. 944) instance

10.193.3.28 const Time& gazebo::common::Time::operator-= (const struct timeval & *_tv*)

Subtraction assignment operator.

Parameters

in	<code>_tv</code>	The time to subtract
----	------------------	----------------------

Returns

a **Time** (p. 944) instance

10.193.3.29 `const Time& gazebo::common::Time::operator-= (const struct timespec & _tv)`

Subtraction assignment operator.

Parameters

in	<code>_tv</code>	The time to subtract
----	------------------	----------------------

Returns

a **Time** (p. 944) instance

10.193.3.30 `const Time& gazebo::common::Time::operator-= (const Time & _time)`

Subtraction assignment operator.

Parameters

in	<code>_time</code>	The time to subtract
----	--------------------	----------------------

Returns

a reference to this instance

10.193.3.31 `Time gazebo::common::Time::operator/ (const struct timeval & _tv) const`

Division operator.

Parameters

in	<code>_tv</code>	a timeval divisor
----	------------------	-------------------

Returns

a **Time** (p. 944) instance

10.193.3.32 `Time gazebo::common::Time::operator/ (const struct timespec & _tv) const`

Division operator.

Parameters

<code>in</code>	<code>_tv</code>	a timespec divisor
-----------------	------------------	--------------------

Returns

a **Time** (p. 944) instance

10.193.3.33 **Time** gazebo::common::Time::operator/ (const Time & *time*) const

Division operator.

Parameters

<code>in</code>	<code>_time</code>	the divisor
-----------------	--------------------	-------------

Returns

a **Time** (p. 944) instance

10.193.3.34 **const Time&** gazebo::common::Time::operator/= (const struct timeval & *tv*)

Division assignment operator.

Parameters

<code>in</code>	<code>_tv</code>	a divisor
-----------------	------------------	-----------

Returns

a **Time** (p. 944) instance

10.193.3.35 **const Time&** gazebo::common::Time::operator/= (const struct timespec & *tv*)

Division assignment operator.

Parameters

<code>in</code>	<code>_tv</code>	a divisor
-----------------	------------------	-----------

Returns

a **Time** (p. 944) instance

10.193.3.36 **const Time&** gazebo::common::Time::operator/= (const Time & *time*)

Division assignment operator.

Parameters

<code>in</code>	<code>time</code>	the divisor
-----------------	-------------------	-------------

Returns

a **Time** (p. 944) instance

10.193.3.37 `bool gazebo::common::Time::operator< (const struct timeval & _tv) const`

Less than operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare with
-----------------	------------------	--------------------------

Returns

true if tv is shorter than this, false otherwise

10.193.3.38 `bool gazebo::common::Time::operator< (const struct timespec & _tv) const`

Less than operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare with
-----------------	------------------	--------------------------

Returns

true if tv is shorter than this, false otherwise

10.193.3.39 `bool gazebo::common::Time::operator< (const Time & _time) const`

Less than operator.

Parameters

<code>in</code>	<code>_time</code>	the time to compare with
-----------------	--------------------	--------------------------

Returns

true if time is shorter than this, false otherwise

10.193.3.40 `bool gazebo::common::Time::operator< (double _time) const`

Less than operator.

Parameters

in	<i>_time</i>	the time to compare with
----	--------------	--------------------------

Returns

true if time is shorter than this, false otherwise

10.193.3.41 `bool gazebo::common::Time::operator<= (const struct timeval & _tv) const`

Less than or equal to operator.

Parameters

in	<i>_tv</i>	the time to compare with
----	------------	--------------------------

Returns

true if tv is shorter than or equal to this, false otherwise

10.193.3.42 `bool gazebo::common::Time::operator<= (const struct timespec & _tv) const`

Less than or equal to operator.

Parameters

in	<i>_tv</i>	the time to compare with
----	------------	--------------------------

Returns

true if tv is shorter than or equal to this, false otherwise

10.193.3.43 `bool gazebo::common::Time::operator<= (const Time & _time) const`

Less than or equal to operator.

Parameters

in	<i>_time</i>	the time to compare with
----	--------------	--------------------------

Returns

true if time is shorter than or equal to this, false otherwise

10.193.3.44 `bool gazebo::common::Time::operator<= (double _time) const`

Less than or equal to operator.

Parameters

in	<i>_time</i>	the time to compare with
----	--------------	--------------------------

Returns

true if time is shorter than or equal to this, false otherwise

10.193.3.45 **Time& gazebo::common::Time::operator= (const struct timeval & *_tv*)**

Assignment operator.

Parameters

in	<i>_tv</i>	the new time
----	------------	--------------

Returns

a reference to this instance

10.193.3.46 **Time& gazebo::common::Time::operator= (const struct timespec & *_tv*)**

Assignment operator.

Parameters

in	<i>_tv</i>	the new time
----	------------	--------------

Returns

a reference to this instance

10.193.3.47 **Time& gazebo::common::Time::operator= (const Time & *_time*)**

Assignment operator.

Parameters

in	<i>_time</i>	the new time
----	--------------	--------------

Returns

a reference to this instance

10.193.3.48 **bool gazebo::common::Time::operator== (const struct timeval & *_tv*) const**

Equal to operator.

Parameters

in	<code>_tv</code>	the time to compare to
----	------------------	------------------------

Returns

true if values are the same, false otherwise

10.193.3.49 `bool gazebo::common::Time::operator==(const struct timespec & _tv) const`

Equal to operator.

Parameters

in	<code>_tv</code>	the time to compare to
----	------------------	------------------------

Returns

true if values are the same, false otherwise

10.193.3.50 `bool gazebo::common::Time::operator==(const Time & _time) const`

Equal to operator.

Parameters

in	<code>_time</code>	the time to compare to
----	--------------------	------------------------

Returns

true if values are the same, false otherwise

10.193.3.51 `bool gazebo::common::Time::operator==(double _time) const`

Equal to operator.

Parameters

in	<code>_time</code>	the time to compare to
----	--------------------	------------------------

Returns

true if values are the same, false otherwise

10.193.3.52 `bool gazebo::common::Time::operator>(const struct timeval & _tv) const`

Greater than operator.

Parameters

in	_tv	the time to compare with
----	-----	--------------------------

Returns

true if time is greater than this, false otherwise

10.193.3.53 `bool gazebo::common::Time::operator> (const struct timespec & _tv) const`

Greater than operator.

Parameters

in	_tv	the time to compare with
----	-----	--------------------------

Returns

true if time is greater than this, false otherwise

10.193.3.54 `bool gazebo::common::Time::operator> (const Time & _time) const`

Greater than operator.

Parameters

in	_time	the time to compare with
----	-------	--------------------------

Returns

true if time is greater than this, false otherwise

10.193.3.55 `bool gazebo::common::Time::operator> (double _time) const`

Greater than operator.

Parameters

in	_time	the time to compare with
----	-------	--------------------------

Returns

true if time is greater than this, false otherwise

10.193.3.56 `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

10.193.3.57 `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

10.193.3.58 `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

10.193.3.59 `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

10.193.3.60 `static double gazebo::common::Time::SecToNano (double _sec) [inline],[static]`

Convert seconds to nanoseconds.

Parameters

in	<code>_sec</code>	duration in seconds
----	-------------------	---------------------

Returns

nanoseconds

10.193.3.61 `void gazebo::common::Time::Set (int32_t _sec, int32_t _nsec)`

Set to sec and nsec.

Parameters

in	<code>_sec</code>	Seconds
in	<code>_nsec</code>	Nanoseconds

10.193.3.62 `void gazebo::common::Time::Set (double _seconds)`

Set to seconds.

Parameters

in	<code>_seconds</code>	Number of seconds
----	-----------------------	-------------------

10.193.3.63 `void gazebo::common::Time::SetToWallTime ()`

Set the time to the wall time.

10.193.3.64 `static Time gazebo::common::Time::Sleep (const common::Time & _time)` [static]

Sleep for the specified time.

Parameters

in	<code>_time</code>	Sleep time
----	--------------------	------------

Returns

Time (p. 944) actually slept

10.193.4 Friends And Related Function Documentation

10.193.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::common::Time & _time)` [friend]

Stream insertion operator.

Parameters

in	<code>_out</code>	the output stream
in	<code>_time</code>	time to write to the stream

Returns

the output stream

10.193.4.2 `std::istream& operator>> (std::istream & in, gazebo::common::Time & time)` [*friend*]

Stream extraction operator.

Parameters

<code>in</code>	<code><i>_in</i></code>	the input stream
<code>in</code>	<code><i>_time</i></code>	time to read from to the stream

Returns

the input stream

10.193.5 Member Data Documentation

10.193.5.1 `int32_t gazebo::common::Time::nsec`

Nanoseconds.

10.193.5.2 `int32_t gazebo::common::Time::sec`

Seconds.

10.193.5.3 `const Time gazebo::common::Time::Zero` [*static*]

A static zero time variable set to `common::Time(0, 0)`.

Referenced by `Joint_TEST::SpawnJoint()`.

The documentation for this class was generated from the following file:

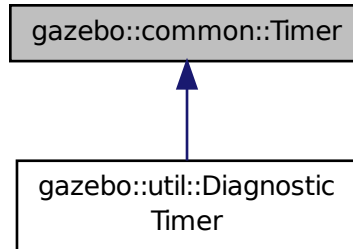
- **Time.hh**

10.194 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.

10.194.1 Detailed Description

A timer class, used to time things in real world walltime.

10.194.2 Constructor & Destructor Documentation

10.194.2.1 gazebo::common::Timer::Timer ()

Constructor.

10.194.2.2 `virtual gazebo::common::Timer::~~Timer () [virtual]`

Destructor.

10.194.3 Member Function Documentation

10.194.3.1 `Time gazebo::common::Timer::GetElapsed () const`

Get the elapsed time.

Returns

The time

10.194.3.2 `bool gazebo::common::Timer::GetRunning () const`

Returns true if the timer is running.

Returns

True if the timer has been started and not stopped.

10.194.3.3 `virtual void gazebo::common::Timer::Start () [virtual]`

Start the timer.

Reimplemented in `gazebo::util::DiagnosticTimer` (p. 284).

10.194.3.4 `virtual void gazebo::common::Timer::Stop () [virtual]`

Stop the timer.

Reimplemented in `gazebo::util::DiagnosticTimer` (p. 284).

10.194.4 Friends And Related Function Documentation

10.194.4.1 `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:

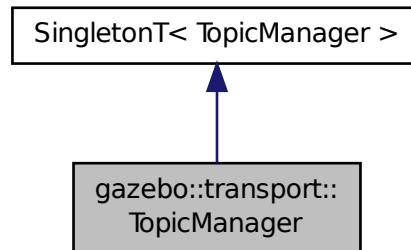
- `Timer.hh`

10.195 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. 587) 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. 240) a local **Publisher** (p. 671) to a remote **Subscriber** (p. 929).
- 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. 929) to a remote **Publisher** (p. 671).
- 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 unpause 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 **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

10.195.1 Detailed Description

Manages topics and their subscriptions.

10.195.2 Member Typedef Documentation

10.195.2.1 typedef std::map<std::string, std::list<NodePtr> > gazebo::transport::TopicManager::SubNodeMap

A map of string->list of **Node** (p. 587) pointers.

10.195.3 Member Function Documentation

10.195.3.1 void gazebo::transport::TopicManager::AddNode (**NodePtr** _node)

Add a node to the manager.

Parameters

in, out	_node	The node to be added
---------	-------	----------------------

10.195.3.2 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. 587) to process.
----	-------------	----------------------------------

10.195.3.3 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. 671)

References FindPublication(), GZ_ASSERT, gzthrow, SingletonT< T >::Instance(), NULL, and UpdatePublications().

10.195.3.4 void gazebo::transport::TopicManager::ClearBuffers ()

Clear all buffers.

10.195.3.5 void gazebo::transport::TopicManager::ConnectPubToSub (const std::string & *_topic*, const **SubscriptionTransportPtr** *_sublink*)

Connection (p. 240) a local **Publisher** (p. 671) to a remote **Subscriber** (p. 929).

Parameters

in	<i>_topic</i>	The topic to use
in	<i>_sublink</i>	The subscription transport object to use

10.195.3.6 void gazebo::transport::TopicManager::ConnectSubscribers (const std::string & *_topic*)

Connect all subscribers on a topic to known publishers.

Parameters

in	<i>_topic</i>	The topic to be connected
----	---------------	---------------------------

10.195.3.7 `void gazebo::transport::TopicManager::ConnectSubToPub (const msgs::Publish & _pub)`

Connect a local **Subscriber** (p. 929) to a remote **Publisher** (p. 671).

Parameters

<code>in</code>	<code><i>_pub</i></code>	The publish object to use
-----------------	--------------------------	---------------------------

10.195.3.8 `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

<code>in</code>	<code><i>_topic</i></code>	The topic to be disconnected
<code>in</code>	<code><i>_host</i></code>	The host to be disconnected
<code>in</code>	<code><i>_port</i></code>	The port to be disconnected

10.195.3.9 `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

<code>in</code>	<code><i>_topic</i></code>	The topic to be disconnected
<code>in</code>	<code><i>_host</i></code>	The host to be disconnected
<code>in</code>	<code><i>_port</i></code>	The port to be disconnected

10.195.3.10 `PublicationPtr gazebo::transport::TopicManager::FindPublication (const std::string & _topic)`

Find a publication object by topic.

Parameters

<code>in</code>	<code><i>_topic</i></code>	The topic to search for
-----------------	----------------------------	-------------------------

Returns

Pointer to the publication object, if found (can be null)

Referenced by `Advertise()`.

10.195.3.11 `void gazebo::transport::TopicManager::Fini ()`

Finalize the manager.

10.195.3.12 `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
-----	--------------------------	---

10.195.3.13 `void gazebo::transport::TopicManager::Init ()`

Initialize the manager.

10.195.3.14 `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

10.195.3.15 `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
----	---------------------	--

10.195.3.16 `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
----	-----------------------	---

10.195.3.17 `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. 671) instead of calling this function directly.

Parameters

in	<code>_topic</code>	Name of the topic
in	<code>_message</code>	The message to send.
in	<code>_cb</code>	Callback, used when the publish is completed.
in	<code>_id</code>	ID associated with the message.

10.195.3.18 `void gazebo::transport::TopicManager::RegisterTopicNamespace (const std::string & _name)`

Register a new topic namespace.

Parameters

in	<code>_name</code>	The name of the new namespace
----	--------------------	-------------------------------

10.195.3.19 `void gazebo::transport::TopicManager::RemoveNode (unsigned int _id)`

Remove a node by its id.

Parameters

in	<code>_id</code>	The ID of the node to be removed
----	------------------	----------------------------------

10.195.3.20 `SubscriberPtr gazebo::transport::TopicManager::Subscribe (const SubscribeOptions & _options)`

Subscribe to a topic.

Parameters

in	<code>_options</code>	The options to use for the subscription
----	-----------------------	---

Returns

Pointer to the newly created subscriber

10.195.3.21 `void gazebo::transport::TopicManager::Unadvertise (const std::string & _topic)`

Unadvertise a topic.

Parameters

in	<code>_topic</code>	The topic to be unadvertised
----	---------------------	------------------------------

10.195.3.22 `void gazebo::transport::TopicManager::Unsubscribe (const std::string & _topic, const NodePtr & _sub)`

Unsubscribe from a topic.

Use a **Subscriber** (p. 929) rather than calling this function directly

Parameters

in	<code>_topic</code>	The topic to unsubscribe from
in	<code>_sub</code>	The node to unsubscribe

10.195.3.23 **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

Referenced by Advertise().

The documentation for this class was generated from the following file:

- **TopicManager.hh**

10.196 gazebo::physics::TrajectoryInfo Struct Reference

```
#include <Actor.hh>
```

Public Attributes

- double **duration**
- double **endTime**
- unsigned int **id**
- double **startTime**
- bool **translated**
- std::string **type**

10.196.1 Member Data Documentation

10.196.1.1 double gazebo::physics::TrajectoryInfo::duration

10.196.1.2 double gazebo::physics::TrajectoryInfo::endTime

10.196.1.3 unsigned int gazebo::physics::TrajectoryInfo::id

10.196.1.4 double gazebo::physics::TrajectoryInfo::startTime

10.196.1.5 bool gazebo::physics::TrajectoryInfo::translated

10.196.1.6 `std::string gazebo::physics::TrajectoryInfo::type`

The documentation for this struct was generated from the following file:

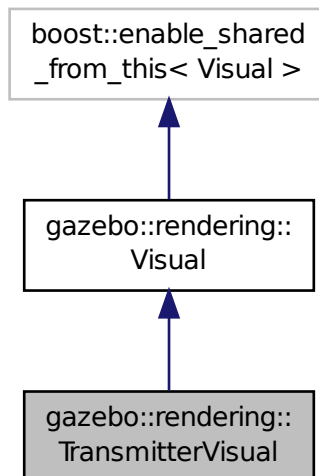
- **Actor.hh**

10.197 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

10.197.1 Detailed Description

Visualization for the wireless propagation data.

10.197.2 Constructor & Destructor Documentation

10.197.2.1 `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. 1034).
<code>in</code>	<code><i>_topicName</i></code>	Name of the topic that has laser data.

10.197.2.2 `virtual gazebo::rendering::TransmitterVisual::~~TransmitterVisual () [virtual]`

Destructor.

10.197.3 Member Function Documentation

10.197.3.1 `virtual void gazebo::rendering::TransmitterVisual::Load () [virtual]`

Documentation inherited from parent.

Reimplemented from `gazebo::rendering::Visual` (p. 1048).

10.197.3.2 `virtual void gazebo::rendering::TransmitterVisual::Update () [virtual]`

Function that runs on the OGRE thread to refresh the UI.

The documentation for this class was generated from the following file:

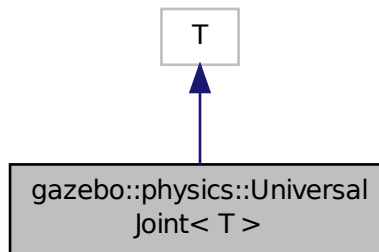
- `TransmitterVisual.hh`

10.198 gazebo::physics::UniversalJoint< T > Class Template Reference

A universal joint.

```
#include <physics/physics.hh>
```


Inheritance diagram for gazebo::physics::UniversalJoint< T >:



Public Member Functions

- **UniversalJoint** (**BasePtr** _parent)
Constructor.
- virtual \sim **UniversalJoint** ()
Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (sdf::ElementPtr _sdf)
*Load a **UniversalJoint** (p. 976).*

10.198.1 Detailed Description

```
template<class T>class gazebo::physics::UniversalJoint< T >
```

A universal joint.

10.198.2 Constructor & Destructor Documentation

10.198.2.1 `template<class T> gazebo::physics::UniversalJoint< T >::UniversalJoint (BasePtr _parent)`
[inline], [explicit]

Constructor.

Parameters

in	<code>_parent</code>	Parent link of the universal joint.
----	----------------------	-------------------------------------

10.198.2.2 `template<class T> virtual gazebo::physics::UniversalJoint< T >::~~UniversalJoint () [inline], [virtual]`

Destuctor.

10.198.3 Member Function Documentation

10.198.3.1 `template<class T> virtual unsigned int gazebo::physics::UniversalJoint< T >::GetAngleCount () const [inline], [virtual]`

10.198.3.2 `template<class T> virtual void gazebo::physics::UniversalJoint< T >::Load (sdf::ElementPtr _sdf) [inline], [virtual]`

Load a **UniversalJoint** (p. 976).

Parameters

in	_sdf	SDF values to load from.
----	------	--------------------------

Reimplemented in **gazebo::physics::SimbodyUniversalJoint** (p. 862).

The documentation for this class was generated from the following file:

- **UniversalJoint.hh**

10.199 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.

10.199.1 Detailed Description

Information for use in an update event.

10.199.2 Member Data Documentation

10.199.2.1 `common::Time gazebo::common::UpdateInfo::realTime`

Current real time.

10.199.2.2 `common::Time` gazebo::common::UpdateInfo::simTime

Current simulation time.

10.199.2.3 `std::string` gazebo::common::UpdateInfo::worldName

Name of the world.

The documentation for this class was generated from the following file:

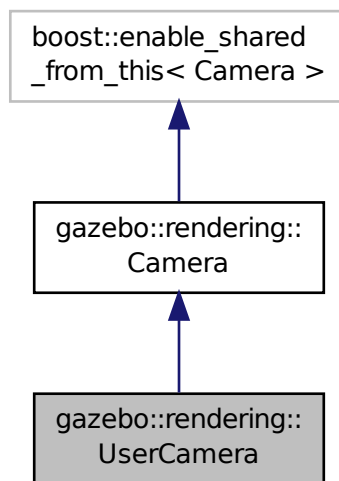
- **UpdateInfo.hh**

10.200 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.
- float **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.
- 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 **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

10.200.1 Detailed Description

A camera used for user visualization of a scene.

10.200.2 Constructor & Destructor Documentation

10.200.2.1 gazebo::rendering::UserCamera::UserCamera (const std::string & _name, ScenePtr _scene)

Constructor.

Parameters

in	<code>_name</code>	Name of the camera.
in	<code>_scene</code>	Scene (p. 728) to put the camera in.

10.200.2.2 virtual gazebo::rendering::UserCamera::~~UserCamera () [virtual]

Destructor.

10.200.3 Member Function Documentation

10.200.3.1 virtual void gazebo::rendering::UserCamera::AnimationComplete () [protected], [virtual]

Internal function used to indicate that an animation has completed.

Reimplemented from **gazebo::rendering::Camera** (p. 186).

10.200.3.2 `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	<code>_visual</code>	The visual to attach to.
in	<code>_inheritOrientation</code>	True if the camera should also rotate when the visual rotates.
in	<code>_minDist</code>	Minimum distance the camera can get to the visual.
in	<code>_maxDist</code>	Maximum distance the camera can get from the visual.

Returns

True if successfully attach to the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 187).

10.200.3.3 `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	<code>_value</code>	True to enable viewcontroller, False to disable.
----	---------------------	--

10.200.3.4 `void gazebo::rendering::UserCamera::Fini ()` [virtual]

Finalize.

Reimplemented from **gazebo::rendering::Camera** (p. 188).

10.200.3.5 `float gazebo::rendering::UserCamera::GetAvgFPS () const`

Get the average frames per second.

Returns

The average rendering frames per second

10.200.3.6 **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. 370).

10.200.3.7 `virtual unsigned int gazebo::rendering::UserCamera::GetImageHeight () const [virtual]`

Get the height of the image.

Returns

Image height

Reimplemented from **gazebo::rendering::Camera** (p. 191).

10.200.3.8 `virtual unsigned int gazebo::rendering::UserCamera::GetImageWidth () const [virtual]`

Get the width of the image.

Returns

Image width

Reimplemented from **gazebo::rendering::Camera** (p. 191).

10.200.3.9 `float gazebo::rendering::UserCamera::GetTriangleCount () const`

Get the triangle count.

Returns

The number of triangles currently being rendered.

10.200.3.10 `std::string gazebo::rendering::UserCamera::GetViewControllerTypeString ()`

Get current view controller type.

Returns

Type of the current view controller: "orbit", "fps"

10.200.3.11 `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

10.200.3.12 **VisualPtr** gazebo::rendering::UserCamera::GetVisual (const math::Vector2i & *_mousePos*) const

Get a visual at a mouse position.

Parameters

in	<i>_mousePos</i>	2D position of the mouse in pixels.
----	------------------	-------------------------------------

10.200.3.13 void gazebo::rendering::UserCamera::HandleKeyPressEvent (const std::string & *_key*)

Handle a key press.

Parameters

in	<i>_key</i>	The key pressed.
----	-------------	------------------

10.200.3.14 void gazebo::rendering::UserCamera::HandleKeyReleaseEvent (const std::string & *_key*)

Handle a key release.

Parameters

in	<i>_key</i>	The key released.
----	-------------	-------------------

10.200.3.15 void gazebo::rendering::UserCamera::HandleMouseEvent (const common::MouseEvent & *_evt*)

Handle a mouse event.

Parameters

in	<i>_evt</i>	The mouse event.
----	-------------	------------------

10.200.3.16 void gazebo::rendering::UserCamera::Init () [virtual]

Initialize.

Reimplemented from **gazebo::rendering::Camera** (p. 196).

10.200.3.17 void gazebo::rendering::UserCamera::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the user camera.

Parameters

in	<code>_sdf</code>	Parameters for the camera.
----	-------------------	----------------------------

Reimplemented from **gazebo::rendering::Camera** (p. 197).

10.200.3.18 void gazebo::rendering::UserCamera::Load () [virtual]

Generic load function.

Reimplemented from **gazebo::rendering::Camera** (p. 197).

10.200.3.19 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. 197)

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. 197).

10.200.3.20 void gazebo::rendering::UserCamera::MoveToVisual (VisualPtr *_visual*)

Move the camera to focus on a visual.

Parameters

in	<code>_visual</code>	Visual (p. 1034) to move the camera to.
----	----------------------	--

10.200.3.21 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.
----	--------------------------	---

10.200.3.22 virtual void gazebo::rendering::UserCamera::PostRender () [virtual]

Post render.

Reimplemented from **gazebo::rendering::Camera** (p. 197).

10.200.3.23 `void gazebo::rendering::UserCamera::Resize (unsigned int _w, unsigned int _h)`

Resize the camera.

Parameters

<code>in</code>	<code>_w</code>	Width of the camera image.
<code>in</code>	<code>_h</code>	Height of the camera image.

10.200.3.24 `void gazebo::rendering::UserCamera::SetFocalPoint (const math::Vector3 & _pt)`

Set the point the camera should orbit around.

Parameters

<code>in</code>	<code>_pt</code>	The focal point
-----------------	------------------	-----------------

10.200.3.25 `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

<code>in</code>	<code>_target</code>	The new rendering target.
-----------------	----------------------	---------------------------

Reimplemented from `gazebo::rendering::Camera` (p. 201).

10.200.3.26 `void gazebo::rendering::UserCamera::SetViewController (const std::string & _type)`

Set view controller.

Parameters

<code>in</code>	<code>_type</code>	The type of view controller: "orbit", "fps"
-----------------	--------------------	---

10.200.3.27 `void gazebo::rendering::UserCamera::SetViewController (const std::string & _type, const math::Vector3 & _pos)`

Set view controller.

Parameters

<code>in</code>	<code>_type</code>	The type of view controller: "orbit", "fps"
<code>in</code>	<code>_pos</code>	The initial pose of the camera.

10.200.3.28 `void gazebo::rendering::UserCamera::SetViewportDimensions (float _x, float _y, float _w, float _h)`

Set the dimensions of the viewport.

Parameters

in	<code>_x</code>	X position of the viewport.
in	<code>_y</code>	Y position of the viewport.
in	<code>_w</code>	Width of the viewport.
in	<code>_h</code>	Height of the viewport.

10.200.3.29 `virtual void gazebo::rendering::UserCamera::SetWorldPose (const math::Pose & _pose) [virtual]`

Set the pose in the world coordinate frame.

Parameters

in	<code>_pose</code>	New pose of the camera.
----	--------------------	-------------------------

Reimplemented from `gazebo::rendering::Camera` (p. 201).

10.200.3.30 `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	<code>_visual</code>	Visual (p. 1034) to track.
----	----------------------	-----------------------------------

Returns

True if the camera is now tracking the visual.

Reimplemented from `gazebo::rendering::Camera` (p. 203).

10.200.3.31 `virtual void gazebo::rendering::UserCamera::Update () [virtual]`

Render the camera.

Reimplemented from `gazebo::rendering::Camera` (p. 203).

The documentation for this class was generated from the following file:

- **UserCamera.hh**

10.201 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.

10.201.1 Detailed Description

Generic double x, y vector.

10.201.2 Constructor & Destructor Documentation

10.201.2.1 gazebo::math::Vector2d::Vector2d ()

Constructor.

10.201.2.2 gazebo::math::Vector2d::Vector2d (const double & `_x`, const double & `_y`)

Constructor.

Parameters

in	<code>_x</code>	value along x
in	<code>_y</code>	value along y

10.201.2.3 gazebo::math::Vector2d::Vector2d (const Vector2d & `_v`)

Copy constructor.

Parameters

in	<code>_v</code>	the value
----	-----------------	-----------

10.201.2.4 `virtual gazebo::math::Vector2d::~~Vector2d () [virtual]`

Destructor.

10.201.3 Member Function Documentation

10.201.3.1 `Vector2d gazebo::math::Vector2d::Cross (const Vector2d & _v) const`

Return the cross product of this vector and `_v`.

Parameters

<code>in</code>	<code>_v</code>	the vector
-----------------	-----------------	------------

Returns

the cross product

10.201.3.2 `double gazebo::math::Vector2d::Distance (const Vector2d & _pt) const`

Calc distance to the given point.

Parameters

<code>in</code>	<code>_pt</code>	The point to measure to
-----------------	------------------	-------------------------

Returns

the distance

10.201.3.3 `bool gazebo::math::Vector2d::IsFinite () const`

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.201.3.4 `void gazebo::math::Vector2d::Normalize ()`

Normalize the vector length.

10.201.3.5 `bool gazebo::math::Vector2d::operator!= (const Vector2d & _v) const`

Not equal to operator.

Returns

true if elements are of different values (tolerance 1e-6)

10.201.3.6 `const Vector2d gazebo::math::Vector2d::operator* (const Vector2d & _v) const`

Multiplication operators.

Parameters

<code>in</code>	<code>_v</code>	the vector
-----------------	-----------------	------------

Returns

the result

10.201.3.7 `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

10.201.3.8 `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

10.201.3.9 `const Vector2d& gazebo::math::Vector2d::operator*= (double _v)`

Multiplication assignment operator.

Parameters

<code>in</code>	<code>_v</code>	the scaling factor
-----------------	-----------------	--------------------

Returns

a scaled vector

10.201.3.10 `Vector2d gazebo::math::Vector2d::operator+ (const Vector2d & _v) const`

Addition operator.

Parameters

in	_v	vector to add
----	----	---------------

Returns

sum vector

10.201.3.11 `const Vector2d& gazebo::math::Vector2d::operator+= (const Vector2d & _v)`

Addition assignment operator.

Parameters

in	_v	the vector to add
----	----	-------------------

10.201.3.12 `Vector2d gazebo::math::Vector2d::operator- (const Vector2d & _v) const`

Subtraction operator.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

the subtracted vector

10.201.3.13 `const Vector2d& gazebo::math::Vector2d::operator-= (const Vector2d & _v)`

Subtraction assignment operator.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

this

10.201.3.14 `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

10.201.3.15 `const Vector2d gazebo::math::Vector2d::operator/ (double _v) const`

Division operator.

Parameters

<code>in</code>	<code>_v</code>	the value
-----------------	-----------------	-----------

Returns

a vector

10.201.3.16 `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

10.201.3.17 `const Vector2d& gazebo::math::Vector2d::operator/= (double _v)`

Division operator.

Parameters

<code>in</code>	<code>_v</code>	the divisor
-----------------	-----------------	-------------

Returns

a vector

10.201.3.18 `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

10.201.3.19 `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

10.201.3.20 `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)

10.201.3.21 `double gazebo::math::Vector2d::operator[] (unsigned int _index) const`

Array subscript operator.

Parameters

in	<code>_index</code>	the index
----	---------------------	-----------

Returns

the value, or 0 if `_index` is out of bounds

10.201.3.22 `void gazebo::math::Vector2d::Set (double _x, double _y)`

Set the contents of the vector.

Parameters

in	<code>_x</code>	value along x
in	<code>_y</code>	value along y

10.201.4 Friends And Related Function Documentation

10.201.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector2d & _pt)` [friend]

Stream extraction operator.

Parameters

in	<code>_out</code>	output stream
in	<code>_pt</code>	Vector2d (p. 987) to output

Returns

The stream

10.201.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector2d & _pt)` [friend]

Stream extraction operator.

Parameters

in	<code>_in</code>	input stream
in	<code>_pt</code>	Vector3 (p. 1004) to read values into

Returns

The stream

10.201.5 Member Data Documentation

10.201.5.1 `double gazebo::math::Vector2d::x`

x data

10.201.5.2 double gazebo::math::Vector2d::y

y data

The documentation for this class was generated from the following file:

- **Vector2d.hh**

10.202 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.

10.202.1 Detailed Description

Generic integer x, y vector.

10.202.2 Constructor & Destructor Documentation

10.202.2.1 gazebo::math::Vector2i::Vector2i ()

Constructor.

10.202.2.2 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

10.202.2.3 gazebo::math::Vector2i::Vector2i (const Vector2i & _pt)

Copy onstructor.

Parameters

in	<code>_pt</code>	a point
----	------------------	---------

10.202.2.4 virtual gazebo::math::Vector2i::~~Vector2i () [virtual]

Destructor.

10.202.3 Member Function Documentation

10.202.3.1 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

10.202.3.2 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

10.202.3.3 `bool gazebo::math::Vector2i::IsFinite () const`

See if a point is finite (e.g., not nan)

Returns

the result

10.202.3.4 `void gazebo::math::Vector2i::Normalize ()`

Normalize the vector length.

10.202.3.5 `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

10.202.3.6 `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

10.202.3.7 `const Vector2i gazebo::math::Vector2i::operator* (int _v) const`

Multiplication operator.

Parameters

<code>in</code>	<code>_v</code>	the scaling factor
-----------------	-----------------	--------------------

Returns

the result

10.202.3.8 `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

10.202.3.9 `const Vector2i& gazebo::math::Vector2i::operator*=(int _v)`

Multiplication operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

this

10.202.3.10 `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

10.202.3.11 `const Vector2i& gazebo::math::Vector2i::operator+=(const Vector2i & _v)`

Addition assignment operator.

Parameters

in	_v	the vector to add
----	----	-------------------

Returns

this

10.202.3.12 Vector2i gazebo::math::Vector2i::operator- (const Vector2i & _v) const

Subtraction operator.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

the result vector

10.202.3.13 const Vector2i& gazebo::math::Vector2i::operator-= (const Vector2i & _v)

Subtraction operators.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

this

10.202.3.14 const Vector2i gazebo::math::Vector2i::operator/ (const Vector2i & _v) const

Division operator.

Remarks

this is an element wise division.

Parameters

in	_v	the vector to divide
----	----	----------------------

Returns

the result

10.202.3.15 `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

10.202.3.16 `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

10.202.3.17 `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

10.202.3.18 Vector2i& gazebo::math::Vector2i::operator= (const Vector2i & *_v*)

Assignment operator.

Parameters

<i>in</i>	<i>_v</i>	the value
-----------	-----------	-----------

Returns

this

10.202.3.19 const Vector2i& gazebo::math::Vector2i::operator= (int *_value*)

Assignment operator.

Parameters

<i>in</i>	<i>_value</i>	the value for x and y
-----------	---------------	-----------------------

Returns

this

10.202.3.20 bool gazebo::math::Vector2i::operator== (const Vector2i & *_v*) const

Equality operator.

Parameters

	<i>_v</i>	the vector to compare with
--	-----------	----------------------------

Returns

true if component have the same values, false otherwise

10.202.3.21 int gazebo::math::Vector2i::operator[] (unsigned int *_index*) const

Array subscript operator.

Parameters

<i>in</i>	<i>_index</i>	the array index
-----------	---------------	-----------------

10.202.3.22 void gazebo::math::Vector2i::Set (int *_x*, int *_y*)

Set the contents of the vector.

Parameters

in	<code>_x</code>	value along x
in	<code>_y</code>	value along y

10.202.4 Friends And Related Function Documentation

10.202.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector2i & _pt)` [friend]

Stream insertion operator.

Parameters

in	<code>_out</code>	output stream
in	<code>pt</code>	Vector2i (p. 996) to output

Returns

the stream

10.202.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector2i & _pt)` [friend]

Stream extraction operator.

Parameters

in	<code>_in</code>	input stream
in	<code>pt</code>	Vector3 (p. 1004) to read values into

Returns

The stream

10.202.5 Member Data Documentation

10.202.5.1 `int gazebo::math::Vector2i::x`

x data

10.202.5.2 `int gazebo::math::Vector2i::y`

y data

The documentation for this class was generated from the following file:

- **Vector2i.hh**

10.203 gazebo::math::Vector3 Class Reference

The **Vector3** (p. 1004) 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.

10.203.1 Detailed Description

The **Vector3** (p. 1004) 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.

10.203.2 Constructor & Destructor Documentation

10.203.2.1 gazebo::math::Vector3::Vector3 ()

Constructor.

Referenced by operator-().

10.203.2.2 `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

10.203.2.3 `gazebo::math::Vector3::Vector3 (const Vector3 & _v)`

Copy constructor.

Parameters

in	<code>_v</code>	a vector
----	-----------------	----------

10.203.2.4 `virtual gazebo::math::Vector3::~~Vector3 () [virtual]`

Destructor.

10.203.3 Member Function Documentation

10.203.3.1 `void gazebo::math::Vector3::Correct () [inline]`

Corrects any nan values.

References x, y, and z.

Referenced by `gazebo::math::Pose::Correct()`.

10.203.3.2 `Vector3 gazebo::math::Vector3::Cross (const Vector3 & _pt) const`

Return the cross product of this vector and pt.

Returns

the product

10.203.3.3 `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

10.203.3.4 `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

10.203.3.5 `double gazebo::math::Vector3::Dot (const Vector3 & _pt) const`

Return the dot product of this vector and pt.

Returns

the product

10.203.3.6 `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

10.203.3.7 `Vector3 gazebo::math::Vector3::GetAbs () const`

Get the absolute value of the vector.

Returns

a vector with positive elements

10.203.3.8 `double gazebo::math::Vector3::GetDistToLine (const Vector3 & _pt1, const Vector3 & _pt2)`

Get distance to a line.

Parameters

<code>in</code>	<code>_pt1</code>	first point on the line
<code>in</code>	<code>_pt2</code>	second point on the line

Returns

the minimum distance from this point to the line

10.203.3.9 `double gazebo::math::Vector3::GetLength () const`

Returns the length (magnitude) of the vector \ return the length.

10.203.3.10 `double gazebo::math::Vector3::GetMax () const`

Get the maximum value in the vector.

Returns

the maximum element

10.203.3.11 `double gazebo::math::Vector3::GetMin () const`

Get the minimum value in the vector.

Returns

the minimum element

10.203.3.12 `static Vector3 gazebo::math::Vector3::GetNormal (const Vector3 & _v1, const Vector3 & _v2, const Vector3 & _v3) [static]`

Get a normal vector to a triangle.

Parameters

<code>in</code>	<code>_v1</code>	first vertex of the triangle
<code>in</code>	<code>_v2</code>	second vertex
<code>in</code>	<code>_v3</code>	third vertex

Returns

the normal

10.203.3.13 **Vector3** gazebo::math::Vector3::GetPerpendicular () const

Return a vector that is perpendicular to this one.

Returns

an orthogonal vector

10.203.3.14 **Vector3** gazebo::math::Vector3::GetRounded () const

Get a rounded version of this vector.

Returns

a rounded vector

10.203.3.15 **double** gazebo::math::Vector3::GetSquaredLength () const

Return the square of the length (magnitude) of the vector.

Returns

the squared length

10.203.3.16 **double** gazebo::math::Vector3::GetSum () const

Return the sum of the values.

Returns

the sum

10.203.3.17 **bool** gazebo::math::Vector3::IsFinite () const

See if a point is finite (e.g., not nan)

10.203.3.18 **Vector3** gazebo::math::Vector3::Normalize ()

Normalize the vector length.

Returns

unit length vector

10.203.3.19 **bool** gazebo::math::Vector3::operator!=(const Vector3 & _v) const

Not equal to operator.

Parameters

in	_v	The vector to compare against
----	----	-------------------------------

Returns

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.203.3.20 `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	
----	----	--

10.203.3.21 `Vector3 gazebo::math::Vector3::operator* (double _v) const`

Multiplication operators.

Parameters

in	_v	the scaling factor
----	----	--------------------

Returns

a scaled vector

10.203.3.22 `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

10.203.3.23 `const Vector3& gazebo::math::Vector3::operator*=(double _v)`

Multiplication operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

this

10.203.3.24 `Vector3 gazebo::math::Vector3::operator+(const Vector3 & _v) const`

Addition operator.

Parameters

<code>in</code>	<code>_v</code>	vector to add
-----------------	-----------------	---------------

Returns

the sum vector

10.203.3.25 `const Vector3& gazebo::math::Vector3::operator+=(const Vector3 & _v)`

Addition assignment operator.

Parameters

<code>in</code>	<code>_v</code>	vector to add
-----------------	-----------------	---------------

10.203.3.26 `Vector3 gazebo::math::Vector3::operator-() const` `[inline]`

Negation operator.

Returns

negative of this vector

References Vector3(), x, y, and z.

10.203.3.27 `Vector3 gazebo::math::Vector3::operator-(const Vector3 & _pt) const` `[inline]`

Subtraction operators.

Parameters

<code>in</code>	<code>_pt</code>	a vector to subtract
-----------------	------------------	----------------------

Returns

a vector

References Vector3(), x, y, and z.

10.203.3.28 `const Vector3& gazebo::math::Vector3::operator-= (const Vector3 & _pt)`

Subtraction operators.

Parameters

<code>in</code>	<code>_pt</code>	subtrahend
-----------------	------------------	------------

10.203.3.29 `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

10.203.3.30 `const Vector3 gazebo::math::Vector3::operator/ (double _v) const`

Division operator.

Remarks

this is an element wise division

Returns

a vector

10.203.3.31 `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

10.203.3.32 `const Vector3& gazebo::math::Vector3::operator/= (double _v)`

Division operator.

Remarks

this is an element wise division

Returns

this

10.203.3.33 `Vector3& gazebo::math::Vector3::operator= (const Vector3 & _v)`

Assignment operator.

Parameters

<code>in</code>	<code>_v</code>	a new value
-----------------	-----------------	-------------

Returns

this

10.203.3.34 `Vector3& gazebo::math::Vector3::operator= (double _value)`

Assignment operator.

Parameters

<code>in</code>	<code>_value</code>	assigned to all elements
-----------------	---------------------	--------------------------

Returns

this

10.203.3.35 `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

10.203.3.36 `double gazebo::math::Vector3::operator[] (unsigned int index) const`

[] operator

10.203.3.37 `Vector3 gazebo::math::Vector3::Round ()`

Round to near whole number, return the result.

Returns

the result

10.203.3.38 `void gazebo::math::Vector3::Round (int _precision)`

Round all values to `_precision` decimal places.

Parameters

<code>in</code>	<code>_precision</code>	the decimal places
-----------------	-------------------------	--------------------

10.203.3.39 `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

References x, y, and z.

10.203.3.40 `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
-----------------	-----------------	-----------------------------

10.203.3.41 `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
-----------------	-----------------	-----------------------------

10.203.4 Friends And Related Function Documentation

10.203.4.1 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

10.203.4.2 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. 1004) to output

Returns

the stream

10.203.4.3 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

10.203.5 Member Data Documentation

10.203.5.1 const Vector3 gazebo::math::Vector3::One [static]

math::Vector3(1, 1, 1)

10.203.5.2 const Vector3 gazebo::math::Vector3::UnitX [static]

math::Vector3(1, 0, 0)

10.203.5.3 `const Vector3 gazebo::math::Vector3::UnitY` [static]

`math::Vector3(0, 1, 0)`

10.203.5.4 `const Vector3 gazebo::math::Vector3::UnitZ` [static]

`math::Vector3(0, 0, 1)`

10.203.5.5 `double gazebo::math::Vector3::x`

X location.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `operator-()`, `gazebo::math::Quaternion::RotateVector()`, `Set()`, and `gazebo::physics::SimbodyBoxShape::SetSize()`.

10.203.5.6 `double gazebo::math::Vector3::y`

Y location.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `operator-()`, `gazebo::math::Quaternion::RotateVector()`, `Set()`, and `gazebo::physics::SimbodyBoxShape::SetSize()`.

10.203.5.7 `double gazebo::math::Vector3::z`

Z location.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `operator-()`, `gazebo::math::Quaternion::RotateVector()`, `Set()`, and `gazebo::physics::SimbodyBoxShape::SetSize()`.

10.203.5.8 `const Vector3 gazebo::math::Vector3::Zero` [static]

`math::Vector3(0, 0, 0)`

The documentation for this class was generated from the following file:

- **Vector3.hh**

10.204 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 \sim **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.

10.204.1 Detailed Description

double Generic x, y, z, w vector

10.204.2 Constructor & Destructor Documentation

10.204.2.1 gazebo::math::Vector4::Vector4 ()

Constructor.

10.204.2.2 gazebo::math::Vector4::Vector4 (const double & _x, const double & _y, const double & _z, const double & _w)

Constructor with component values.

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

10.204.2.3 gazebo::math::Vector4::Vector4 (const Vector4 & _v)

Copy constructor.

Parameters

in	_v	vector
----	----	--------

10.204.2.4 virtual gazebo::math::Vector4::~~Vector4 () [virtual]

Destructor.

10.204.3 Member Function Documentation

10.204.3.1 double gazebo::math::Vector4::Distance (const Vector4 & _pt) const

Calc distance to the given point.

Parameters

in	_pt	the point
----	-----	-----------

Returns

the distance

10.204.3.2 double gazebo::math::Vector4::GetLength () const

Returns the length (magnitude) of the vector.

10.204.3.3 double gazebo::math::Vector4::GetSquaredLength () const

Return the square of the length (magnitude) of the vector.

Returns

the length

10.204.3.4 bool gazebo::math::Vector4::IsFinite () const

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.204.3.5 void gazebo::math::Vector4::Normalize ()

Normalize the vector length.

10.204.3.6 `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

10.204.3.7 `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

10.204.3.8 `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`

10.204.3.9 `const Vector4 gazebo::math::Vector4::operator*(double _v) const`

Multiplication operators.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

a scaled vector

10.204.3.10 `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

10.204.3.11 `const Vector4& gazebo::math::Vector4::operator*=(double _v)`

Multiplication assignment operator.

Parameters

in	_v	scaling factor
----	----	----------------

Returns

this

10.204.3.12 `Vector4 gazebo::math::Vector4::operator+(const Vector4 & _v) const`

Addition operator.

Parameters

in	_v	the vector to add
----	----	-------------------

Returns

a sum vector

10.204.3.13 `const Vector4& gazebo::math::Vector4::operator+=(const Vector4 & _v)`

Addition operator.

Parameters

in	_v	the vector to add
----	----	-------------------

Returns

this vector

10.204.3.14 **Vector4** gazebo::math::Vector4::operator- (const Vector4 & _v) const

Subtraction operator.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

a vector

10.204.3.15 **const Vector4&** gazebo::math::Vector4::operator-= (const Vector4 & _v)

Subtraction assignment operators.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

this vector

10.204.3.16 **const Vector4** gazebo::math::Vector4::operator/ (const Vector4 & _v) const

Division assignment operator.

Remarks

Performs element wise division, which has limited use.

Parameters

in	_v	the vector to perform element wise division with
----	----	--

Returns

a result vector

10.204.3.17 `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

10.204.3.18 `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

10.204.3.19 `const Vector4& gazebo::math::Vector4::operator/=(double _v)`

Division operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

a vector

10.204.3.20 `Vector4& gazebo::math::Vector4::operator=(const Vector4 & _v)`

Assignment operator.

Parameters

in	<code>_v</code>	the vector
----	-----------------	------------

Returns

a reference to this vector

10.204.3.21 `Vector4& gazebo::math::Vector4::operator=(double _value)`

Assignment operator.

Parameters

in	<code>_value</code>	
----	---------------------	--

10.204.3.22 `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

10.204.3.23 `double gazebo::math::Vector4::operator[] (unsigned int _index) const`

Array subscript operator.

Parameters

in	<code>_index</code>	
----	---------------------	--

10.204.3.24 `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

10.204.4 Friends And Related Function Documentation

10.204.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector4 & _pt)` [friend]

Stream insertion operator.

Parameters

<code>in</code>	<code>_out</code>	output stream
<code>in</code>	<code>_pt</code>	Vector4 (p. 1018) to output

Returns

The stream

10.204.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector4 & _pt)` [friend]

Stream extraction operator.

Parameters

<code>in</code>	<code>_in</code>	input stream
<code>in</code>	<code>_pt</code>	Vector4 (p. 1018) to read values into

Returns

the stream

10.204.5 Member Data Documentation

10.204.5.1 `double gazebo::math::Vector4::w`

W value.

10.204.5.2 `double gazebo::math::Vector4::x`

X value.

10.204.5.3 `double gazebo::math::Vector4::y`

Y value.

10.204.5.4 `double gazebo::math::Vector4::z`

Z value.

The documentation for this class was generated from the following file:

- **Vector4.hh**

10.205 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.

10.205.1 Detailed Description

Handle video encoding and decoding using libavcodec.

10.205.2 Constructor & Destructor Documentation

10.205.2.1 gazebo::common::Video::Video ()

Constructor.

10.205.2.2 virtual gazebo::common::Video::~~Video () [virtual]

Destructor.

10.205.3 Member Function Documentation

10.205.3.1 int gazebo::common::Video::GetHeight () const

Get the height of the video in pixels.

Returns

the height

10.205.3.2 `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. 389) in which the frame is stored
------------------	--------------------------	--

Returns

false if HAVE_FFmpeg is not defined, true otherwise

10.205.3.3 `int gazebo::common::Video::GetWidth () const`

Get the width of the video in pixels.

Returns

the width

10.205.3.4 `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:

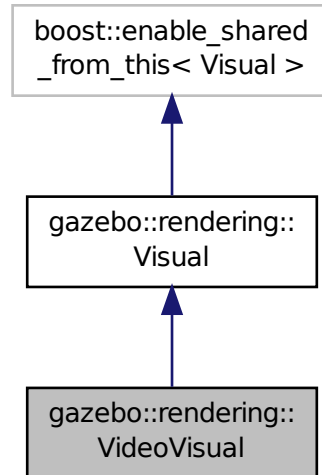
- **Video.hh**

10.206 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

10.206.1 Detailed Description

A visual element that displays a video as a texture.

10.206.2 Constructor & Destructor Documentation

10.206.2.1 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.

10.206.2.2 virtual gazebo::rendering::VideoVisual::~~VideoVisual () [virtual]

Destructor.

The documentation for this class was generated from the following file:

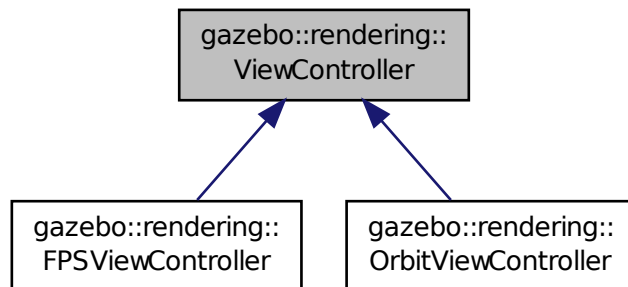
- **VideoVisual.hh**

10.207 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. 179).*

Protected Attributes

- **UserCameraPtr camera**

Pointer to the camera to control.

- bool **enabled**

True if enabled.

- std::string **typeString**

Type of view controller.

10.207.1 Detailed Description

Base class for view controllers.

10.207.2 Constructor & Destructor Documentation

10.207.2.1 gazebo::rendering::ViewController::ViewController (UserCameraPtr `_camera`)

Constructor.

Parameters

in	<code>_camera</code>	The user camera to controll.
----	----------------------	------------------------------

10.207.2.2 virtual gazebo::rendering::ViewController::~~ViewController () [virtual]

Destructor.

10.207.3 Member Function Documentation

10.207.3.1 std::string gazebo::rendering::ViewController::GetTypeString () const

Get the type of view controller.

Returns

The view controller type string.

10.207.3.2 virtual void gazebo::rendering::ViewController::HandleKeyPressEvent (const std::string & `_key`) [pure virtual]

Handle a key press event.

Parameters

in	_key	The key that was pressed.
----	------	---------------------------

Implemented in **gazebo::rendering::OrbitViewController** (p. 619), and **gazebo::rendering::FPSViewController** (p. 339).

10.207.3.3 virtual void gazebo::rendering::ViewController::HandleKeyReleaseEvent (const std::string & _key) [pure virtual]

Handle a key release event.

Parameters

in	_key	The key that was released.
----	------	----------------------------

Implemented in **gazebo::rendering::OrbitViewController** (p. 619), and **gazebo::rendering::FPSViewController** (p. 339).

10.207.3.4 virtual void gazebo::rendering::ViewController::HandleMouseEvent (const common::MouseEvent & _event) [pure virtual]

Handle a mouse event.

Parameters

in	_event	The mouse position.
----	--------	---------------------

Implemented in **gazebo::rendering::OrbitViewController** (p. 619), and **gazebo::rendering::FPSViewController** (p. 339).

10.207.3.5 virtual void gazebo::rendering::ViewController::Init () [pure virtual]

Initialize the view controller.

Implemented in **gazebo::rendering::OrbitViewController** (p. 619), and **gazebo::rendering::FPSViewController** (p. 340).

10.207.3.6 virtual void gazebo::rendering::ViewController::Init (const math::Vector3 & _focalPoint) [virtual]

Initialize with a focus point.

Parameters

in	_focalPoint	The point to look at.
----	-------------	-----------------------

Reimplemented in **gazebo::rendering::OrbitViewController** (p. 619).

10.207.3.7 void gazebo::rendering::ViewController::SetEnabled (bool _value)

Set whether the controller is enabled.

Parameters

in	<code>_value</code>	True if the controller is enabled.
----	---------------------	------------------------------------

10.207.3.8 `virtual void gazebo::rendering::ViewController::Update ()` [pure virtual]

Update the controller, which should update the position of the **Camera** (p. 179).

Implemented in **`gazebo::rendering::OrbitViewController`** (p. 620), and **`gazebo::rendering::FPSViewController`** (p. 340).

10.207.4 Member Data Documentation

10.207.4.1 `UserCameraPtr gazebo::rendering::ViewController::camera` [protected]

Pointer to the camera to control.

10.207.4.2 `bool gazebo::rendering::ViewController::enabled` [protected]

True if enabled.

10.207.4.3 `std::string gazebo::rendering::ViewController::typeString` [protected]

Type of view controller.

The documentation for this class was generated from the following file:

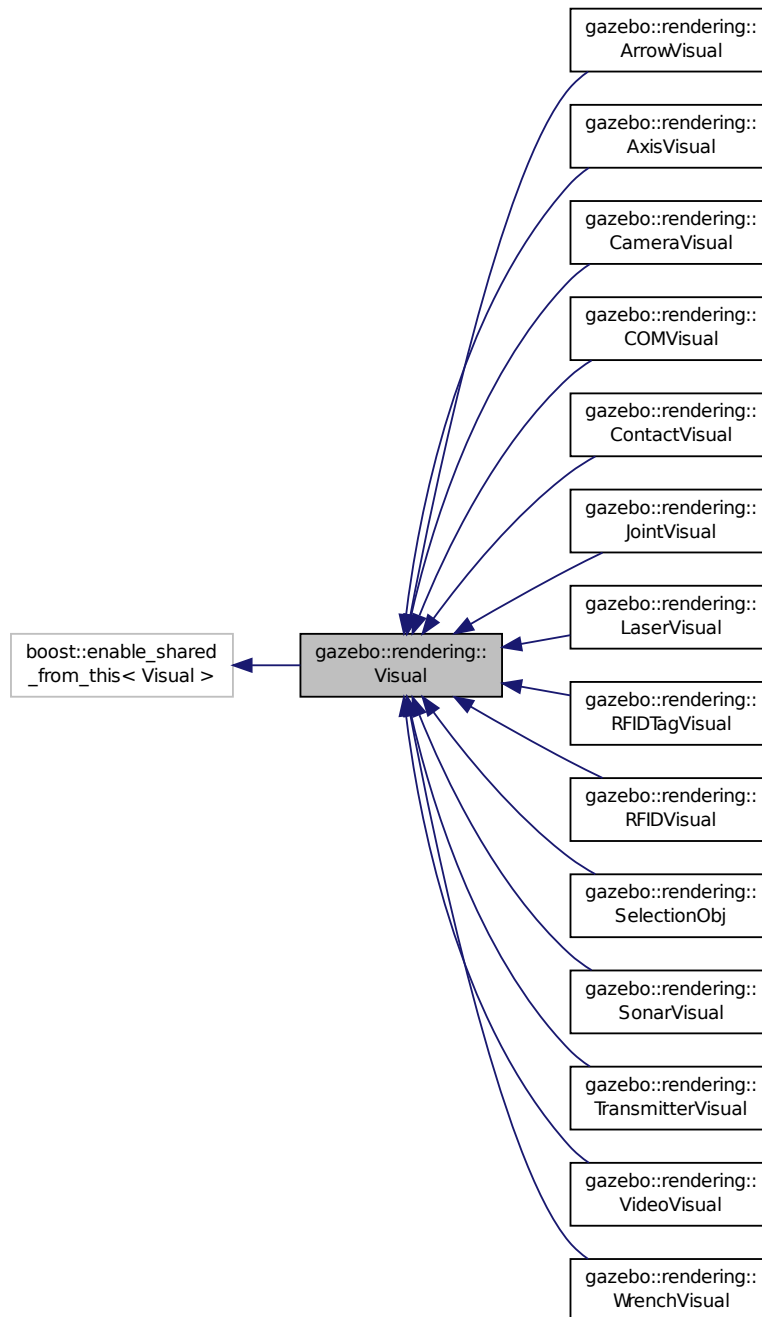
- **`ViewController.hh`**

10.208 `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.

- 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. 123).*
- `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 **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. 123).*

Protected Attributes

- **VisualPtr parent**
Parent visual.
- **ScenePtr scene**
Pointer to the visual's scene.
- `Ogre::SceneNode * sceneNode`
*Pointer to the visual's scene node in **Ogre** (p. 123).*

10.208.1 Detailed Description

A renderable object.

10.208.2 Constructor & Destructor Documentation

10.208.2.1 `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).

10.208.2.2 `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. 728) containing the visual.
in	<code>_useRTShader</code>	True if the visual should use the real-time shader system (RTShader).

10.208.2.3 `virtual gazebo::rendering::Visual::~Visual () [virtual]`

Destructor.

10.208.3 Member Function Documentation

10.208.3.1 `void gazebo::rendering::Visual::AttachAxes ()`

Attach visualization axes.

10.208.3.2 `void gazebo::rendering::Visual::AttachLineVertex (DynamicLines * _line, unsigned int _index)`

Attach a vertex of a line to the position of the visual.

Parameters

in	<code>_line</code>	Line to attach to this visual.
in	<code>_index</code>	Index of the line vertex to attach.

10.208.3.3 `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	<code>_meshName</code>	Name of the mesh.
in	<code>_subMesh</code>	Name of the submesh. Empty string to use all submeshes.
in	<code>_centerSubmesh</code>	True to center a submesh.
in	<code>_objName</code>	Name of the attached Object to put the mesh onto.

10.208.3.4 `void gazebo::rendering::Visual::AttachObject (Ogre::MovableObject * _obj)`

Attach a renerable object to the visual.

Parameters

in	<code>_obj</code>	A movable object to attach to the visual.
----	-------------------	---

10.208.3.5 `void gazebo::rendering::Visual::AttachVisual (VisualPtr _vis)`

Attach a visual to this visual.

Parameters

in	<code>_vis</code>	Visual (p. 1034) to attach.
----	-------------------	------------------------------------

10.208.3.6 `void gazebo::rendering::Visual::ClearParent ()`

Clear parents.

10.208.3.7 `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. 1034).
in	<code>_newParent</code>	Parent of the cloned Visual (p. 1034).

Returns

The visual.

10.208.3.8 **DynamicLines*** gazebo::rendering::Visual::CreateDynamicLine (RenderOpType *_type* = RENDERING_LINE_STRIP)

Add a line to the visual.

Parameters

in	<i>_type</i>	The type of line to make.
----	--------------	---------------------------

Returns

A pointer to the new dynamic line.

10.208.3.9 void gazebo::rendering::Visual::DeleteDynamicLine (DynamicLines * *_line*)

Delete a dynamic line.

Parameters

in	<i>_line</i>	Pointer to the line to delete.
----	--------------	--------------------------------

10.208.3.10 void gazebo::rendering::Visual::DetachObjects ()

Detach all objects.

10.208.3.11 void gazebo::rendering::Visual::DetachVisual (VisualPtr *_vis*)

Detach a visual.

Parameters

in	<i>_vis</i>	Visual (p. 1034) to detach.
----	-------------	------------------------------------

10.208.3.12 void gazebo::rendering::Visual::DetachVisual (const std::string & *_name*)

Detach a visual.

Parameters

in	<i>_name</i>	Name of the visual to detach.
----	--------------	-------------------------------

10.208.3.13 void gazebo::rendering::Visual::DisableTrackVisual ()

Disable tracking of a visual.

10.208.3.14 void gazebo::rendering::Visual::EnableTrackVisual (VisualPtr *_vis*)

Set one visual to track/follow another.

Parameters

<code>in</code>	<code>_vis</code>	Visual (p. 1034) to track.
-----------------	-------------------	-----------------------------------

10.208.3.15 `void gazebo::rendering::Visual::Fini ()`

Helper for the destructor.

10.208.3.16 `unsigned int gazebo::rendering::Visual::GetAttachedObjectCount () const`

Return the number of attached movable objects.

Returns

The number of attached movable objects.

10.208.3.17 `math::Box gazebo::rendering::Visual::GetBoundingBox () const`

Get the bounding box for the visual.

Returns

The bounding box in world coordinates.

10.208.3.18 `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. 1043).

Parameters

<code>in</code>	<code><i>index</i></code>	Index of the child to retrieve.
-----------------	---------------------------	---------------------------------

Returns

Pointer to the child visual, NULL if index is invalid.

10.208.3.19 `unsigned int gazebo::rendering::Visual::GetChildCount ()`

Get the number of attached visuals.

Returns

The number of children.

10.208.3.20 `uint32_t gazebo::rendering::Visual::GetId () const`

Get the id associated with this visual.

10.208.3.21 `std::string gazebo::rendering::Visual::GetMaterialName () const`

Get the name of the material.

Returns

The name of the visual applied to this visual.

10.208.3.22 `std::string gazebo::rendering::Visual::GetMeshName () const`

The name of the mesh set in the visual's SDF.

Returns

Name of the mesh.

10.208.3.23 `std::string gazebo::rendering::Visual::GetName () const`

Get the name of the visual.

Returns

The name of the visual.

10.208.3.24 `std::string gazebo::rendering::Visual::GetNormalMap () const`

Get the normal map.

Returns

The name of the normal map material.

10.208.3.25 `VisualPtr gazebo::rendering::Visual::GetParent () const`

Get the parent visual, if one exists.

Returns

Pointer to the parent visual, NULL if no parent.

10.208.3.26 `math::Pose gazebo::rendering::Visual::GetPose () const`

Get the pose of the visual.

Returns

The **Visual** (p. 1034)'s pose.

10.208.3.27 `math::Vector3 gazebo::rendering::Visual::GetPosition () const`

Get the position of the visual.

Returns

The visual's position.

10.208.3.28 `VisualPtr gazebo::rendering::Visual::GetRootVisual ()`

Get the root visual.

Returns

The root visual, which is one level below the world visual.

10.208.3.29 `math::Quaternion gazebo::rendering::Visual::GetRotation () const`

Get the rotation of the visual.

Returns

The visual's rotation.

10.208.3.30 `math::Vector3 gazebo::rendering::Visual::GetScale ()`

Get the scale.

Returns

The scaling factor.

10.208.3.31 `ScenePtr gazebo::rendering::Visual::GetScene () const`

Get current.

Returns

Pointer to the scene.

10.208.3.32 `Ogre::SceneNode* gazebo::rendering::Visual::GetSceneNode () const`

Return the scene Node of this visual entity.

Returns

The **Ogre** (p. 123) scene node.

10.208.3.33 `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".

10.208.3.34 `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.

10.208.3.35 `float gazebo::rendering::Visual::GetTransparency ()`

Get the transparency.

Returns

The transparency.

10.208.3.36 `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. 1236)

GZ_VISIBILITY_GUI (p. 1236)

GZ_VISIBILITY_SELECTABLE (p. 1236)

10.208.3.37 `bool gazebo::rendering::Visual::GetVisible () const`

Get whether the visual is visible.

Returns

True if the visual is visible.

10.208.3.38 `math::Pose gazebo::rendering::Visual::GetWorldPose () const`

Get the global pose of the node.

Returns

The pose in the world coordinate frame.

10.208.3.39 `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.
----	--------------------	----------------------------

10.208.3.40 `void gazebo::rendering::Visual::Init ()`

Helper for the constructor.

10.208.3.41 `void gazebo::rendering::Visual::InsertMesh (const std::string & _meshName, const std::string & _subMesh = "", bool _centerSubmesh = false)`

Insert a mesh into **Ogre** (p. 123).

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.

10.208.3.42 `static void gazebo::rendering::Visual::InsertMesh (const common::Mesh * _mesh, const std::string & _subMesh = "", bool _centerSubmesh = false) [static]`

Insert a mesh into **Ogre** (p. 123).

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.

10.208.3.43 `bool gazebo::rendering::Visual::IsPlane () const`

Return true if the visual is a plane.

Returns

True if a plane.

10.208.3.44 `bool gazebo::rendering::Visual::IsStatic () const`

Return true if the visual is a static geometry.

Returns

True if the visual is static.

10.208.3.45 `void gazebo::rendering::Visual::Load (sdf::ElementPtr _sdf)`

Load the visual with a set of parameters.

Parameters

<code>in</code>	<code>_sdf</code>	Load from an SDF element.
-----------------	-------------------	---------------------------

10.208.3.46 `virtual void gazebo::rendering::Visual::Load () [virtual]`

Load the visual with default parameters.

Reimplemented in `gazebo::rendering::SelectionObj` (p. 78), `gazebo::rendering::ArrowVisual` (p. 145), `gazebo::rendering::SonarVisual` (p. 896), `gazebo::rendering::TransmitterVisual` (p. 976), and `gazebo::rendering::Axis-Visual` (p. 150).

10.208.3.47 `void gazebo::rendering::Visual::LoadFromMsg (ConstVisualPtr & _msg)`

Load from a message.

Parameters

<code>in</code>	<code>_msg</code>	A visual message.
-----------------	-------------------	-------------------

10.208.3.48 `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.

10.208.3.49 `void gazebo::rendering::Visual::MakeStatic ()`

Make the visual objects static renderables.

10.208.3.50 `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><i>_pose</i></code>	Pose the visual will end at.
<code>in</code>	<code><i>_time</i></code>	Time it takes the visual to move to the pose.

10.208.3.51 `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><i>_poses</i></code>	Series of poses the visual will move to.
<code>in</code>	<code><i>_time</i></code>	Time it takes the visual to move to the pose.
<code>in</code>	<code><i>_onComplete</i></code>	Callback used when the move is complete.

10.208.3.52 `void gazebo::rendering::Visual::RemovePlugin (const std::string & _name)`

Remove a running plugin.

Parameters

<code><i>_name</i></code>	The unique name of the plugin to remove
---------------------------	---

10.208.3.53 `void gazebo::rendering::Visual::SetAmbient (const common::Color & _color)`

Set the ambient color of the visual.

Parameters

<code>in</code>	<code><i>_color</i></code>	The ambient color.
-----------------	----------------------------	--------------------

10.208.3.54 `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.
-----------------	------------------------------	-------------------------

10.208.3.55 `void gazebo::rendering::Visual::SetDiffuse (const common::Color & _color)`

Set the diffuse color of the visual.

Parameters

<code>in</code>	<code>_color</code>	Set the diffuse color.
-----------------	---------------------	------------------------

10.208.3.56 `virtual void gazebo::rendering::Visual::SetEmissive (const common::Color & _color)` [virtual]

Set the emissive value.

Parameters

<code>in</code>	<code>_color</code>	The emissive color.
-----------------	---------------------	---------------------

Reimplemented in `gazebo::rendering::LaserVisual` (p. 448).

10.208.3.57 `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>_highlighted</code>	True to enable the highlighting.
-----------------	---------------------------	----------------------------------

10.208.3.58 `void gazebo::rendering::Visual::SetId (uint32_t _id)`

Set the id associated with this visual.

10.208.3.59 `void gazebo::rendering::Visual::SetMaterial (const std::string & _materialName, bool _unique = true)`

Set the material.

Parameters

<code>in</code>	<code>_materialName</code>	The name of the material.
<code>in</code>	<code>_unique</code>	True to make the material unique, which allows the material to change without changing materials that originally had the same name.

10.208.3.60 `void gazebo::rendering::Visual::SetName (const std::string & _name)`

Set the name of the visual.

Parameters

<code>in</code>	<code>_name</code>	Name of the visual
-----------------	--------------------	--------------------

10.208.3.61 void gazebo::rendering::Visual::SetNormalMap (const std::string & *_nmap*)

Set the normal map.

Parameters

in	<i>_nmap</i>	Name of the normal map material.
----	--------------	----------------------------------

10.208.3.62 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.
----	--------------	-----------------------------

10.208.3.63 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.
----	-------------	------------------------------------

10.208.3.64 void gazebo::rendering::Visual::SetRibbonTrail (bool *_value*, const common::Color & *_initialColor*, const common::Color & *_changeColor*)

True on or off a ribbon trail.

Parameters

in	<i>_value</i>	True to enable ribbon trail.
in	<i>_initialColor</i>	The initial color of the ribbon trail.
in	<i>_changeColor</i>	Color to change too as the trail grows.

10.208.3.65 void gazebo::rendering::Visual::SetRotation (const math::Quaternion & *_rot*)

Set the rotation of the visual.

Parameters

in	<i>_rot</i>	The rotation of the visual.
----	-------------	-----------------------------

10.208.3.66 void gazebo::rendering::Visual::SetScale (const math::Vector3 & *_scale*)

Set the scale.

Parameters

in	<i>_scale</i>	The scaling factor for the visual.
----	---------------	------------------------------------

10.208.3.67 void gazebo::rendering::Visual::SetScene (ScenePtr *_scene*)

Set current scene.

Parameters

in	<i>_scene</i>	Pointer to the scene.
----	---------------	-----------------------

10.208.3.68 void gazebo::rendering::Visual::SetShaderType (const std::string & *_type*)

Set the shader type for the visual's material.

Parameters

in	<i>_type</i>	Shader type string: "vertex", "pixel", "normal_map_object_space", "normal_map_tangent_space".
----	--------------	---

10.208.3.69 void gazebo::rendering::Visual::SetSkeletonPose (const msgs::PoseAnimation & *_pose*)

Set animation skeleton pose.

Parameters

in	<i>_pose</i>	Skelton message
----	--------------	-----------------

10.208.3.70 void gazebo::rendering::Visual::SetSpecular (const common::Color & *_color*)

Set the specular color of the visual.

Parameters

in	<i>_color</i>	Specular color.
----	---------------	-----------------

10.208.3.71 void gazebo::rendering::Visual::SetTransparency (float *_trans*)

Set the transparency.

Parameters

in	<i>_trans</i>	The transparency, between 0 and 1 where 0 is no transparency.
----	---------------	---

10.208.3.72 void gazebo::rendering::Visual::SetVisibilityFlags (uint32_t *_flags*)

Set visibility flags for this visual and all children.

Parameters

in	<i>_flags</i>	The visibility flags.
----	---------------	-----------------------

See Also

GZ_VISIBILITY_ALL (p. 1236)

GZ_VISIBILITY_GUI (p. 1236)

GZ_VISIBILITY_SELECTABLE (p. 1236)

10.208.3.73 void gazebo::rendering::Visual::SetVisible (bool *_visible*, bool *_cascade* = true)

Set whether the visual is visible.

Parameters

in	<i>_visible</i>	set this node visible.
in	<i>_cascade</i>	setting this parameter in children too.

10.208.3.74 void gazebo::rendering::Visual::SetWireframe (bool *_show*)

Enable or disable wireframe for this visual.

Parameters

in	<i>_show</i>	True to enable wireframe for this visual.
----	--------------	---

10.208.3.75 void gazebo::rendering::Visual::SetWorldPose (const math::Pose *_pose*)

Set the world pose of the visual.

Parameters

in	<i>_pose</i>	Pose of the visual in the world coordinate frame.
----	--------------	---

10.208.3.76 void gazebo::rendering::Visual::SetWorldPosition (const math::Vector3 & *_pos*)

Set the world linear position of the visual.

Parameters

in	<i>_pose</i>	Position in the world coordinate frame.
----	--------------	---

10.208.3.77 void gazebo::rendering::Visual::SetWorldRotation (const math::Quaternion & *_rot*)

Set the world orientation of the visual.

Parameters

<code>in</code>	<code>_rot</code>	Rotation in the world coordinate frame.
-----------------	-------------------	---

10.208.3.78 `void gazebo::rendering::Visual::ShowBoundingBox ()`

Display the bounding box visual.

10.208.3.79 `void gazebo::rendering::Visual::ShowCollision (bool _show)`

Display the collision visuals.

Parameters

<code>in</code>	<code>_show</code>	True to show visuals labeled as collision objects.
-----------------	--------------------	--

10.208.3.80 `void gazebo::rendering::Visual::ShowCOM (bool _show)`

Display Center of Mass visuals.

Parameters

<code>in</code>	<code>_show</code>	True to show center of mass visualizations.
-----------------	--------------------	---

10.208.3.81 `void gazebo::rendering::Visual::ShowJoints (bool _show)`

Display joint visuals.

Parameters

<code>in</code>	<code>_show</code>	True to show joint visualizations.
-----------------	--------------------	------------------------------------

10.208.3.82 `void gazebo::rendering::Visual::ShowSkeleton (bool _show)`

Display the skeleton visuals.

Parameters

<code>in</code>	<code>_show</code>	True to show skeleton visuals.
-----------------	--------------------	--------------------------------

10.208.3.83 `void gazebo::rendering::Visual::ToggleVisible ()`

Toggle whether this visual is visible.

10.208.3.84 `void gazebo::rendering::Visual::Update ()`

Update the visual.

10.208.3.85 void gazebo::rendering::Visual::UpdateFromMsg (ConstVisualPtr & _msg)

Update a visual based on a message.

Parameters

in	<code>_msg</code>	The visual message.
----	-------------------	---------------------

10.208.4 Member Data Documentation

10.208.4.1 VisualPtr gazebo::rendering::Visual::parent [protected]

Parent visual.

10.208.4.2 ScenePtr gazebo::rendering::Visual::scene [protected]

Pointer to the visual's scene.

10.208.4.3 Ogre::SceneNode* gazebo::rendering::Visual::sceneNode [protected]

Pointer to the visual's scene node in **Ogre** (p. 123).

The documentation for this class was generated from the following file:

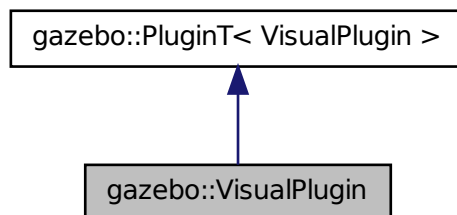
- **Visual.hh**

10.209 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

10.209.1 Detailed Description

A plugin loaded within the gzserver on startup.

See [reference](#).

10.209.2 Constructor & Destructor Documentation

10.209.2.1 gazebo::VisualPlugin::VisualPlugin () [inline]

References gazebo::PluginT< VisualPlugin >::type, and gazebo::VISUAL_PLUGIN.

10.209.3 Member Function Documentation

10.209.3.1 virtual void gazebo::VisualPlugin::Init () [inline],[virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.209.3.2 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	<code>_visual</code>	Pointer the Visual Object.
in	<code>_sdf</code>	Pointer the the SDF element of the plugin.

10.209.3.3 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**

10.210 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. 123) 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.

10.210.1 Detailed Description

Class to manage render windows.

10.210.2 Constructor & Destructor Documentation

10.210.2.1 gazebo::rendering::WindowManager::WindowManager ()

Constructor.

10.210.2.2 virtual gazebo::rendering::WindowManager::~~WindowManager () [virtual]

Destructor.

10.210.3 Member Function Documentation

10.210.3.1 `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.

10.210.3.2 `void gazebo::rendering::WindowManager::Fini ()`

Shutdown all the windows.

10.210.3.3 `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.

10.210.3.4 `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.

10.210.3.5 `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.

10.210.3.6 void gazebo::rendering::WindowManager::Moved (uint32_t *_id*)

Tells **Ogre** (p. 123) the window has moved, and needs updating.

Parameters

in	<i>_id</i>	ID of the window.
----	------------	-------------------

10.210.3.7 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.

10.210.3.8 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**

10.211 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.
- void **Init** (const **math::Box** &*_box*)

Builds the wireframe line list.

- void **SetVisible** (bool *_visible*)

Set the visibility of the box.

10.211.1 Detailed Description

Draws a wireframe box.

10.211.2 Constructor & Destructor Documentation

10.211.2.1 gazebo::rendering::WireBox::WireBox (VisualPtr *_parent*, const math::Box & *_box*) [explicit]

Constructor.

Parameters

<i>in</i>	<i>_box</i>	Dimension of the box to draw.
-----------	-------------	-------------------------------

10.211.2.2 gazebo::rendering::WireBox::~~WireBox ()

Destructor.

10.211.3 Member Function Documentation

10.211.3.1 void gazebo::rendering::WireBox::Init (const math::Box & *_box*)

Builds the wireframe line list.

Parameters

<i>in</i>	<i>_box</i>	Box to build a wireframe from.
-----------	-------------	--------------------------------

10.211.3.2 void gazebo::rendering::WireBox::SetVisible (bool *_visible*)

Set the visibility of the box.

Parameters

<i>in</i>	<i>_visible</i>	True to make the box visible, False to hide.
-----------	-----------------	--

The documentation for this class was generated from the following file:

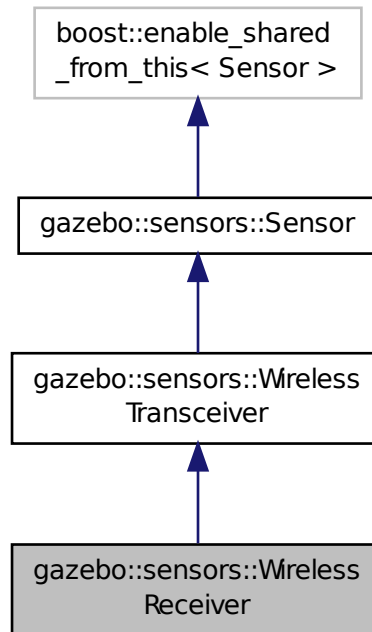
- **WireBox.hh**

10.212 gazebo::sensors::WirelessReceiver Class Reference

Sensor (p. 751) class for receiving wireless signals.

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::WirelessReceiver:



Public Member Functions

- **WirelessReceiver** ()
Constructor.
- virtual **~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

10.212.1 Detailed Description

Sensor (p. 751) class for receiving wireless signals.

10.212.2 Constructor & Destructor Documentation

10.212.2.1 gazebo::sensors::WirelessReceiver::WirelessReceiver ()

Constructor.

10.212.2.2 virtual gazebo::sensors::WirelessReceiver::~WirelessReceiver () [virtual]

Constructor.

10.212.3 Member Function Documentation

10.212.3.1 virtual void gazebo::sensors::WirelessReceiver::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1065).

10.212.3.2 double gazebo::sensors::WirelessReceiver::GetMaxFreqFiltered () const

Returns the maximum frequency filtered (MHz).

Returns

Reception frequency (MHz).

10.212.3.3 double gazebo::sensors::WirelessReceiver::GetMinFreqFiltered () const

Returns the minimum frequency filtered (MHz).

Returns

Reception frequency (MHz).

10.212.3.4 double gazebo::sensors::WirelessReceiver::GetSensitivity () const

Returns the receiver sensitivity (dBm).

Returns

Receiver sensitivity (dBm).

10.212.3.5 virtual void gazebo::sensors::WirelessReceiver::Init () [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1065).

10.212.3.6 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. 1065).

The documentation for this class was generated from the following file:

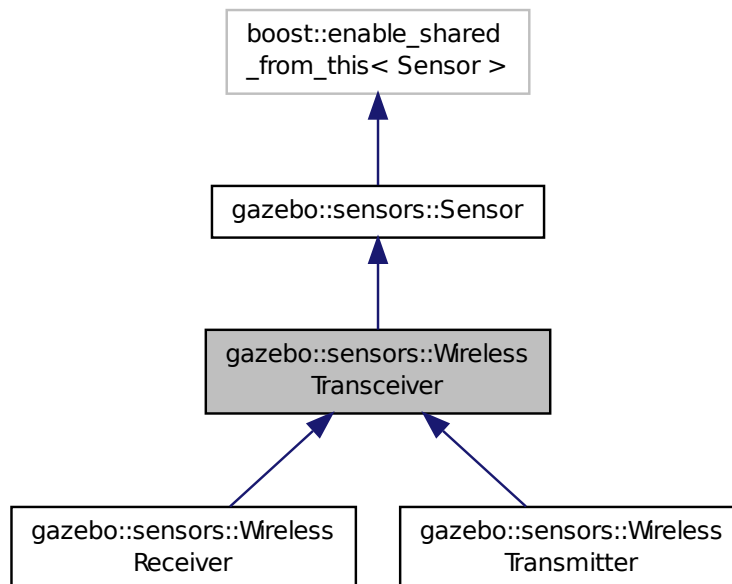
- **WirelessReceiver.hh**

10.213 gazebo::sensors::WirelessTransceiver Class Reference

Sensor (p. 751) 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. 751) reference pose.

Additional Inherited Members

10.213.1 Detailed Description

Sensor (p. 751) class for receiving wireless signals.

10.213.2 Constructor & Destructor Documentation

10.213.2.1 gazebo::sensors::WirelessTransceiver::WirelessTransceiver ()

Constructor.

10.213.2.2 gazebo::sensors::WirelessTransceiver::~~WirelessTransceiver ()

Constructor.

10.213.3 Member Function Documentation

10.213.3.1 virtual void gazebo::sensors::WirelessTransceiver::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 755).

Reimplemented in **gazebo::sensors::WirelessReceiver** (p. 1062).

10.213.3.2 double gazebo::sensors::WirelessTransceiver::GetGain () const

Returns the antenna's gain of the receiver (dBi).

Returns

Antenna's gain of the receiver (dBi).

10.213.3.3 double gazebo::sensors::WirelessTransceiver::GetPower () const

Returns the receiver power (dBm).

Returns

Receiver power (dBm).

10.213.3.4 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. 757).

10.213.3.5 virtual void gazebo::sensors::WirelessTransceiver::Init () [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 758).

Reimplemented in **gazebo::sensors::WirelessTransmitter** (p. 1069), and **gazebo::sensors::WirelessReceiver** (p. 1063).

10.213.3.6 virtual void gazebo::sensors::WirelessTransceiver::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. 759).

Reimplemented in `gazebo::sensors::WirelessTransmitter` (p. 1069), and `gazebo::sensors::WirelessReceiver` (p. 1063).

10.213.4 Member Data Documentation

10.213.4.1 `double gazebo::sensors::WirelessTransceiver::gain` [protected]

Antenna's gain of the receiver (dBi).

10.213.4.2 `boost::weak_ptr<physics::Link> gazebo::sensors::WirelessTransceiver::parentEntity` [protected]

Parent entity which the sensor is attached to.

10.213.4.3 `double gazebo::sensors::WirelessTransceiver::power` [protected]

Receiver's power (dBm).

10.213.4.4 `transport::PublisherPtr gazebo::sensors::WirelessTransceiver::pub` [protected]

Publisher to publish propagation model data.

10.213.4.5 `math::Pose gazebo::sensors::WirelessTransceiver::referencePose` [protected]

`Sensor` (p. 751) reference pose.

The documentation for this class was generated from the following file:

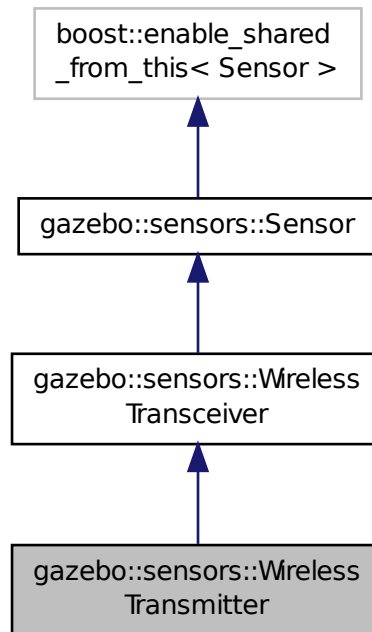
- `WirelessTransceiver.hh`

10.214 `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 dev 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 void **UpdateImpl** (bool _force)

This gets overwritten by derived sensor types.

Protected Attributes

- double **freq**

Reception frequency (MHz).

10.214.1 Detailed Description

Transmitter to send wireless signals.

10.214.2 Constructor & Destructor Documentation

10.214.2.1 gazebo::sensors::WirelessTransmitter::WirelessTransmitter ()

Constructor.

10.214.2.2 virtual gazebo::sensors::WirelessTransmitter::~~WirelessTransmitter () [virtual]

Destructor.

10.214.3 Member Function Documentation

10.214.3.1 std::string gazebo::sensors::WirelessTransmitter::GetESSID () const

Returns the Service Set Identifier (network name).

Returns

Service Set Identifier (network name).

10.214.3.2 double gazebo::sensors::WirelessTransmitter::GetFreq () const

Returns reception frequency (MHz).

Returns

Reception frequency (MHz).

10.214.3.3 `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).

10.214.3.4 `virtual void gazebo::sensors::WirelessTransmitter::Init () [virtual]`

Initialize the sensor.

Reimplemented from `gazebo::sensors::WirelessTransceiver` (p. 1065).

10.214.3.5 `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. 1065).

10.214.3.6 `virtual void 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
-----------------	---------------------	--

Reimplemented from `gazebo::sensors::Sensor` (p. 760).

10.214.4 Member Data Documentation

10.214.4.1 `double gazebo::sensors::WirelessTransmitter::freq [protected]`

Reception frequency (MHz).

10.214.4.2 `const double gazebo::sensors::WirelessTransmitter::ModelStdDesv [static]`

Std desv of the Gaussian random variable used in the propagation model.

10.214.4.3 `const double gazebo::sensors::WirelessTransmitter::NEmpty` [static]

Constant used in the propagation model when there are no obstacles between transmitter and receiver.

10.214.4.4 `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:

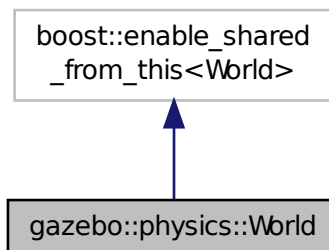
- **WirelessTransmitter.hh**

10.215 gazebo::physics::World Class Reference

The world provides access to all other object within a simulated environment.

```
#include <physics/physics.hh>
```

Inheritance diagram for gazebo::physics::World:



Public Member Functions

- **World** (const std::string &_name="")
Constructor.
- **~World** ()
Destructor.
- void **Clear** ()
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. 293) based on a name.*
- **EntityPtr GetEntityBelowPoint** (const math::Vector3 &_pt)
Get the nearest entity below a point.
- **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 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.
- **EntityPtr GetSelectedEntity** () const
*Get the selected **Entity** (p. 293).*
- 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. 949).*
- **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. 293) 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 **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 **StepWorld** (int _steps)
 - Step callback.*
 - 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. 303) on the **physics::Link** (p. 455) in **World::Update**.*

10.215.1 Detailed Description

The world provides access to all other object within a simulated environment.

The **World** (p. 1070) is the container for all models and their components (links, joints, sensors, plugins, etc), and **World-Plugin** (p. 1082) instances. Many core function are also handled in the **World** (p. 1070), including physics update, model updates, and message processing.

10.215.2 Constructor & Destructor Documentation

10.215.2.1 `gazebo::physics::World::World (const std::string & _name = " ") [explicit]`

Constructor.

Constructor for the **World** (p. 1070). Must specify a unique name.

Parameters

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

10.215.2.2 `gazebo::physics::World::~~World ()`

Destructor.

10.215.3 Member Function Documentation

10.215.3.1 `void gazebo::physics::World::Clear ()`

Remove all entities from the world.

10.215.3.2 `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.

10.215.3.3 `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.

10.215.3.4 `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.
-----------------	-----------------------------	------------------------------------

10.215.3.5 void gazebo::physics::World::Fini ()

Finalize the world.

Call this function to tear-down the world.

10.215.3.6 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

in	<i>_name</i>	The name of the Model (p. 537) to find.
----	--------------	--

Returns

A pointer to the entity, or NULL if no entity was found.

10.215.3.7 bool gazebo::physics::World::GetEnablePhysicsEngine () [inline]

check if physics engine is enabled/disabled.

Parameters

	<i>True</i>	if the physics engine is enabled.
--	-------------	-----------------------------------

10.215.3.8 EntityPtr gazebo::physics::World::GetEntity (const std::string & _name)

Get a pointer to an **Entity** (p. 293) based on a name.

This function is the same as GetByName, but limits the search to only Entities.

Parameters

in	<i>_name</i>	The name of the Entity (p. 293) to find.
----	--------------	---

Returns

A pointer to the **Entity** (p. 293), or NULL if no **Entity** (p. 293) was found.

10.215.3.9 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

in	<i>_pt</i>	The 3D point to search below
----	------------	------------------------------

Returns

A pointer to nearest **Entity** (p. 293), NULL if none is found.

10.215.3.10 ModelPtr gazebo::physics::World::GetModel (unsigned int *_index*) const

Get a model based on an index.

Get a **Model** (p. 537) using an index, where index must be greater than zero and less than **World::GetModelCount()** (p. 1075)

Parameters

<i>in</i>	<i>_index</i>	The index of the model [0..GetModelCount)
-----------	---------------	---

Returns

A pointer to the **Model** (p. 537). NULL if *_index* is invalid.

10.215.3.11 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

<i>in</i>	<i>_name</i>	The name of the Model (p. 537) to find.
-----------	--------------	--

Returns

A pointer to the **Model** (p. 537), or NULL if no model was found.

10.215.3.12 ModelPtr gazebo::physics::World::GetModelBelowPoint (const math::Vector3 & *_pt*)

Get the nearest model below a point.

This function makes use of **World::GetEntityBelowPoint** (p. 1074).

Parameters

<i>in</i>	<i>_pt</i>	The 3D point to search below.
-----------	------------	-------------------------------

Returns

A pointer to nearest **Model** (p. 537), NULL if none is found.

10.215.3.13 unsigned int gazebo::physics::World::GetModelCount () const

Get the number of models.

Returns

The number of models in the **World** (p. 1070).

10.215.3.14 Model_V gazebo::physics::World::GetModels () const

Get a list of all the models.

Returns

A list of all the Models in the world.

10.215.3.15 std::string gazebo::physics::World::GetName () const

Get the name of the world.

Returns

The name of the world.

10.215.3.16 common::Time gazebo::physics::World::GetPauseTime () const

Get the amount of time simulation has been paused.

Returns

The pause time.

10.215.3.17 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.

10.215.3.18 common::Time gazebo::physics::World::GetRealTime () const

Get the real time (elapsed time).

Returns

The real time.

10.215.3.19 `bool gazebo::physics::World::GetRunning () const`

Return the running state of the world.

Returns

True if the world is running.

10.215.3.20 `EntityPtr gazebo::physics::World::GetSelectedEntity () const`

Get the selected **Entity** (p. 293).

The selected entity is set via the GUI.

Returns

A point to the **Entity** (p. 293), NULL if nothing is selected.

10.215.3.21 `boost::mutex* gazebo::physics::World::GetSetWorldPoseMutex () const` `[inline]`

Get the set world pose mutex.

Returns

Pointer to the mutex.

10.215.3.22 `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. 949).

Returns

The current simulation time

10.215.3.23 `common::SphericalCoordinatesPtr gazebo::physics::World::GetSphericalCoordinates () const`

Return the spherical coordinates converter.

Returns

Pointer to the spherical coordinates converter.

10.215.3.24 `common::Time gazebo::physics::World::GetStartTime () const`

Get the wall time simulation was started.

Returns

The start time.

10.215.3.25 `void gazebo::physics::World::Init ()`

Initialize the world.

This is called after Load.

10.215.3.26 `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).
-----------------	---------------------------	--

10.215.3.27 `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.
-----------------	-------------------	-------------------------------

10.215.3.28 `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.
-----------------	-------------------------	---------------------------------------

10.215.3.29 `bool gazebo::physics::World::IsLoaded () const`

Return true if the world has been loaded.

Returns

True if **World::Load** (p. 1079) has completed.

10.215.3.30 `bool gazebo::physics::World::IsPaused () const`

Returns the state of the simulation true if paused.

Returns

True if paused.

10.215.3.31 `void gazebo::physics::World::Load (sdf::ElementPtr _sdf)`

Load the world using SDF parameters.

Load a world from and SDF pointer.

Parameters

<code>in</code>	<code>_sdf</code>	SDF parameters.
-----------------	-------------------	-----------------

10.215.3.32 `void gazebo::physics::World::LoadPlugin (const std::string & _filename, const std::string & _name, sdf::ElementPtr _sdf)`

Load a plugin.

Parameters

<code>in</code>	<code>_filename</code>	The filename of the plugin.
<code>in</code>	<code>_name</code>	A unique name for the plugin.
<code>in</code>	<code>_sdf</code>	The SDF to pass into the plugin.

10.215.3.33 `void gazebo::physics::World::PrintEntityTree ()`

Print **Entity** (p. 293) tree.

Prints all the entities to stdout.

10.215.3.34 `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

<code>in</code>	<code>_model</code>	Pointer to the model to publish.
-----------------	---------------------	----------------------------------

10.215.3.35 `void gazebo::physics::World::RemovePlugin (const std::string & _name)`

Remove a running plugin.

Parameters

<code>in</code>	<code>_name</code>	The unique name of the plugin to remove.
-----------------	--------------------	--

10.215.3.36 `void gazebo::physics::World::Reset ()`

Reset time and model poses, configurations in simulation.

10.215.3.37 `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. 156).

Parameters

<code>in</code>	<code>_type</code>	The type of reset.
-----------------	--------------------	--------------------

10.215.3.38 `void gazebo::physics::World::ResetTime ()`

Reset simulation time back to zero.

10.215.3.39 `void gazebo::physics::World::Run (unsigned int _iterations = 0)`

Run the world in a thread.

Run the update loop.

Parameters

<code>in</code>	<code>_iterations</code>	Run for this many iterations, then stop. A value of zero disables run stop.
-----------------	--------------------------	---

10.215.3.40 `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>_filename</code>	Name of the file to save into.
-----------------	------------------------	--------------------------------

10.215.3.41 `void gazebo::physics::World::SetPaused (bool _p)`

Set whether the simulation is paused.

Parameters

<code>in</code>	<code>_p</code>	True pauses the simulation. False runs the simulation.
-----------------	-----------------	--

10.215.3.42 `void gazebo::physics::World::SetSimTime (const common::Time & _t)`

Set the sim time.

Parameters

<code>in</code>	<code>_t</code>	The new simulation time
-----------------	-----------------	-------------------------

10.215.3.43 `void gazebo::physics::World::SetState (const WorldState & _state)`

Set the current world state.

Parameters

<code>_state</code>	The state to set the World (p. 1070) to.
---------------------	---

10.215.3.44 `void gazebo::physics::World::StepWorld (int _steps)`

Step callback.

Parameters

<code>in</code>	<code>_steps</code>	The number of steps the World (p. 1070) should take.
-----------------	---------------------	---

10.215.3.45 `void gazebo::physics::World::Stop ()`

Stop the world.

Stop the update loop.

10.215.3.46 `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.

10.215.3.47 `void gazebo::physics::World::UpdateStateSDF ()`

Update the state SDF value from the current state.

10.215.4 Member Data Documentation

10.215.4.1 `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. 303) on the **physics::Link** (p. 455) in `World::Update`.

The documentation for this class was generated from the following file:

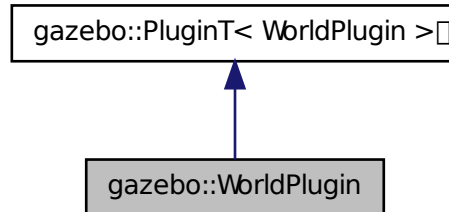
- **World.hh**

10.216 gazebo::WorldPlugin Class Reference

A plugin with access to **physics::World** (p. 1070).

```
#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

10.216.1 Detailed Description

A plugin with access to **physics::World** (p. 1070).

See reference.

10.216.2 Constructor & Destructor Documentation

10.216.2.1 gazebo::WorldPlugin::WorldPlugin () [inline]

Constructor.

References gazebo::PluginT< WorldPlugin >::type, and gazebo::WORLD_PLUGIN.

10.216.2.2 virtual gazebo::WorldPlugin::~~WorldPlugin () [inline],[virtual]

Destructor.

10.216.3 Member Function Documentation

10.216.3.1 virtual void gazebo::WorldPlugin::Init () [inline],[virtual]

10.216.3.2 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	<i>_world</i>	Pointer the World
in	<i>_sdf</i>	Pointer the the SDF element of the plugin.

10.216.3.3 virtual void gazebo::WorldPlugin::Reset () [inline],[virtual]

The documentation for this class was generated from the following file:

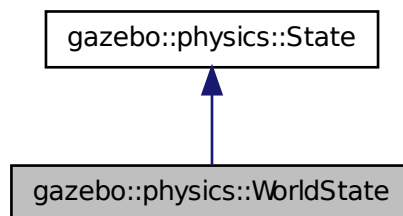
- **Plugin.hh**

10.217 gazebo::physics::WorldState Class Reference

Store state information of a **physics::World** (p. 1070) 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. 1083) has a **ModelState** (p. 554) 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. 1070) 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

10.217.1 Detailed Description

Store state information of a **physics::World** (p. 1070) object.

Instances of this class contain the state of a **World** (p. 1070) at a specific time. **World** (p. 1070) state includes the state of all models, and their children.

10.217.2 Constructor & Destructor Documentation

10.217.2.1 gazebo::physics::WorldState::WorldState ()

Default constructor.

10.217.2.2 gazebo::physics::WorldState::WorldState (const WorldPtr *_world*) [explicit]

Constructor.

Generate a **WorldState** (p. 1083) from an instance of a **World** (p. 1070).

Parameters

in	<i>_world</i>	Pointer to a world
----	---------------	--------------------

10.217.2.3 gazebo::physics::WorldState::WorldState (const sdf::ElementPtr *_sdf*) [explicit]

Constructor.

Build a **WorldState** (p. 1083) from SDF data

Parameters

in	<i>_sdf</i>	SDF data to load a world state from.
----	-------------	--------------------------------------

10.217.2.4 virtual gazebo::physics::WorldState::~~WorldState () [virtual]

Destructor.

10.217.3 Member Function Documentation

10.217.3.1 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.
-----	-------------	--------------------------

10.217.3.2 `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. 331)	When the <i>_modelName</i> doesn't exist.
---	---

10.217.3.3 `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.

10.217.3.4 `ModelState_M` gazebo::physics::WorldState::GetModelStates (const boost::regex & *_regex*) const

Get model states based on a regular expression.

Parameters

in	<i>_regex</i>	The regular expression.
----	---------------	-------------------------

Returns

List of model states whose names match the regular expression.

10.217.3.5 `const ModelState_M&` gazebo::physics::WorldState::GetModelStates () const

Get the model states.

Returns

A vector of model states.

10.217.3.6 `bool` gazebo::physics::WorldState::HasModelState (const std::string & *_modelName*) const

Return true if `WorldState` (p. 1083) has a `ModelState` (p. 554) with the given name.

Parameters

in	<code>_modelName</code>	Name of the model to search for.
----	-------------------------	----------------------------------

Returns

True if the **ModelState** (p. 554) exists.

10.217.3.7 `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.

10.217.3.8 `void gazebo::physics::WorldState::Load (const WorldPtr _world)`

Load from a **World** (p. 1070) pointer.

Generate a **WorldState** (p. 1083) from an instance of a **World** (p. 1070).

Parameters

in	<code>_world</code>	Pointer to a world
----	---------------------	--------------------

10.217.3.9 `virtual void gazebo::physics::WorldState::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Set a **WorldState** (p. 1083) from an SDF element containing **WorldState** (p. 1083) info.

Parameters

in	<code>_elem</code>	Pointer to the WorldState (p. 1083) SDF element.
----	--------------------	---

Reimplemented from **gazebo::physics::State** (p. 913).

10.217.3.10 `WorldState gazebo::physics::WorldState::operator+ (const WorldState & _state) const`

Addition operator.

Parameters

in	<code>_pt</code>	A state to add.
----	------------------	-----------------

Returns

The resulting state.

10.217.3.11 **WorldState** gazebo::physics::WorldState::operator- (const WorldState & *_state*) const

Subtraction operator.

Parameters

<i>in</i>	<i>_pt</i>	A state to subtract.
-----------	------------	----------------------

Returns

The resulting state.

10.217.3.12 **WorldState&** gazebo::physics::WorldState::operator= (const WorldState & *_state*)

Assignment operator.

Parameters

<i>in</i>	<i>_state</i>	State (p.910) value
-----------	---------------	----------------------------

Returns

Reference to this

10.217.3.13 **virtual void** gazebo::physics::WorldState::SetRealTime (const common::Time & *_time*) [virtual]

Set the real time when this state was generated.

Parameters

<i>in</i>	<i>_time</i>	Clock time since simulation was stated.
-----------	--------------	---

Reimplemented from **gazebo::physics::State** (p.914).

10.217.3.14 **virtual void** gazebo::physics::WorldState::SetSimTime (const common::Time & *_time*) [virtual]

Set the sim time when this state was generated.

Parameters

<i>in</i>	<i>_time</i>	Simulation time when the data was recorded.
-----------	--------------	---

Reimplemented from **gazebo::physics::State** (p.914).

10.217.3.15 **virtual void** gazebo::physics::WorldState::SetWallTime (const common::Time & *_time*) [virtual]

Set the wall time when this state was generated.

Parameters

in	<code>_time</code>	The absolute clock time when the State (p. 910) data was recorded.
----	--------------------	---

Reimplemented from **gazebo::physics::State** (p. 914).

10.217.3.16 `void gazebo::physics::WorldState::SetWorld (const WorldPtr _world)`

Set the world.

Parameters

in	<code>_world</code>	Pointer to the world.
----	---------------------	-----------------------

10.217.4 Friends And Related Function Documentation

10.217.4.1 `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. 1070) state to output

Returns

the stream

The documentation for this class was generated from the following file:

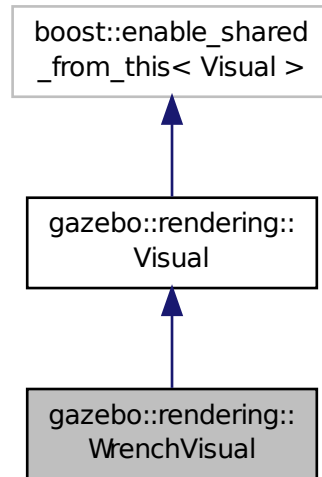
- **WorldState.hh**

10.218 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

10.218.1 Detailed Description

Visualization for sonar data.

10.218.2 Constructor & Destructor Documentation

- 10.218.2.1 `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. 1034).
in	<code>_topicName</code>	Name of the topic that has sonar data.

10.218.2.2 `virtual gazebo::rendering::WrenchVisual::~~WrenchVisual () [virtual]`

Destructor.

10.218.3 Member Function Documentation

10.218.3.1 `void gazebo::rendering::WrenchVisual::Load (ConstJointPtr & _msg)`

Load the visual based on a message.

Parameters

in	<code>_msg</code>	Joint message
----	-------------------	---------------

10.218.3.2 `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**

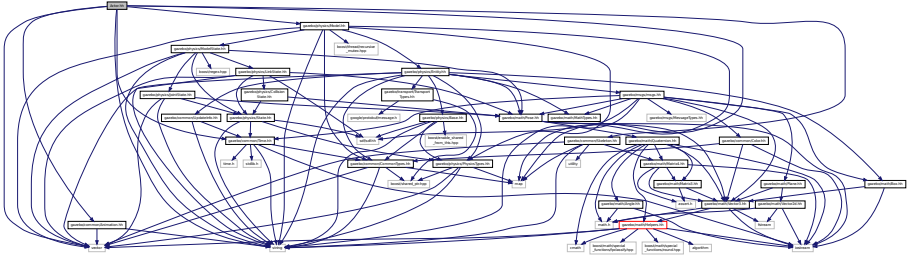
Chapter 11

File Documentation

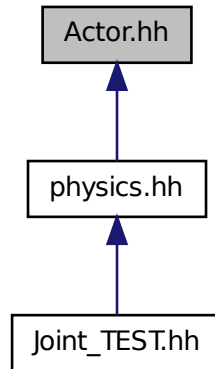
11.1 Actor.hh File Reference

```
#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 dependency graph for Actor.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Actor**
Actor (p. 125) class enables GPU based mesh model / skeleton scriptable animation.
- struct **gazebo::physics::TrajectoryInfo**

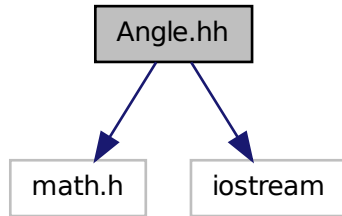
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::physics**
namespace for physics

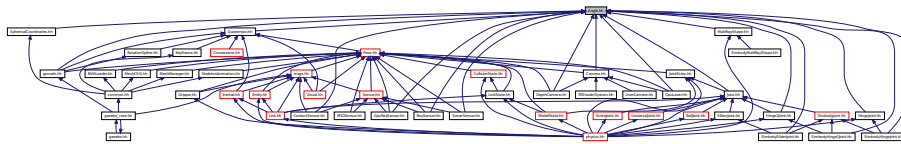
11.2 Angle.hh File Reference

```
#include <math.h>  
#include <iostream>
```

Include 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

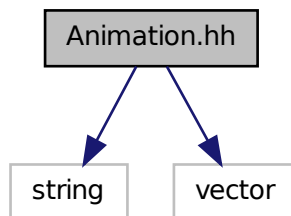
degrees

11.3 Animation.hh File Reference

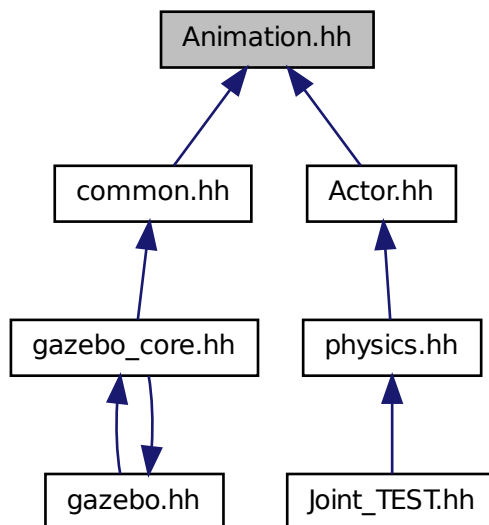
```
#include <string>
```

```
#include <vector>
```

Include dependency graph for Animation.hh:



This graph shows which files directly or indirectly include this file:



Classes

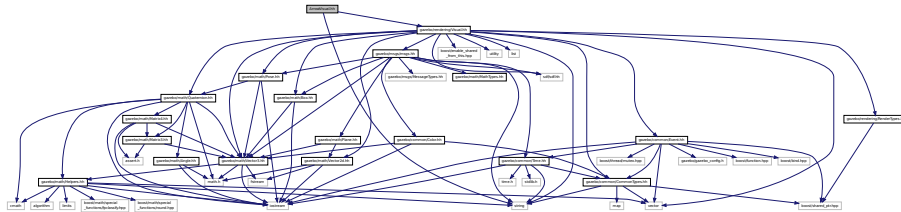
- class **gazebo::common::Animation**
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.
- class **gazebo::common::NumericAnimation**
A numeric animation.
- class **gazebo::common::PoseAnimation**
A pose animation.

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 dependency graph for ArrowVisual.hh:
```



Classes

- class **gazebo::rendering::ArrowVisual**
Basic arrow visualization.

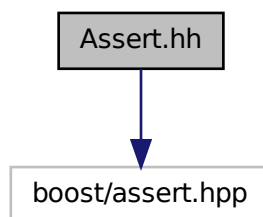
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **ogre**

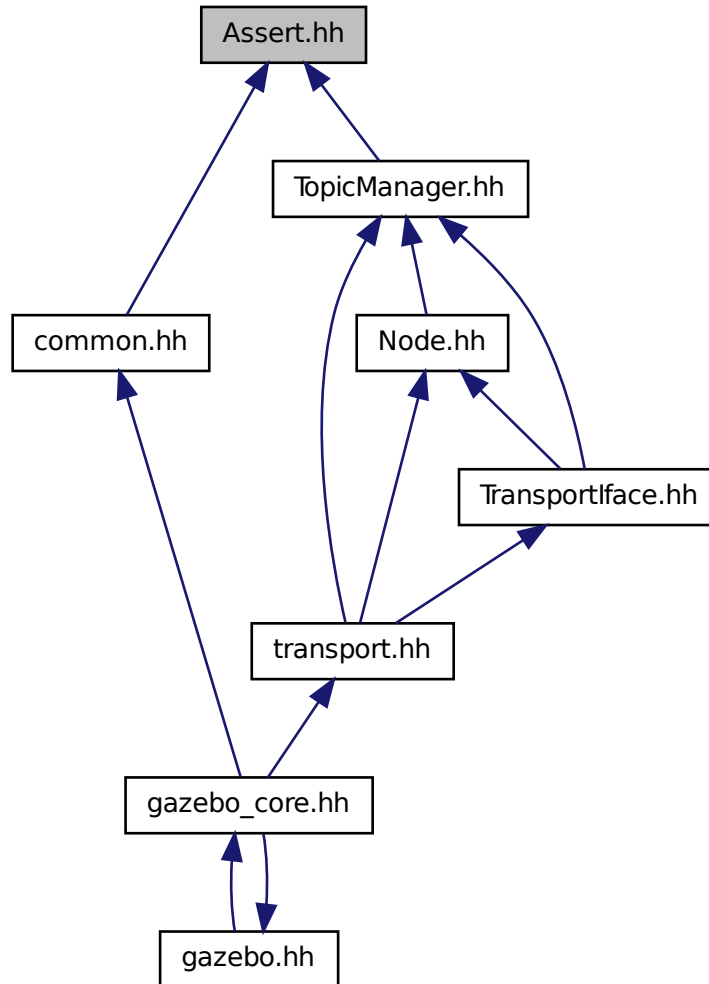
11.5 Assert.hh File Reference

```
#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.

11.5.1 Macro Definition Documentation

11.5.1.1 `#define GZ_ASSERT(_expr, _msg) BOOST_ASSERT_MSG(_expr, _msg)`

This macro define the standard way of launching an exception inside gazebo.

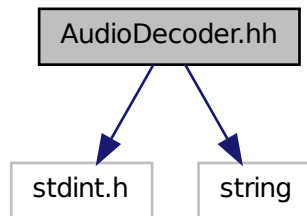
Referenced by gazebo::transport::TopicManager::Advertise().

11.6 AudioDecoder.hh File Reference

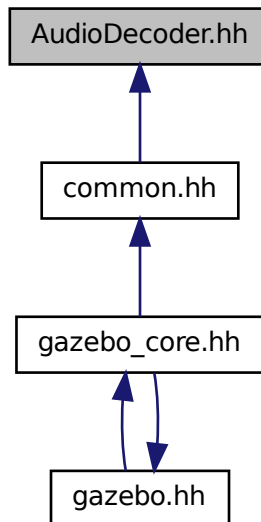
```
#include <stdint.h>
```

```
#include <string>
```

Include dependency graph for AudioDecoder.hh:



This graph shows which files directly or indirectly include this file:



Classes

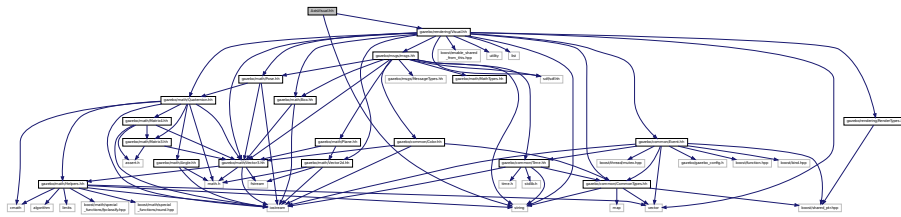
- class **gazebo::common::AudioDecoder**
An audio decoder based on FFmpeg.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.7 AxisVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
Include dependency graph for AxisVisual.hh:
```



Classes

- class **gazebo::rendering::AxisVisual**
Basic axis visualization.

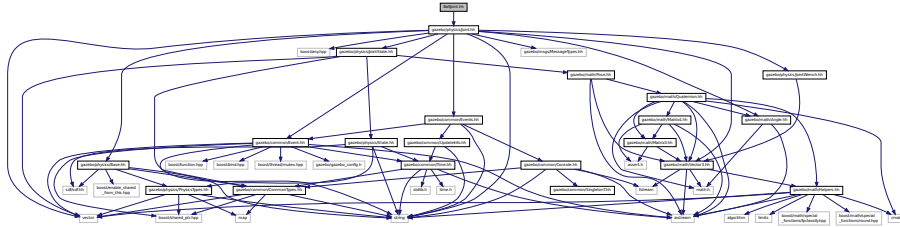
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

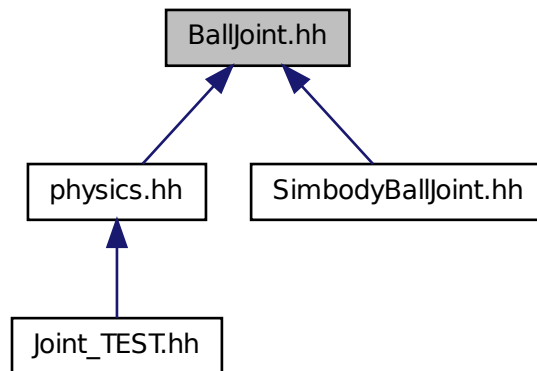
11.8 BallJoint.hh File Reference

```
#include "gazebo/physics/Joint.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. 153) class for a ball joint.

Namespaces

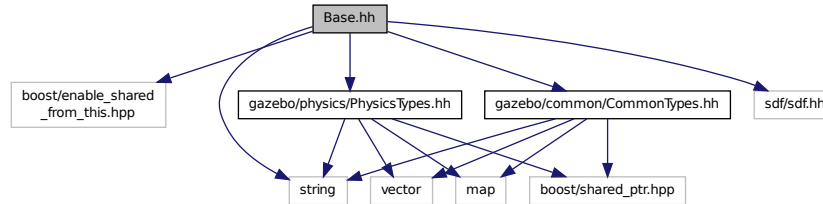
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.9 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 dependency graph for Base.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Base**
Base (p. 153) class for most physics classes.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

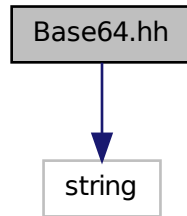
Variables

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

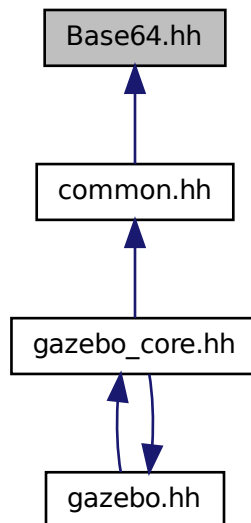
11.10 Base64.hh File Reference

```
#include <string>
```


Include dependency graph for Base64.hh:



This graph shows which files directly or indirectly include this file:



Functions

- std::string **Base64Decode** (const std::string &_encodedString)
Decode a base64 string.
- void **Base64Encode** (const char *_bytesToEncode, unsigned int _len, std::string &_result)
Encode a binary string into base 64.

11.10.1 Function Documentation

11.10.1.1 `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.

11.10.1.2 `void Base64Encode (const char * _bytesToEncode, unsigned int _len, std::string & _result)`

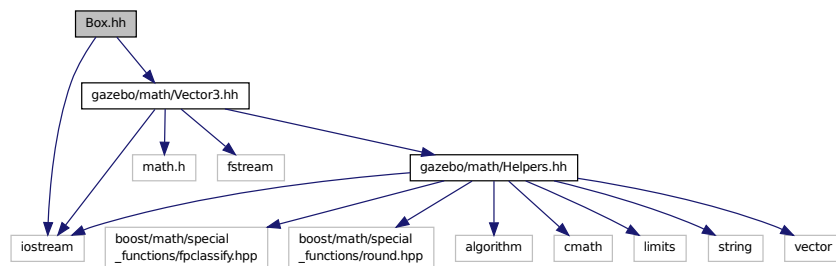
Encode a binary string into base 64.

Parameters

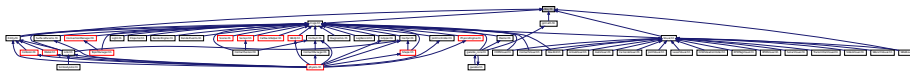
in	<code>_bytesToEncode</code>	String of bytes to encode.
in	<code>_len</code>	Length of <code>_bytesToEncode</code> .
out	<code>_result</code>	Based64 string is appended to this string.

11.11 Box.hh File Reference

```
#include <iostream>
#include "gazebo/math/Vector3.hh"
Include dependency graph for Box.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

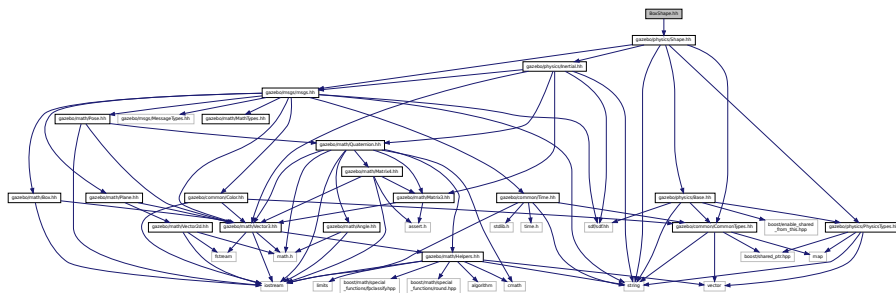
- class **gazebo::math::Box**
Mathematical representation of a box and related functions.

Namespaces

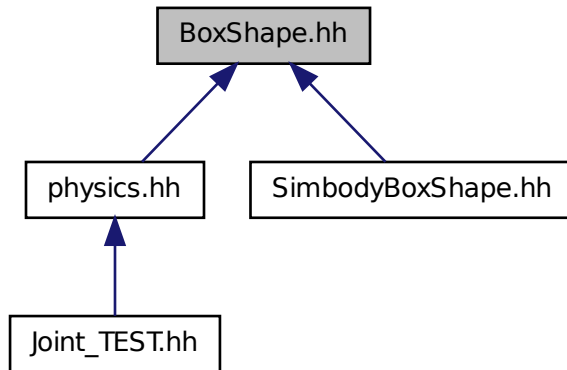
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

11.12 BoxShape.hh File Reference

```
#include "gazebo/physics/Shape.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**

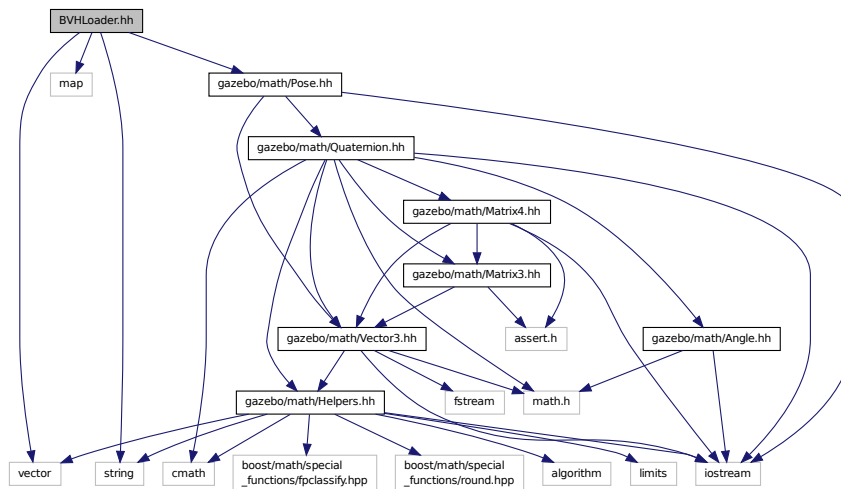
Forward declarations for the common classes.

- namespace **gazebo::physics**

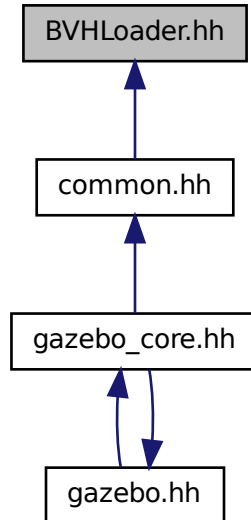
namespace for physics

11.13 BVHLoader.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include "gazebo/math/Pose.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**

11.13.1 Macro Definition Documentation

11.13.1.1 `#define X_POSITION 0`

11.13.1.2 `#define X_ROTATION 3`

11.13.1.3 `#define Y_POSITION 1`

11.13.1.4 `#define Y_ROTATION 4`

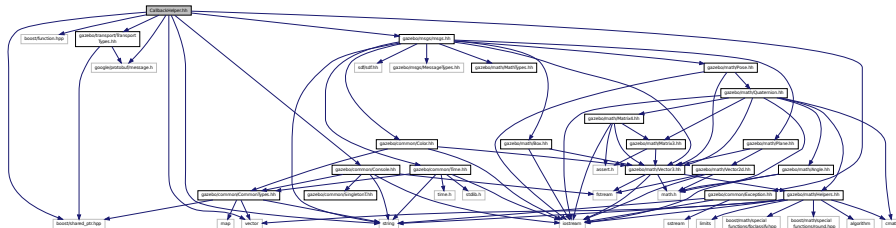
11.13.1.5 `#define Z_POSITION 2`

11.13.1.6 `#define Z_ROTATION 5`

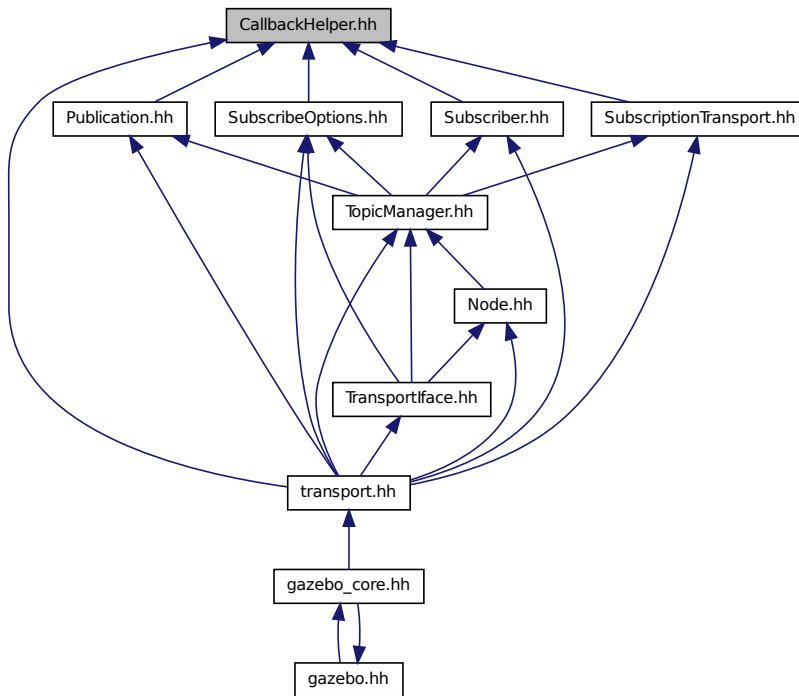
11.14 CallbackHelper.hh File Reference

```
#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 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**

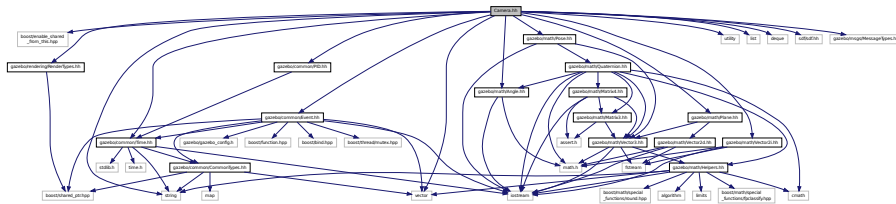
Typedefs

- typedef boost::shared_ptr
< CallbackHelper > **gazebo::transport::CallbackHelperPtr**
*boost shared pointer to **transport::CallbackHelper** (p. 173)*

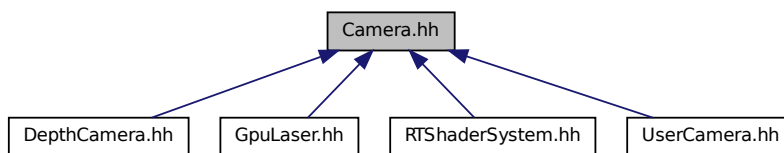
11.15 Camera.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include <utility>
#include <list>
#include <vector>
#include <deque>
#include <sdf/sdf.hh>
#include "gazebo/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 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

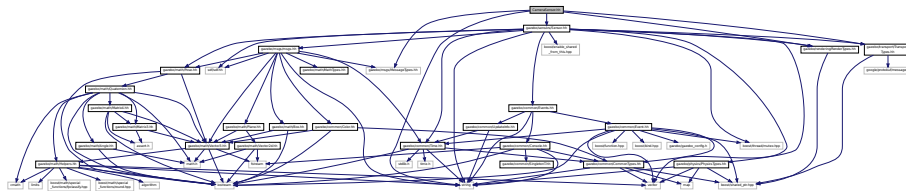
- namespace **gazebo**
Forward declarations for the common classes.

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

11.16 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 dependency graph for CameraSensor.hh:



Classes

- class **gazebo::sensors::CameraSensor**
Basic camera sensor.

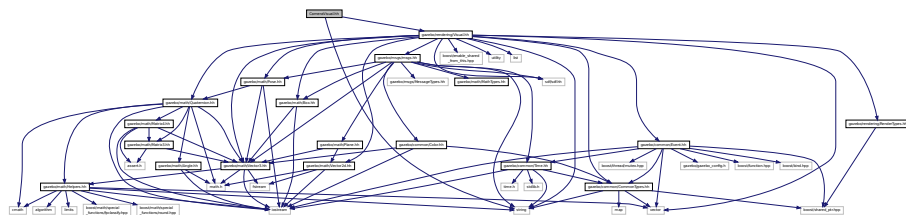
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.17 CameraVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for CameraVisual.hh:



Classes

- class **gazebo::rendering::CameraVisual**

Basic camera visualization.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

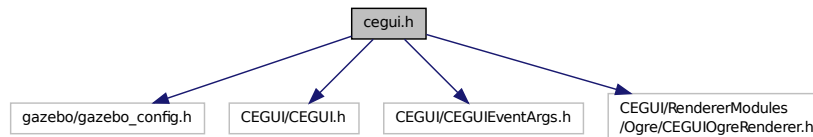
- namespace **gazebo::rendering**

Rendering namespace.

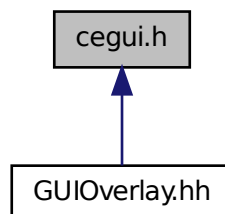
11.18 cegui.h File Reference

```
#include "gazebo/gazebo_config.h"
#include <CEGUI/CEGUI.h>
#include <CEGUI/CEGUIEventArgs.h>
#include <CEGUI/RendererModules/Ogre/CEGUIOgreRenderer.h>
```

Include dependency graph for cegui.h:

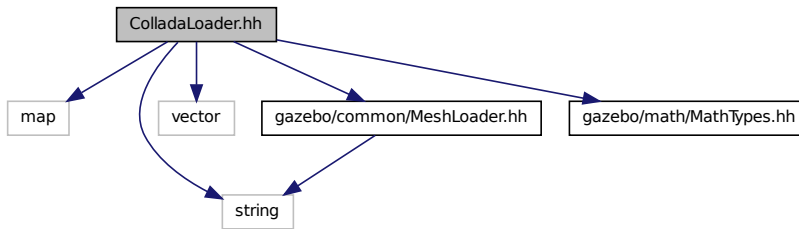


This graph shows which files directly or indirectly include this file:

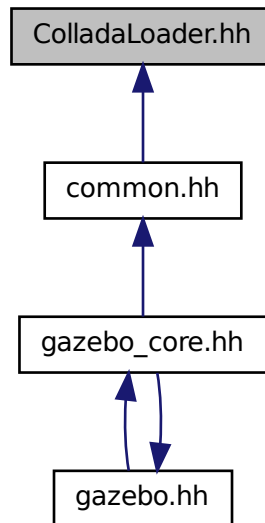


11.19 ColladaLoader.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include "gazebo/common/MeshLoader.hh"
#include "gazebo/math/MathTypes.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.

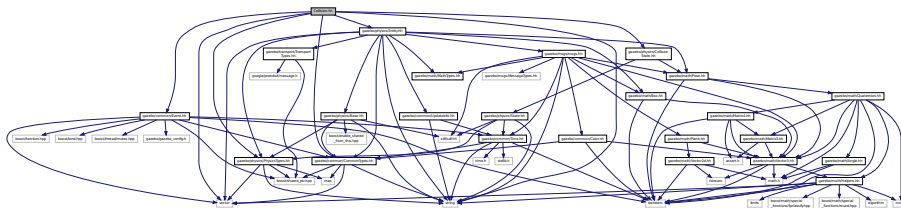
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

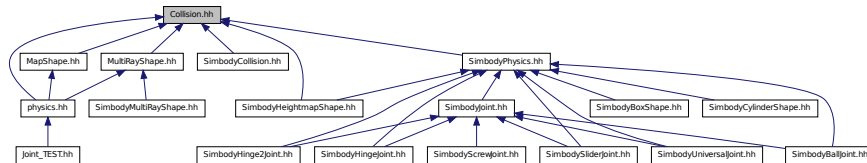
11.20 Collision.hh File Reference

```
#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 dependency graph for Collision.hh:



This graph shows which files directly or indirectly include this file:



Classes

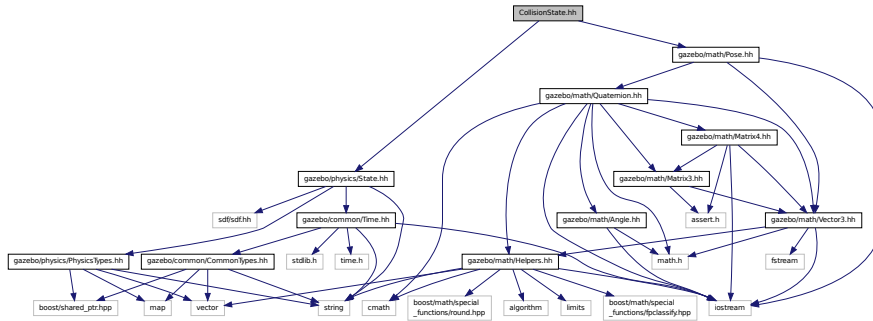
- class **gazebo::physics::Collision**
Base (p. 153) class for all collision entities.

Namespaces

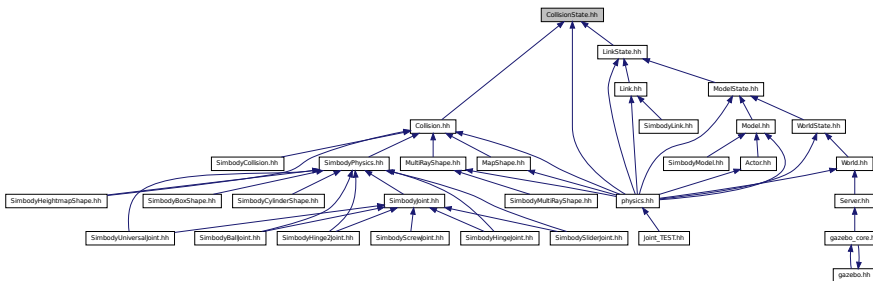
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.21 CollisionState.hh File Reference

```
#include "gazebo/physics/State.hh"
#include "gazebo/math/Pose.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. 213) object.

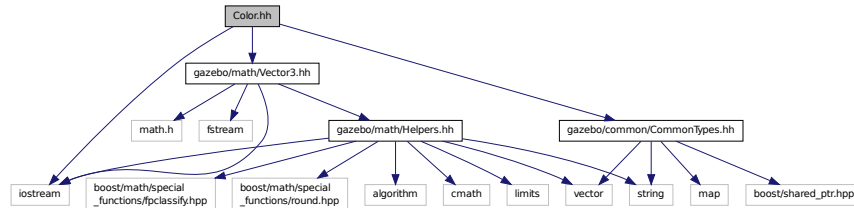
Namespaces

- namespace **`gazebo`**
Forward declarations for the common classes.
- namespace **`gazebo::physics`**
namespace for physics

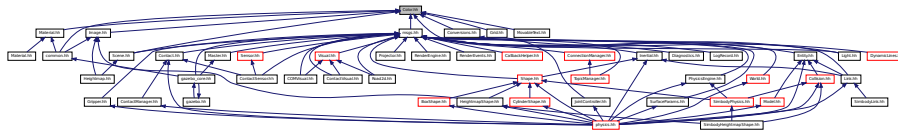
11.22 Color.hh File Reference

```
#include <iostream>
```

```
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/math/Vector3.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.

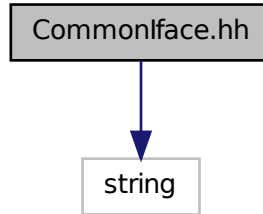
- namespace `gazebo::common`

Common namespace.

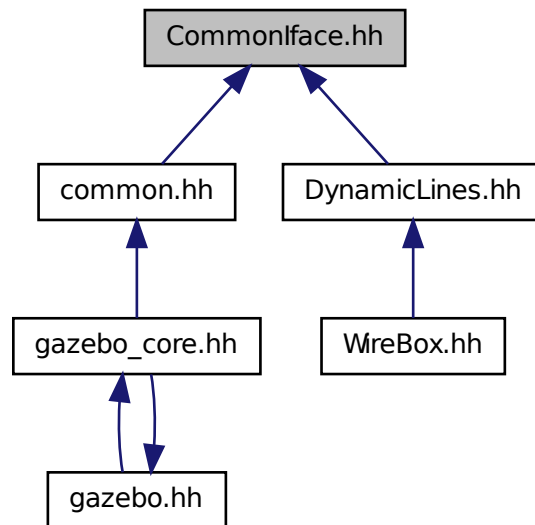
11.23 Commonface.hh File Reference

```
#include <string>
```

Include dependency graph for Commonface.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

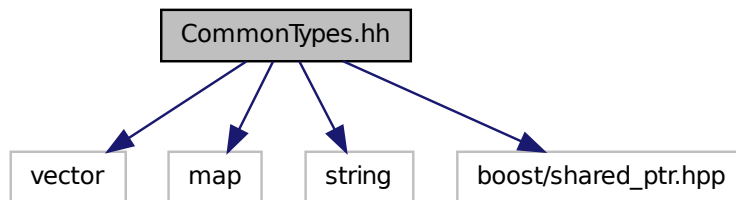
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

Functions

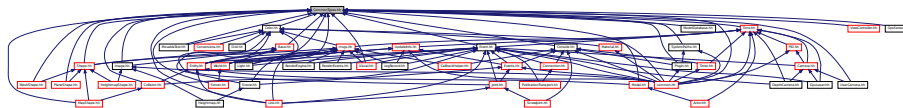
- void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)
*add path prefix to **common::SystemPaths** (p. 937)*
- std::string **gazebo::common::find_file** (const std::string &_file)
*search for file in **common::SystemPaths** (p. 937)*
- std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath)
*search for file in **common::SystemPaths** (p. 937)*
- std::string **gazebo::common::find_file_path** (const std::string &_file)
*search for a file in **common::SystemPaths** (p. 937)*
- void **gazebo::common::load** ()
Load the common library.

11.24 CommonTypes.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
Include dependency graph for CommonTypes.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::ParamT** < T >

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::event**
Event (p. 305) 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.

11.24.1 Macro Definition Documentation

11.24.1.1 `#define GAZEBO_DEPRECATED(version)()`

11.24.1.2 `#define GAZEBO_FORCEINLINE`

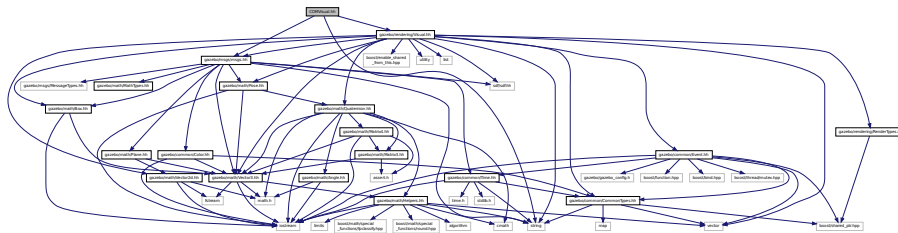
11.24.1.3 `#define NULL 0`

Referenced by `gazebo::transport::TopicManager::Advertise()`, `gazebo::PluginT< ModelPlugin >::Create()`, `gazebo::event::EventT< T >::Disconnect()`, `gazebo::transport::CallbackHelperT< M >::GetMsgType()`, `gazebo::transport::SubscribeOptions::Init()`, `gazebo::PluginT< ModelPlugin >::PluginT()`, and `Joint_TEST::SpawnJoint()`.

11.25 COMVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo msgs/msgs.hh"
```

Include dependency graph for COMVisual.hh:



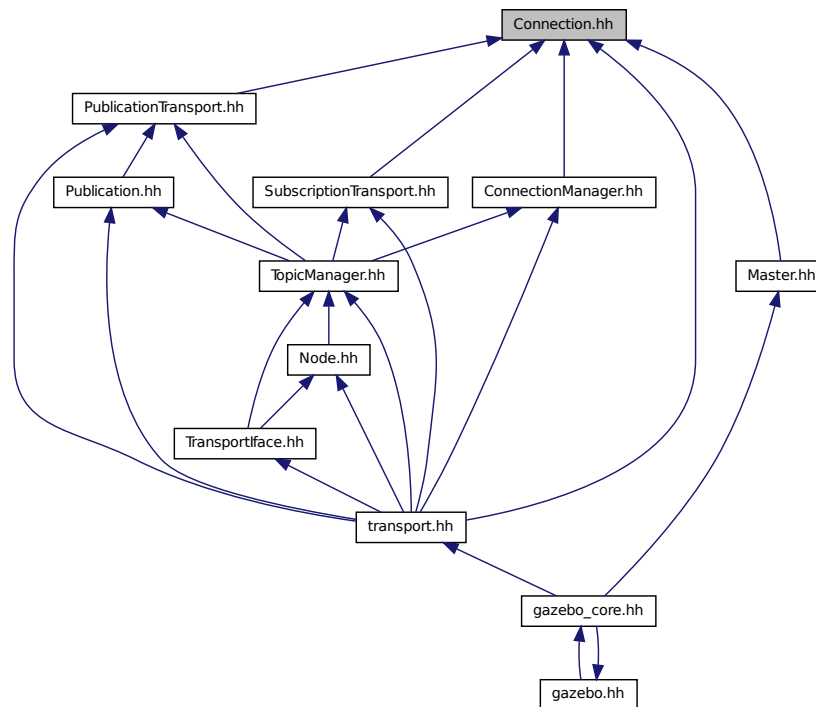
Classes

- class **gazebo::rendering::COMVisual**
Basic Center of Mass visualization.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **ogre**

This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Connection**
Single TCP/IP connection manager.

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?

11.26.1 Macro Definition Documentation

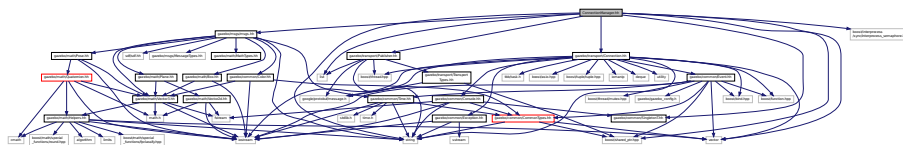
11.26.1.1 #define HEADER_LENGTH 8

Referenced by `gazebo::transport::Connection::AsyncRead()`.

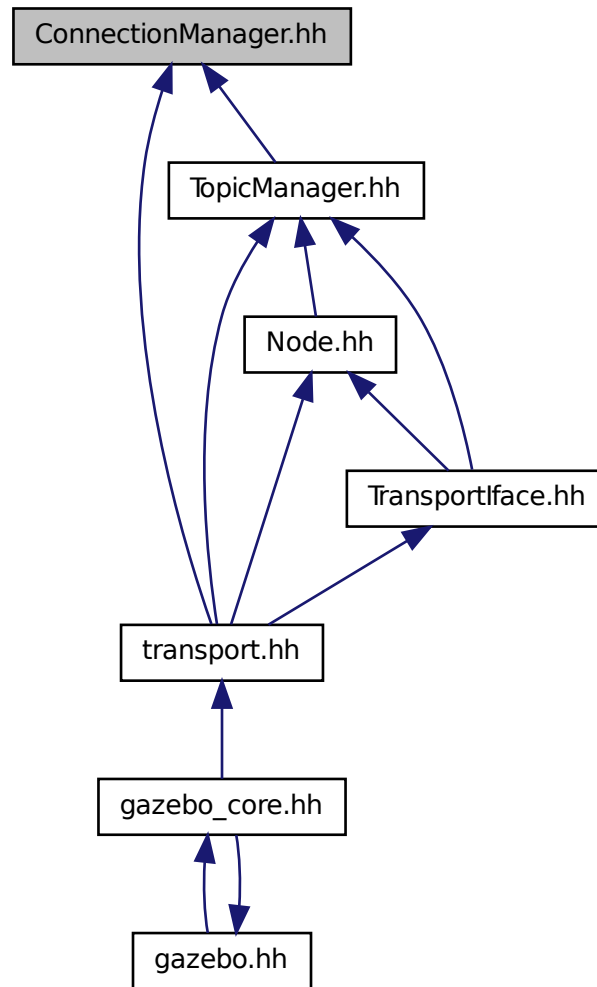
11.27 ConnectionManager.hh File Reference

```
#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 dependency graph for ConnectionManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::ConnectionManager**
Manager of connections.

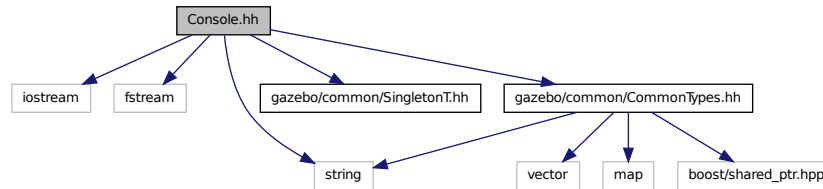
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

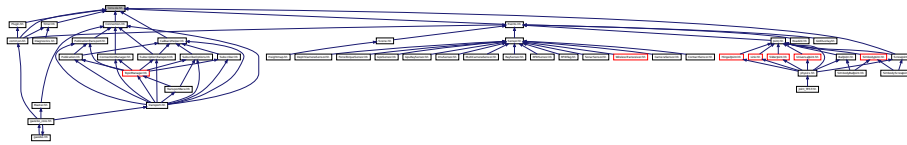
11.28 Console.hh File Reference

```
#include <iostream>
#include <fstream>
#include <string>
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/CommonTypes.hh"
```

Include dependency graph for Console.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Console**
Message, error, warning functionality.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

Macros

- #define **gzclr_end** "\033[0m"
End marker.
- #define **gzclr_start**(clr) "\033[1;33m"
Start marker.
- #define **gzdbg** (gazebo::common::Console::Instance()->ColorMsg("Dbg", 36))
Output a debug message.

- **#define gzerr**

Output an error message.

- **#define gzlog (gazebo::common::Console::Instance()->Log())**

Output a message to a log file.

- **#define gzmsg (gazebo::common::Console::Instance()->ColorMsg("Msg", 32))**

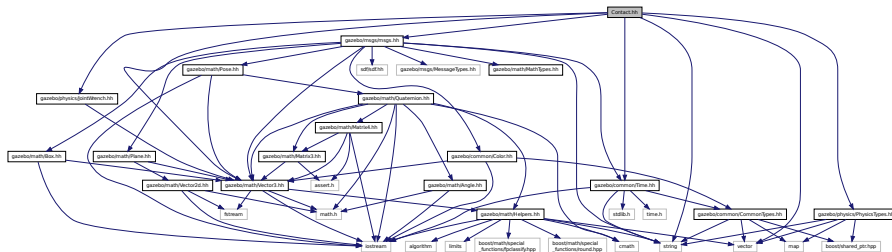
Output a message.

- **#define gzwarn**

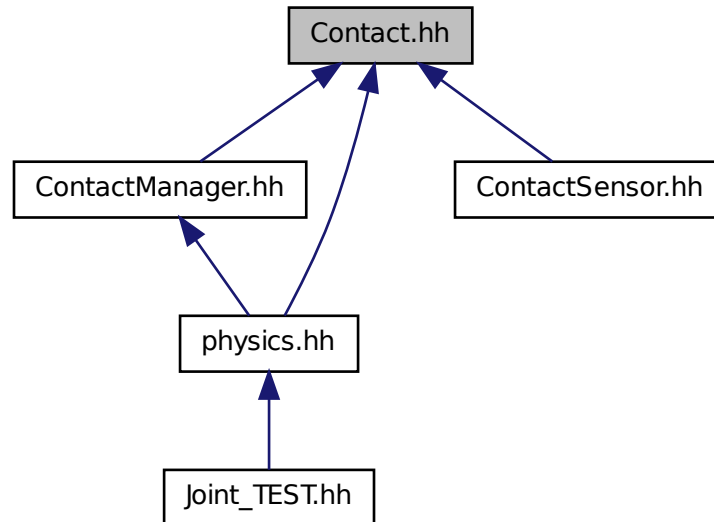
Output a warning message.

11.29 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 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

11.29.1 Macro Definition Documentation

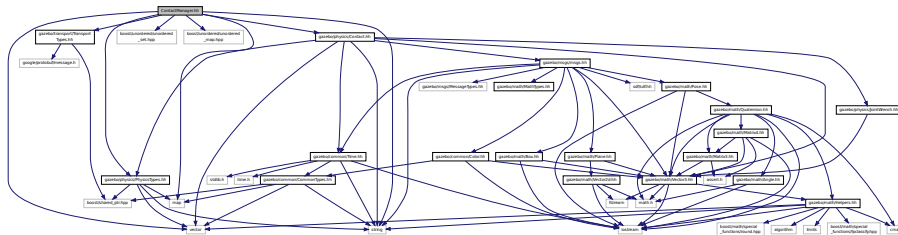
11.29.1.1 #define MAX_COLLIDE_RETURNS 250

11.29.1.2 #define MAX_CONTACT_JOINTS 32

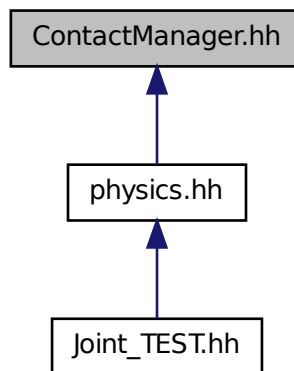
11.30 ContactManager.hh File Reference

```
#include <vector>
#include <string>
#include <map>
#include <boost/unordered/unordered_set.hpp>
#include <boost/unordered/unordered_map.hpp>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Contact.hh"
```

Include dependency graph for ContactManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

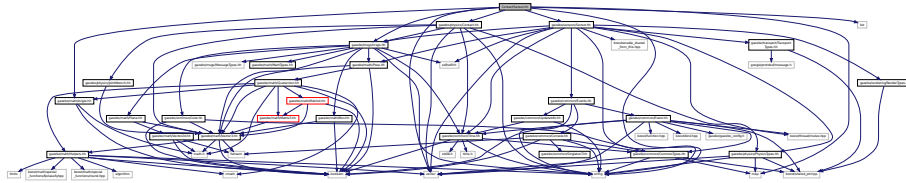
- class **gazebo::physics::ContactManager**
Aggregates all the contact information generated by the collision detection engine.
- class **gazebo::physics::ContactPublisher**
*A custom contact publisher created for each contact filter in the **Contact** (p. 253) Manager.*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.31 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 dependency graph for ContactSensor.hh:
```



Classes

- class **gazebo::sensors::ContactSensor**
Contact sensor.

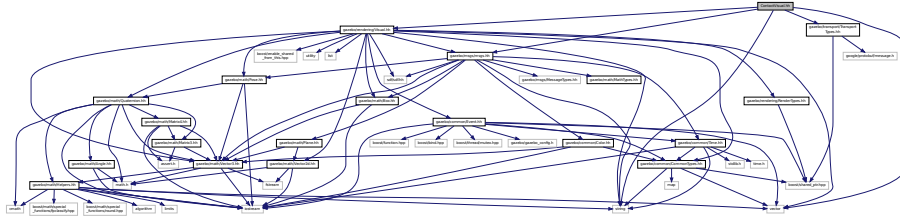
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.32 ContactVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.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.

- namespace **Ogre**

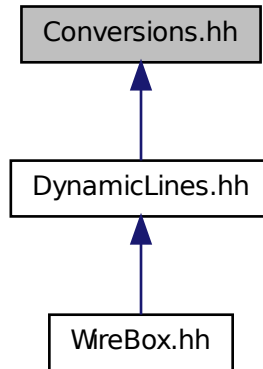
11.33 Conversions.hh File Reference

```
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/common/Color.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Quaternion.hh"
```

Include dependency graph for Conversions.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::Conversions**
Conversions (p. 266) *Conversions.hh* (p. 1132) *rendering/Conversions.hh* (p. 1132).

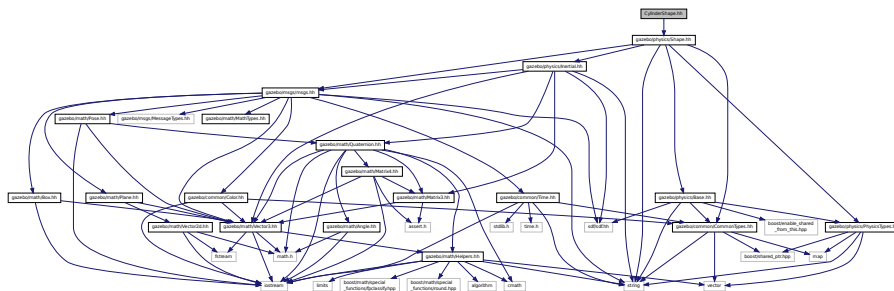
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

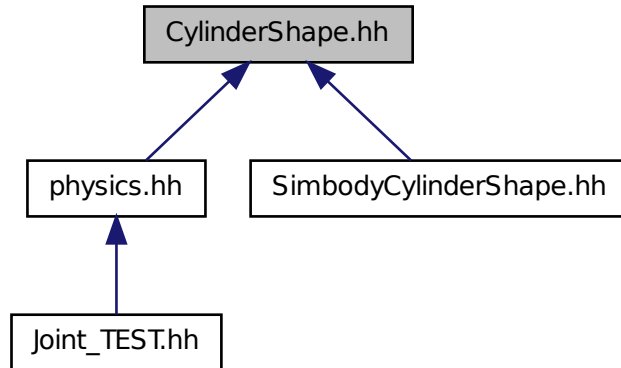
11.34 CylinderShape.hh File Reference

```
#include "gazebo/physics/Shape.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.

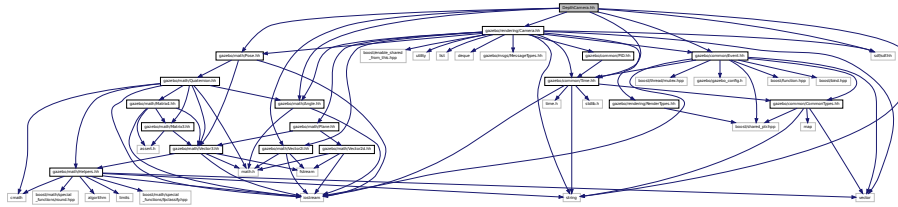
- namespace **gazebo::physics**

namespace for physics

11.35 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 dependency graph for DepthCamera.hh:



Classes

- class **gazebo::rendering::DepthCamera**
Depth camera used to render depth data into an image buffer.

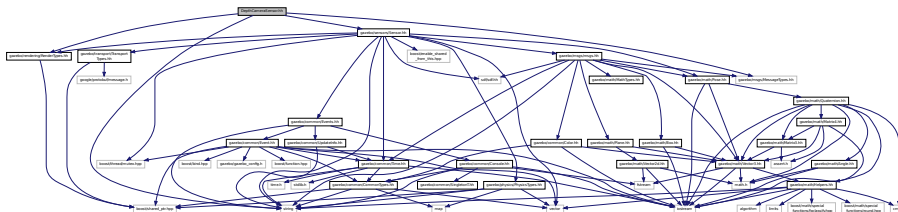
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.36 DepthCameraSensor.hh File Reference

```
#include <string>
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/messages/MessageTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for DepthCameraSensor.hh:



Classes

- class **gazebo::sensors::DepthCameraSensor**

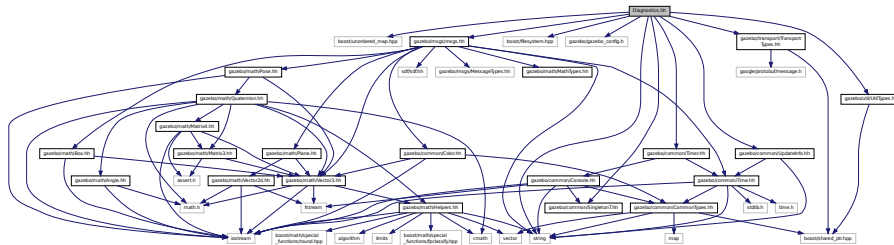
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.37 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 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)

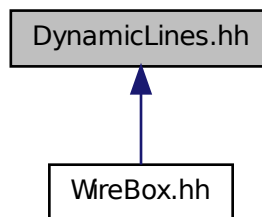
11.38 DynamicLines.hh File Reference

```
#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 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.

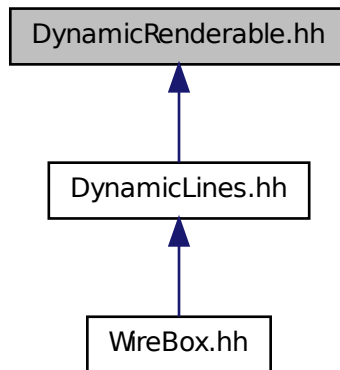
11.39 DynamicRenderable.hh File Reference

```
#include <string>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/rendering/RenderTypes.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.

11.40 Entity.hh File Reference

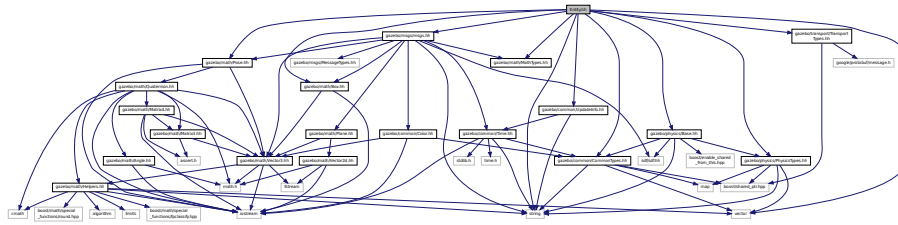
```
#include <string>
```

```

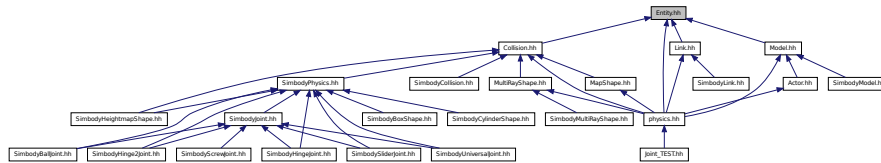
#include <vector>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/math/MathTypes.hh"
#include "gazebo/math/Box.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Base.hh"

```

Include dependency graph for Entity.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Entity**
Base (p. 153) class for all physics objects in Gazebo.

Namespaces

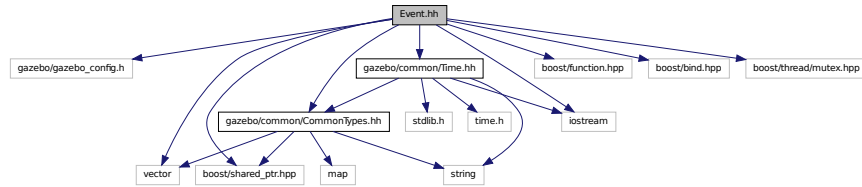
- namespace **boost**
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.41 Event.hh File Reference

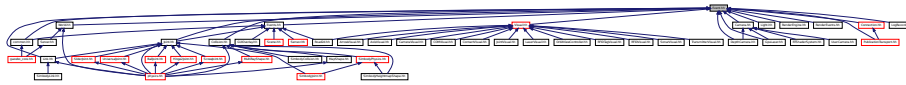
```
#include <gazebo/gazebo_config.h>
```

```
#include <gazebo/common/Time.hh>
#include <gazebo/common/CommonTypes.hh>
#include <boost/function.hpp>
#include <boost/bind.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/thread/mutex.hpp>
#include <iostream>
#include <vector>
```

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::Event**
Base class for all events.
- class **gazebo::event::EventT < T >**
A class for event processing.

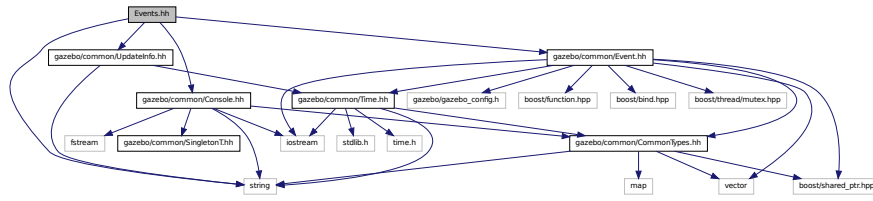
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::event**
Event (p. 305) namespace.

11.42 Events.hh File Reference

```
#include <string>
#include "gazebo/common/Console.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/Event.hh"
```

Include dependency graph for Events.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::event::Events**

An **Event** (p. 305) class to get notifications for simulator events.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

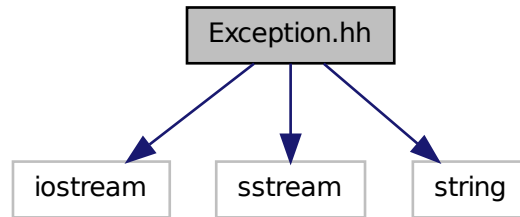
- namespace **gazebo::event**

Event (p. 305) namespace.

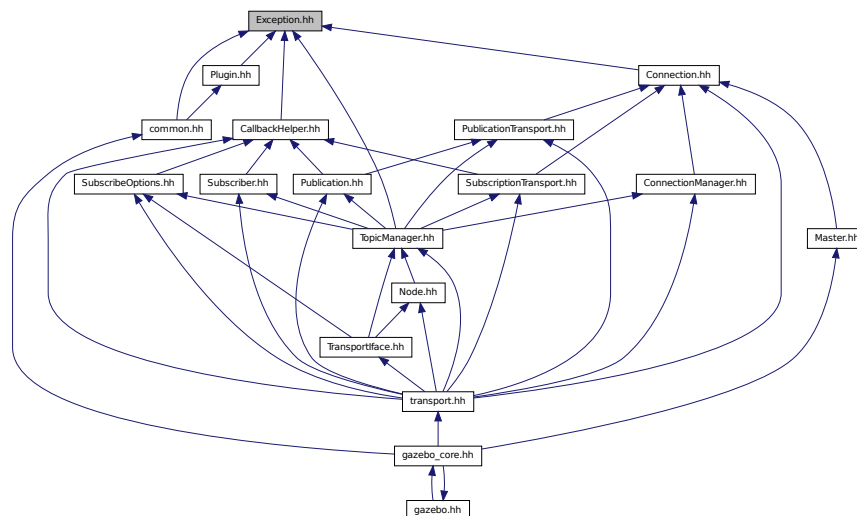
11.43 Exception.hh File Reference

```
#include <iostream>
#include <sstream>
#include <string>
```

Include dependency graph for Exception.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

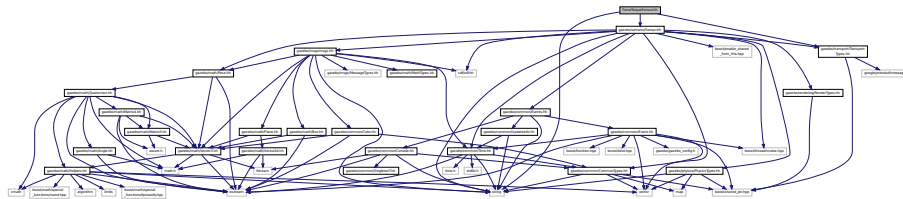
- **#define gzthrow(msg)**

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

11.44 ForceTorqueSensor.hh File Reference

```
#include <string>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/sensors/Sensor.hh"
```

Include dependency graph for ForceTorqueSensor.hh:



Classes

- class **gazebo::sensors::ForceTorqueSensor**

Sensor (p. 751) for measure force and torque on a joint.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

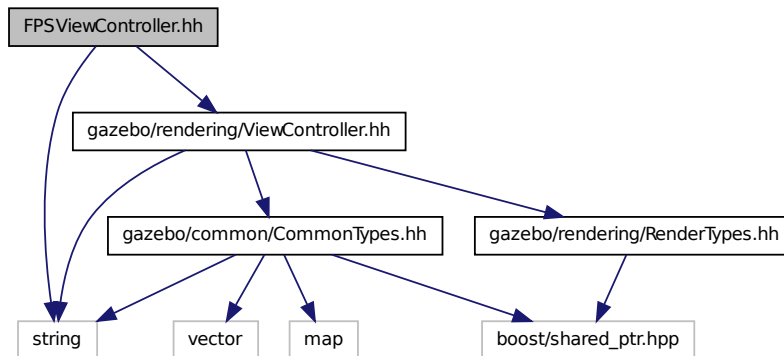
- namespace **gazebo::sensors**

Sensors namespace.

11.45 FPSViewController.hh File Reference

```
#include <string>
#include "gazebo/rendering/ViewController.hh"
```

Include dependency graph for FPSViewController.hh:



Classes

- class **gazebo::rendering::FPSViewController**

First Person Shooter style view controller.

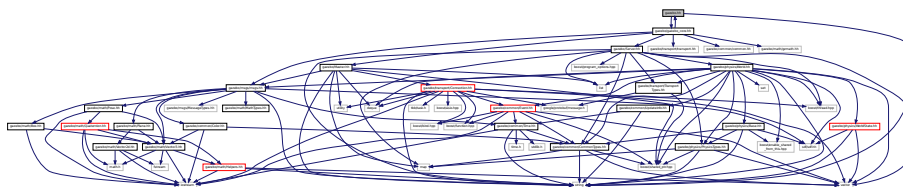
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

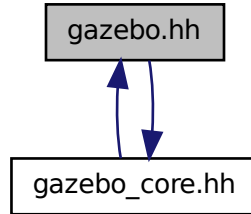
11.46 gazebo.hh File Reference

```
#include <gazebo/gazebo_core.hh>
#include <string>
```

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.

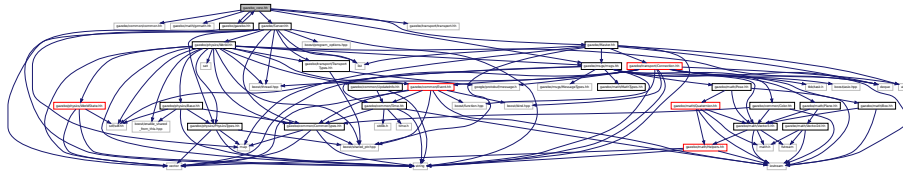
Functions

- void **gazebo::add_plugin** (const std::string &_filename)
- std::string **gazebo::find_file** (const std::string &_file)
Find a file in the gazebo search paths.
- void **gazebo::fini** ()
- bool **gazebo::init** ()
- bool **gazebo::load** (int _argc=0, char **_argv=0)
- void **gazebo::print_version** ()
- void **gazebo::run** ()
- void **gazebo::stop** ()

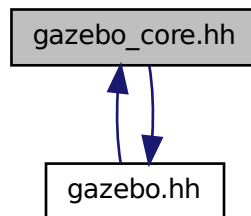
11.47 gazebo_core.hh File Reference

```
#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:



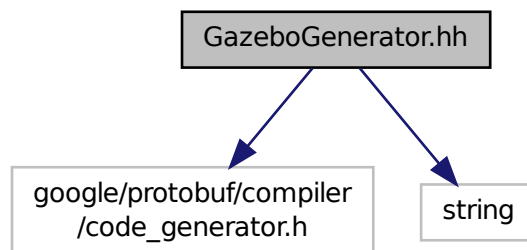
This graph shows which files directly or indirectly include this file:



11.48 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. 102).*

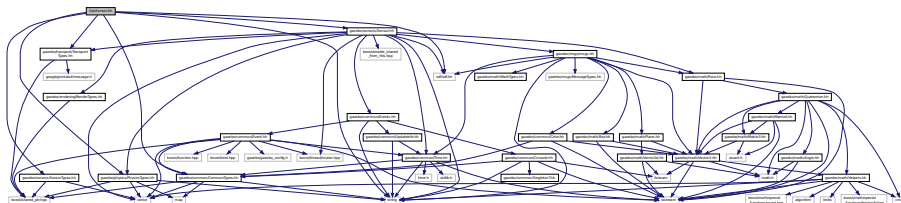
Namespaces

- namespace **google**
- namespace **google::protobuf**
- namespace **google::protobuf::compiler**
- namespace **google::protobuf::compiler::cpp**

11.49 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 dependency graph for GpsSensor.hh:



Classes

- class **gazebo::sensors::GpsSensor**
***GpsSensor** (p. 341) to provide position measurement.*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.50 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 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**

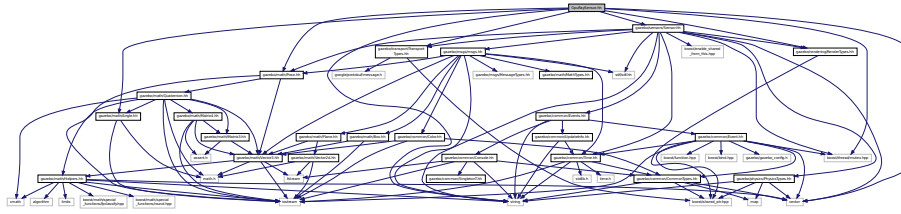
Rendering namespace.

- namespace **Ogre**

11.51 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 dependency graph for GpuRaySensor.hh:



Classes

- class **gazebo::sensors::GpuRaySensor**

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.52 Grid.hh File Reference

```
#include <stdint.h>
#include <vector>
#include <string>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/common/Color.hh"
```

Include dependency graph for Grid.hh:



Classes

- class **gazebo::rendering::Grid**
Displays a grid of cells, drawn with lines.

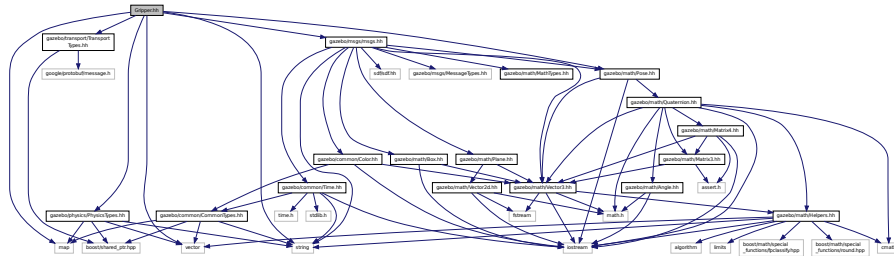
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

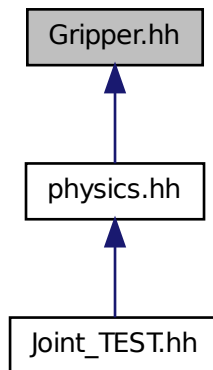
11.53 Gripper.hh File Reference

```
#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 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**

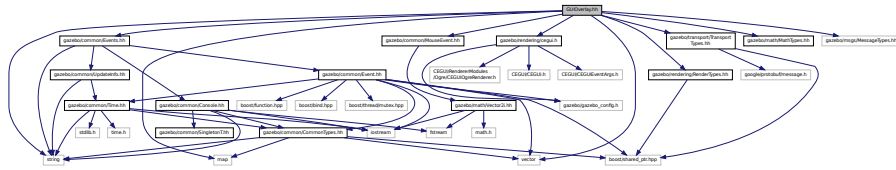
Forward declarations for the common classes.

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

11.54 GUIOverlay.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "gazebo/rendering/cegui.h"
#include "gazebo/common/MouseEvent.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/math/MathTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for GUIOverlay.hh:



Classes

- class **gazebo::rendering::GUIOverlay**
A class that creates a CEGUI overlay on a render window.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.55 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 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.

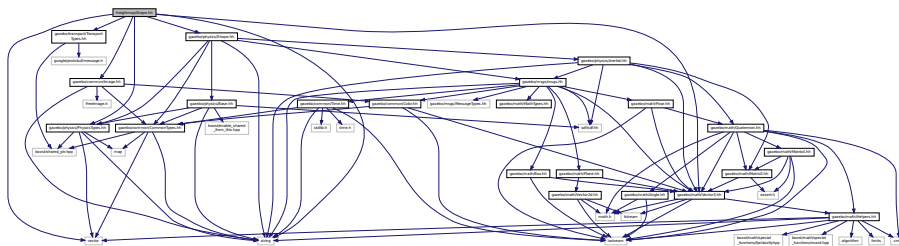
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

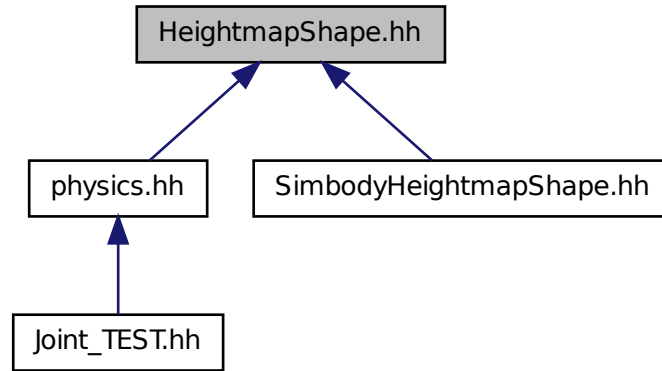
11.56 HeightmapShape.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/common/Image.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.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. 380) collision shape builds a heightmap from an image.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

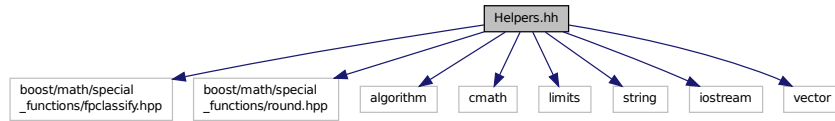
- namespace **gazebo::physics**

namespace for physics

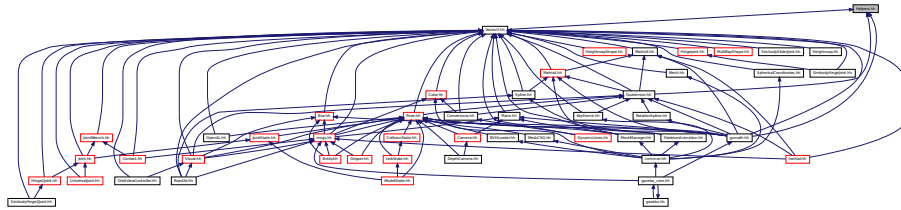
11.57 Helpers.hh File Reference

```
#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_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
- `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
- `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)

11.57.1 Macro Definition Documentation

11.57.1.1 `#define GZ_DBL_MAX std::numeric_limits<double>::max()`

Double maximum value.

11.57.1.2 `#define GZ_DBL_MIN std::numeric_limits<double>::min()`

Double min value.

11.57.1.3 `#define GZ_FLT_MAX std::numeric_limits<float>::max()`

Float maximum value.

11.57.1.4 `#define GZ_FLT_MIN std::numeric_limits<float>::min()`

Float minimum value.

11.57.1.5 `#define GZ_UINT32_MAX std::numeric_limits<uint32_t>::max()`

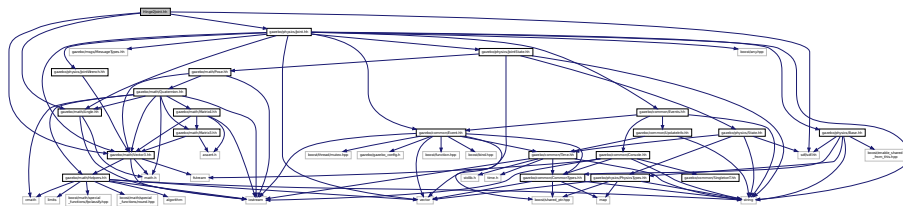
32bit unsigned integer maximum value

11.57.1.6 `#define GZ_UINT32_MIN std::numeric_limits<uint32_t>::min()`

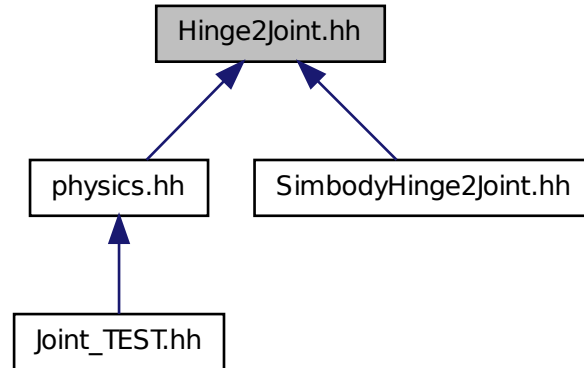
32bit unsigned integer minimum value

11.58 Hinge2Joint.hh File Reference

```
#include <sdf/sdf.hh>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/Joint.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.

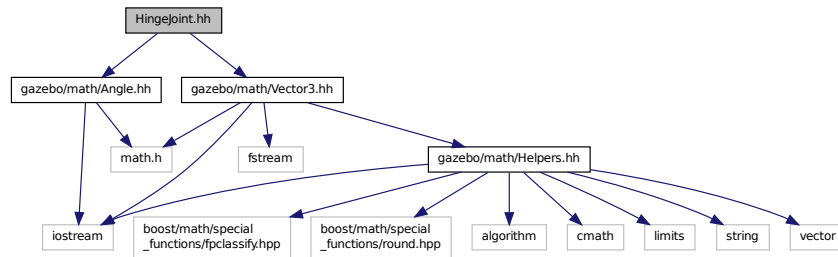
- namespace **gazebo::physics**

namespace for physics

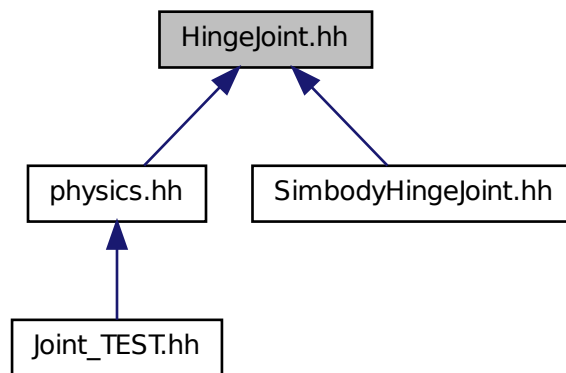
11.59 HingeJoint.hh File Reference

```
#include "gazebo/math/Angle.hh"  
#include "gazebo/math/Vector3.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.

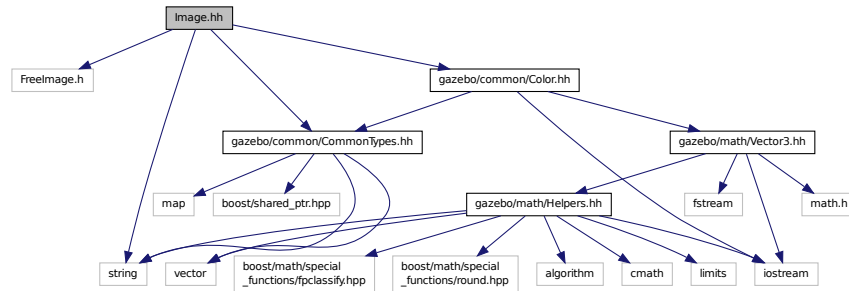
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

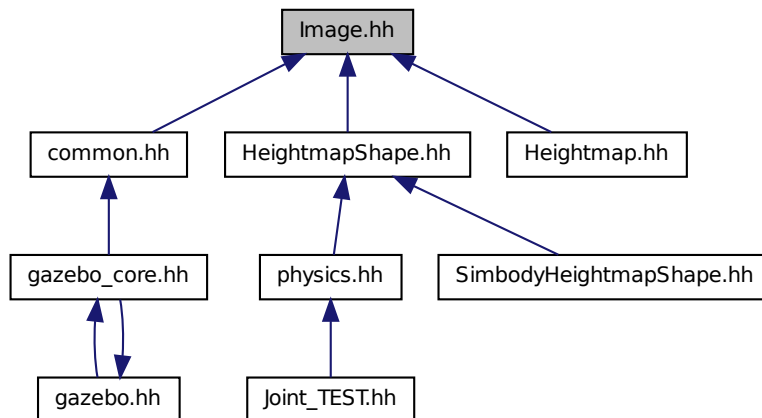
11.60 Image.hh File Reference

```
#include <FreeImage.h>
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/Color.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

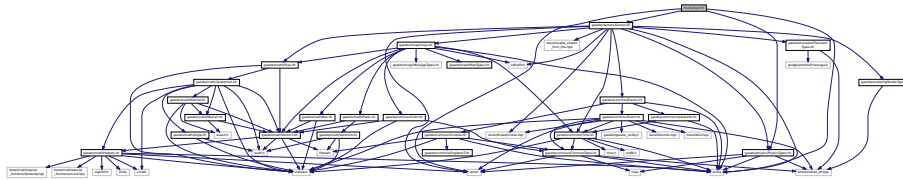
- static std::string **gazebo::common::PixelFormatNames []**

String names for the pixel formats.

11.61 ImuSensor.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/Sensor.hh"
```

Include dependency graph for ImuSensor.hh:



Classes

- class **gazebo::sensors::ImuSensor**

An IMU sensor.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

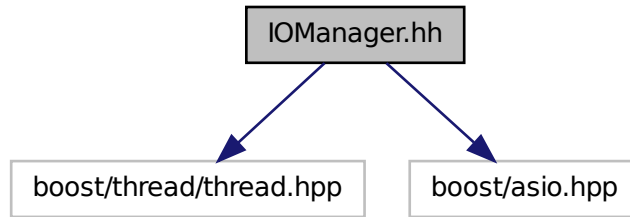
- namespace **gazebo::sensors**

Sensors namespace.

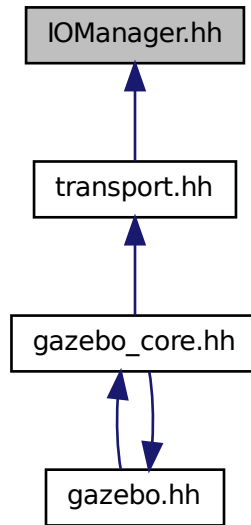
11.62 Inertial.hh File Reference

```
#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 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**

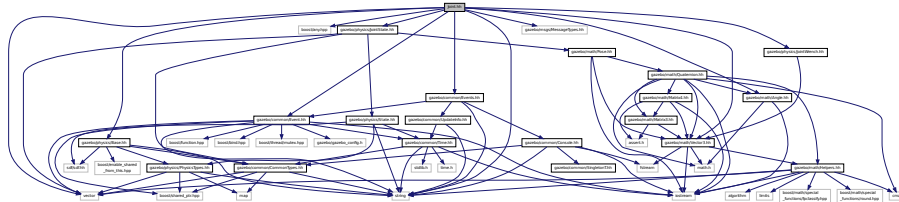
Forward declarations for the common classes.

- namespace **gazebo::transport**

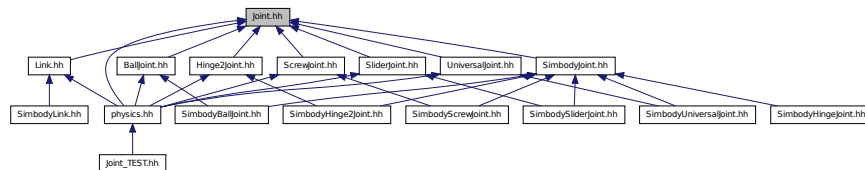
11.64 Joint.hh File Reference

```
#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/messages/MessageTypes.hh"
#include "gazebo/physics/JointState.hh"
#include "gazebo/physics/Base.hh"
#include "gazebo/physics/JointWrench.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. 153) 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.

11.64.1 Macro Definition Documentation

11.64.1.1 `#define MAX_JOINT_AXIS 2`

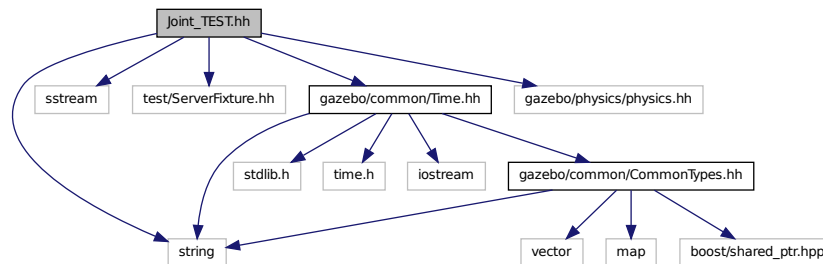
maximum number of axis per joint anticipated.

Currently, this is 2 as 3-axis joints (e.g. ball) actuation, control is not there yet.

11.65 Joint_TEST.hh File Reference

```
#include <string>
#include <sstream>
#include "test/ServerFixture.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/physics/physics.hh"
```

Include dependency graph for Joint_TEST.hh:



Classes

- class **Joint_TEST**
- class **Joint_TEST::SpawnJointOptions**
Class to hold parameters for spawning joints.

Typedefs

- typedef `std::tr1::tuple< const char *, const char * >` **std_string2**

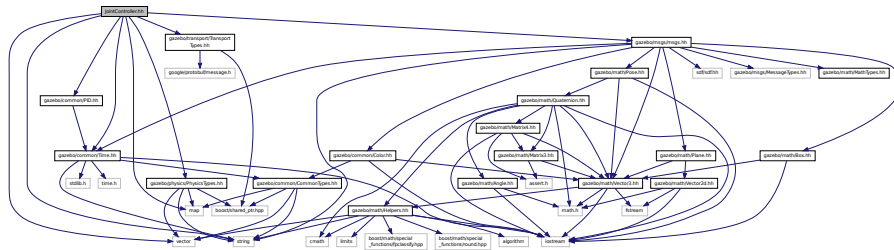
11.65.1 Typedef Documentation

11.65.1.1 `typedef std::tr1::tuple<const char *, const char *> std_string2`

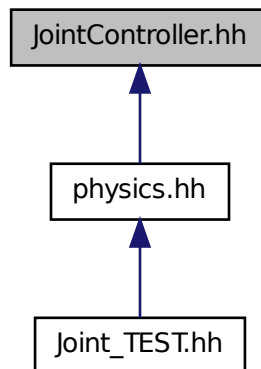
11.66 JointController.hh File Reference

```
#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 dependency graph for JointController.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class `gazebo::physics::JointController`

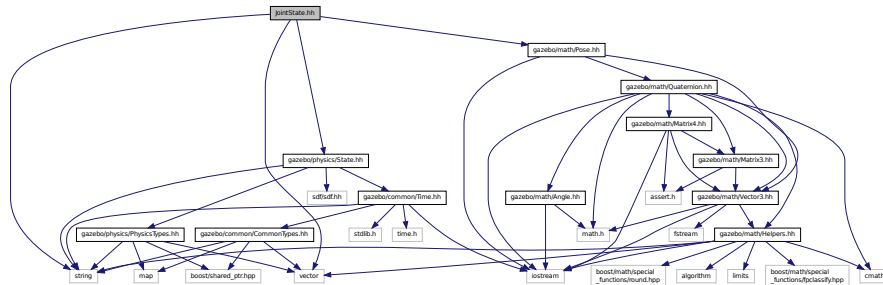
A class for manipulating `physics::Joint` (p. 411).

Namespaces

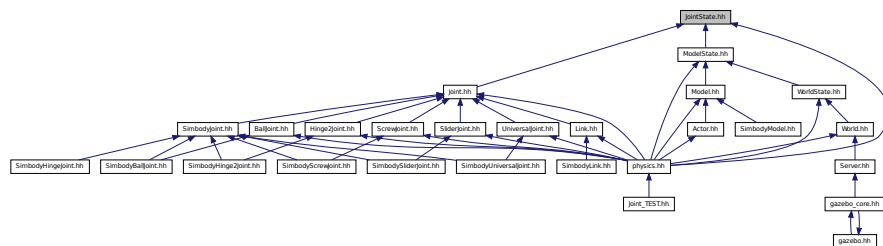
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.67 JointState.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/physics/State.hh"
#include "gazebo/math/Pose.hh"
Include dependency graph for JointState.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::JointState**
keeps track of state of a `physics::Joint` (p. 411)

Namespaces

- namespace **gazebo**

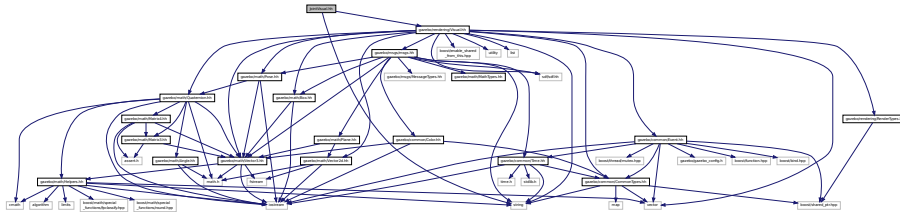
Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

11.68 JointVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
Include dependency graph for JointVisual.hh:
```



Classes

- class **gazebo::rendering::JointVisual**

Visualization for joints.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

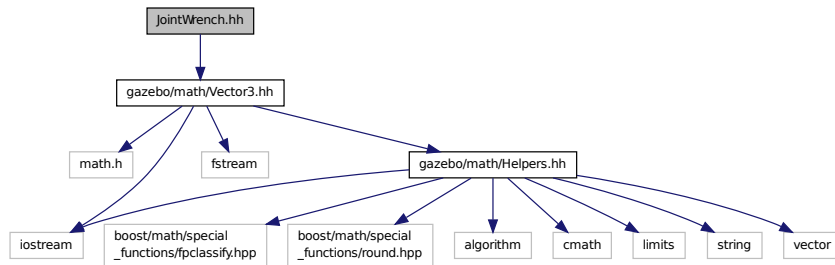
- namespace **gazebo::rendering**

Rendering namespace.

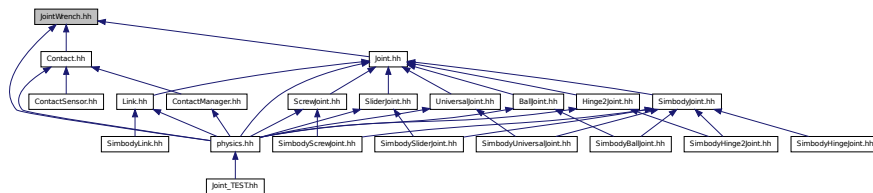
11.69 JointWrench.hh File Reference

```
#include "gazebo/math/Vector3.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**

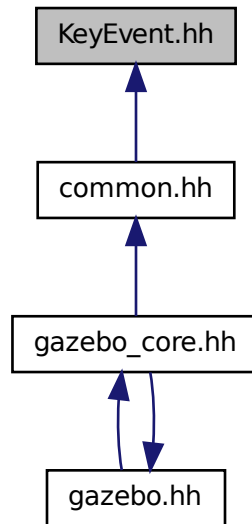
Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

11.70 KeyEvent.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::KeyEvent**
Generic description of a keyboard event.

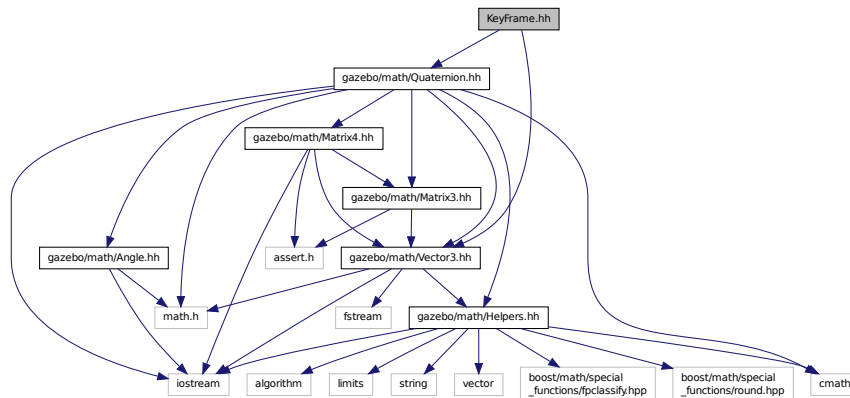
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

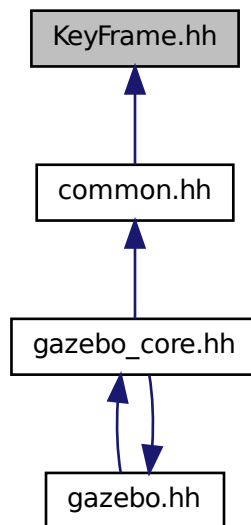
11.71 KeyFrame.hh File Reference

```
#include "gazebo/math/Vector3.hh"  
#include "gazebo/math/Quaternion.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. 606).

- class **gazebo::common::PoseKeyFrame**

A keyframe for a *PoseAnimation* (p. 657).

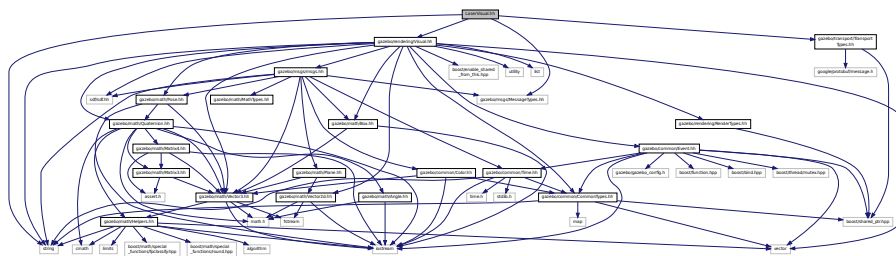
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.72 LaserVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for LaserVisual.hh:



Classes

- class **gazebo::rendering::LaserVisual**
Visualization for laser data.

Namespaces

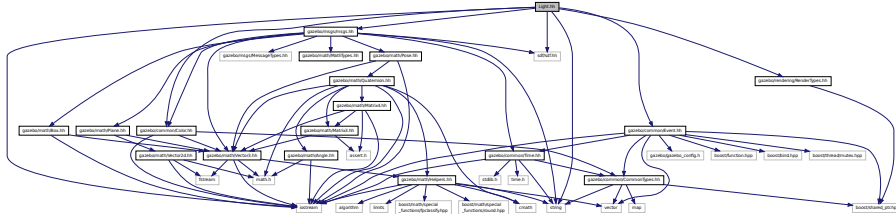
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.73 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 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**

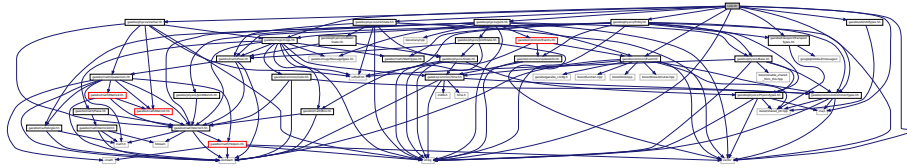
Rendering namespace.

- namespace **Ogre**

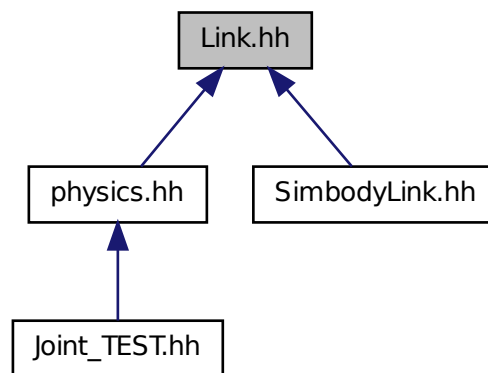
11.74 Link.hh File Reference

```
#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 dependency graph for Link.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Link**

Link (p. 455) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

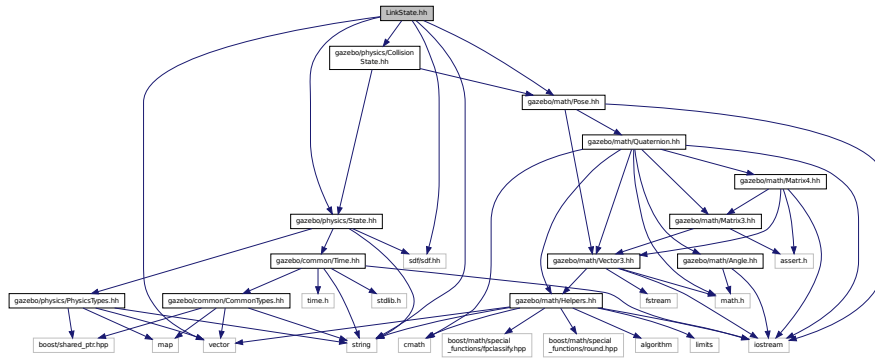
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics
- namespace **gazebo::util**

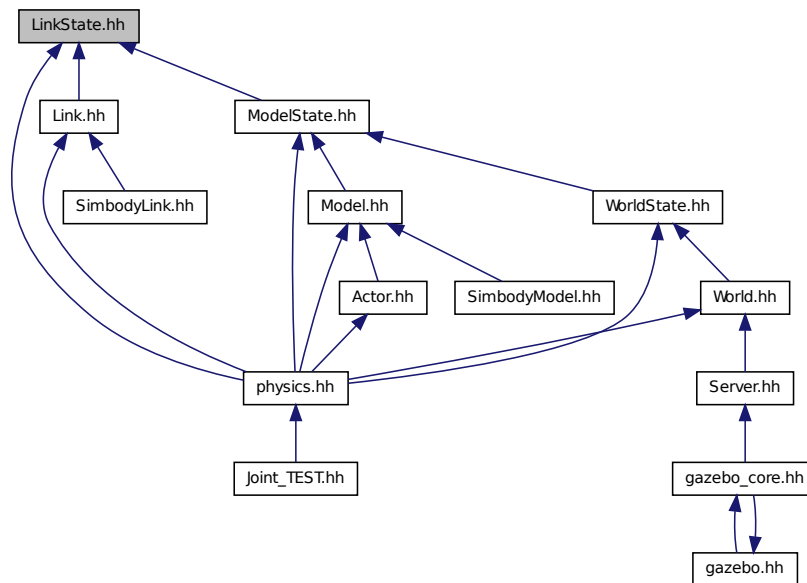
11.75 LinkState.hh File Reference

```
#include <vector>
```

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/physics/State.hh"
#include "gazebo/physics/CollisionState.hh"
#include "gazebo/math/Pose.hh"
Include 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. 455) object.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

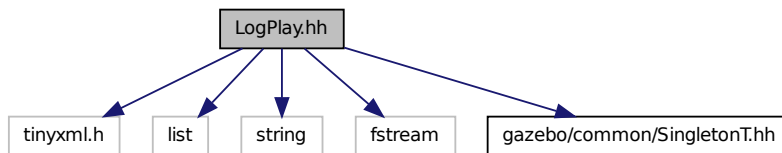
- namespace **gazebo::physics**

namespace for physics

11.76 LogPlay.hh File Reference

```
#include <tinyxml.h>
#include <list>
#include <string>
#include <fstream>
#include "gazebo/common/SingletonT.hh"
```

Include dependency graph for LogPlay.hh:



Classes

- class **gazebo::util::LogPlay**

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::util**

11.77 LogRecord.hh File Reference

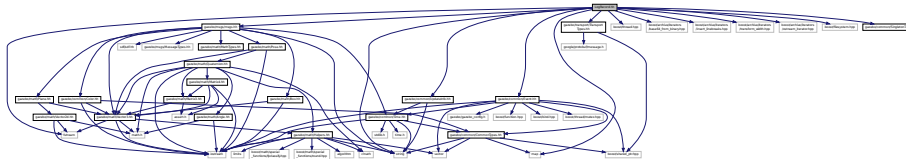
```
#include <fstream>
```

```

#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 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"**

11.77.1 Macro Definition Documentation

11.77.1.1 #define GZ_LOG_VERSION "1.0"

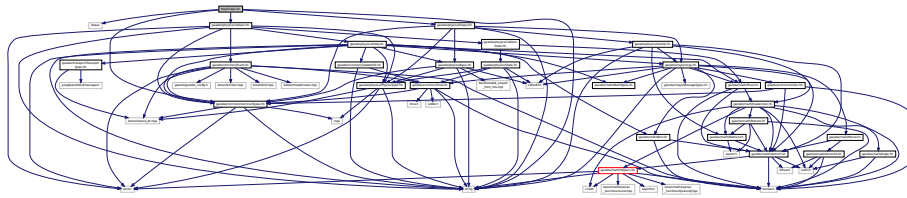
11.78 mainpage.html File Reference

11.79 MapShape.hh File Reference

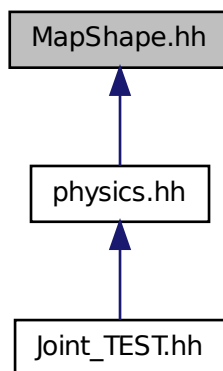
```
#include <deque>
```



```
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/physics/Shape.hh"
Include dependency graph for MapShape.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::MapShape**
Creates box extrusions based on an image.

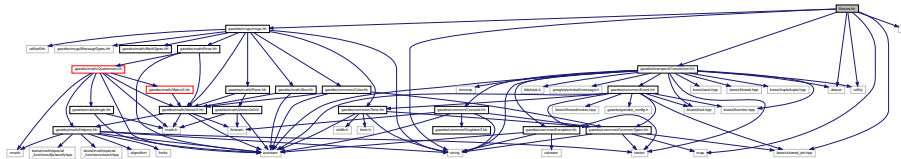
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

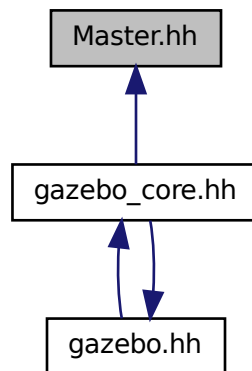
11.80 Master.hh File Reference

```
#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 dependency graph for Master.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::Master**

A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

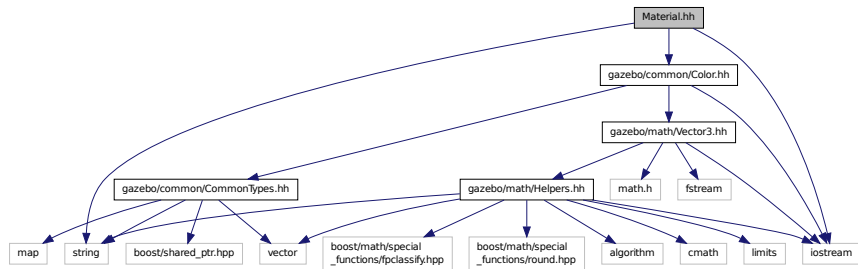
Namespaces

- namespace **gazebo**

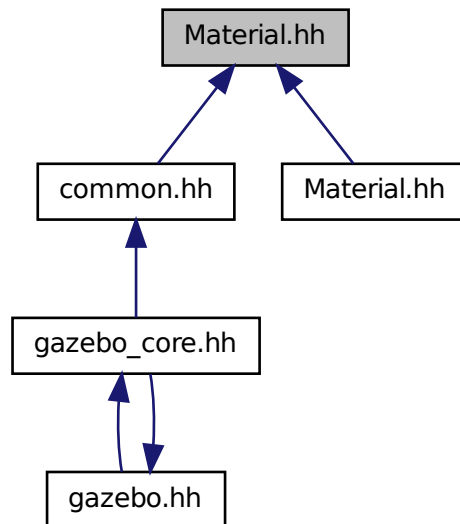
Forward declarations for the common classes.

11.81 Material.hh File Reference

```
#include <string>
#include <iostream>
#include "gazebo/common/Color.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.

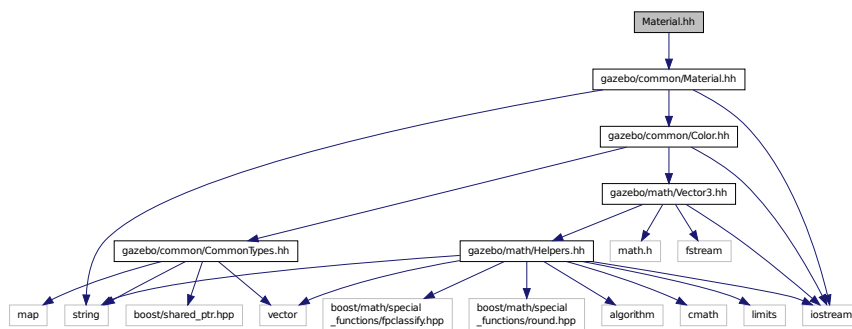
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.82 Material.hh File Reference

```
#include "gazebo/common/Material.hh"
```

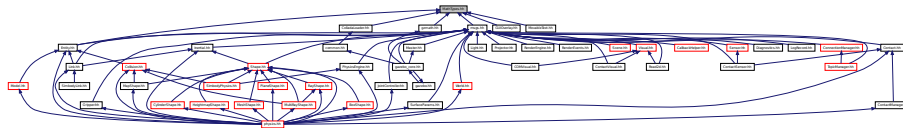
Include dependency graph for rendering/Material.hh:



11.83 MathTypes.hh File Reference

Forward declarations for the math classes.

This graph shows which files directly or indirectly include this file:



Namespaces

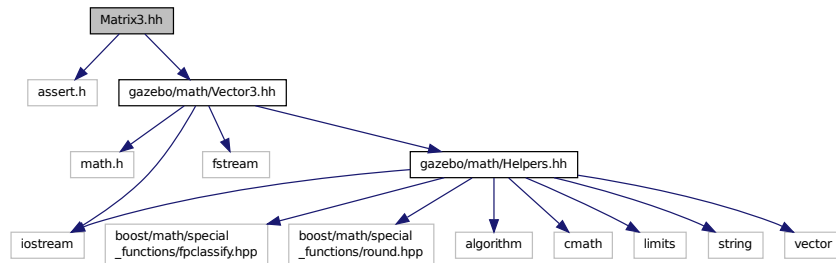
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

11.83.1 Detailed Description

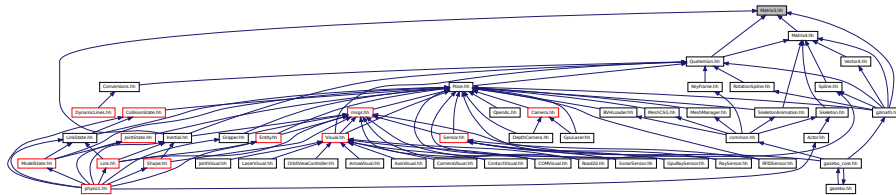
Forward declarations for the math classes.

11.84 Matrix3.hh File Reference

```
#include <assert.h>
#include "gazebo/math/Vector3.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.

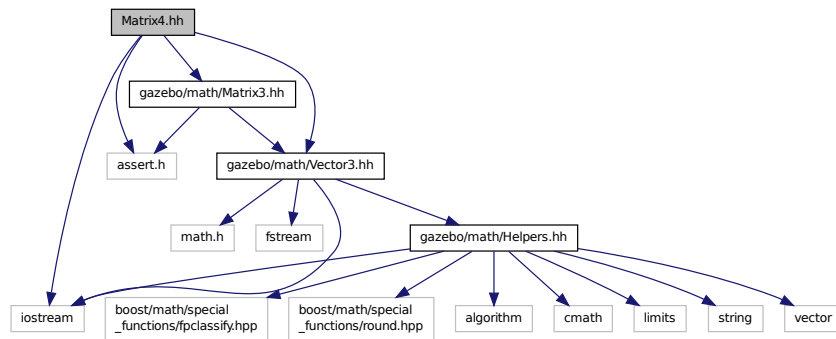
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

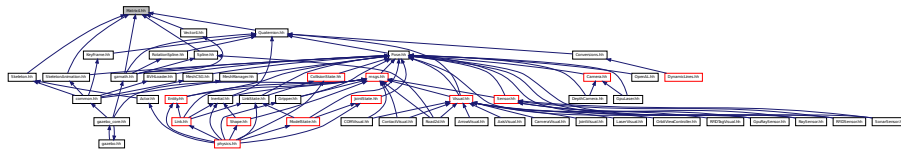
11.85 Matrix4.hh File Reference

```
#include <assert.h>
#include <iostream>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Matrix3.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**

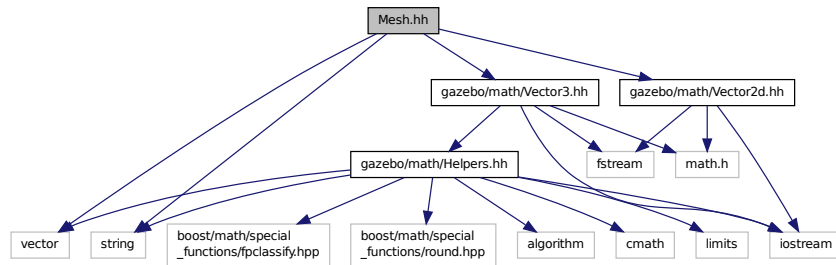
Math namespace.

11.86 Mesh.hh File Reference

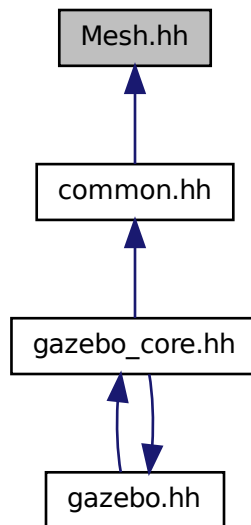
```

#include <vector>
#include <string>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2d.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.
- struct **gazebo::common::NodeAssignment**
Vertex to node weighted assignement for skeleton animation visualization.
- class **gazebo::common::SubMesh**
A child mesh.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

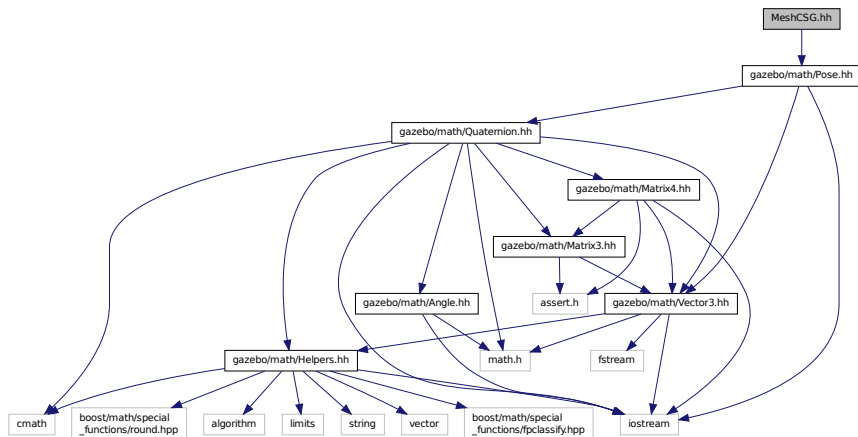
- namespace **gazebo::common**

Common namespace.

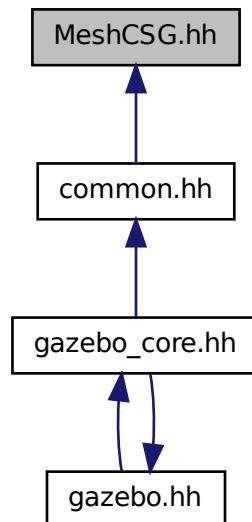
11.87 MeshCSG.hh File Reference

```
#include "gazebo/math/Pose.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**

11.87.1 Typedef Documentation

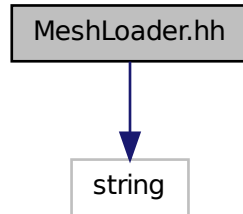
11.87.1.1 typedef `_GPtrArray` **GPtrArray**

11.87.1.2 typedef `_GtsSurface` **GtsSurface**

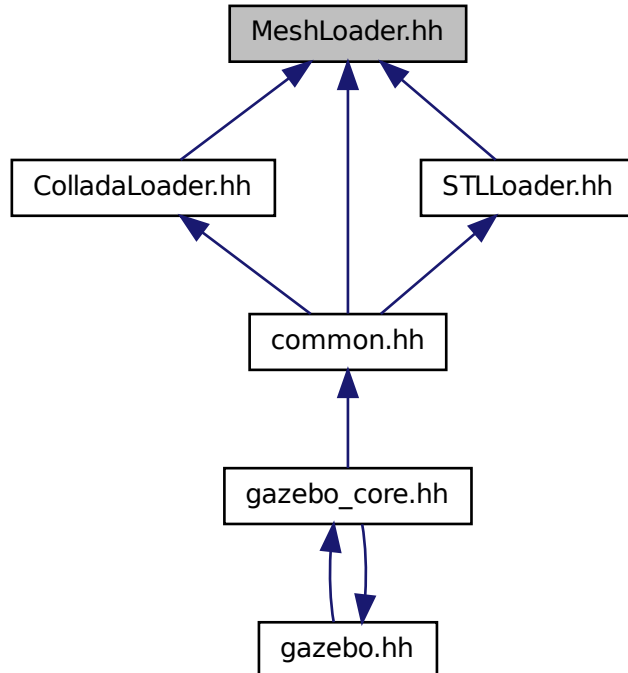
11.88 MeshLoader.hh File Reference

```
#include <string>
```

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.

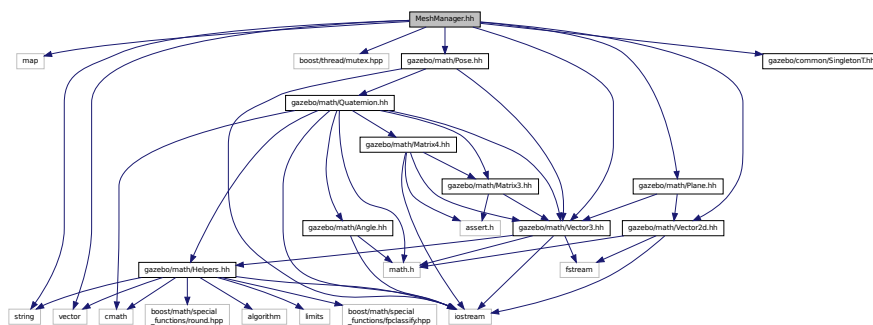
- namespace `gazebo::common`

Common namespace.

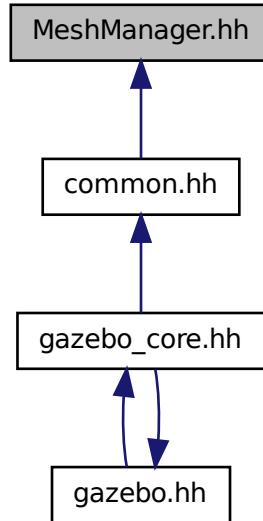
11.89 MeshManager.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include <boost/thread/mutex.hpp>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2d.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Plane.hh"
#include "gazebo/common/SingletonT.hh"
```

Include dependency graph for MeshManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::MeshManager**

Maintains and manages all meshes.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

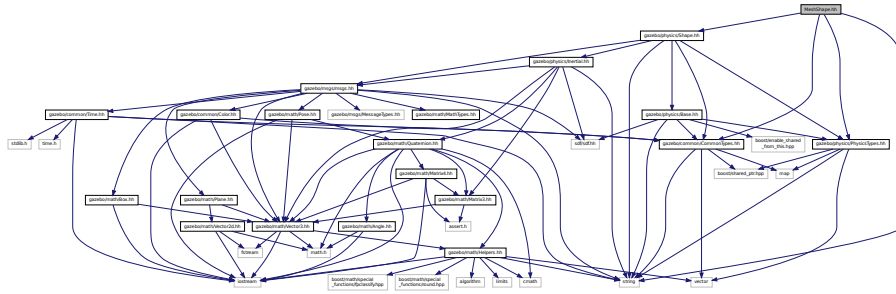
- namespace **gazebo::common**

Common namespace.

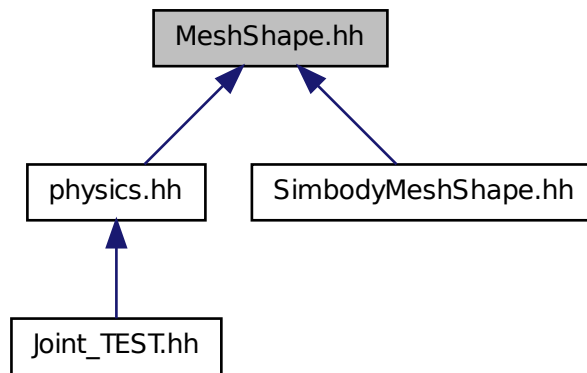
11.90 MeshShape.hh File Reference

```
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.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.

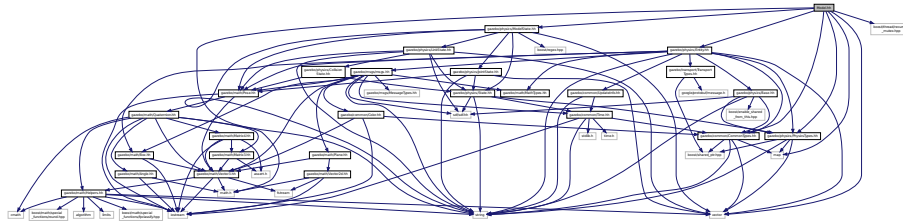
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

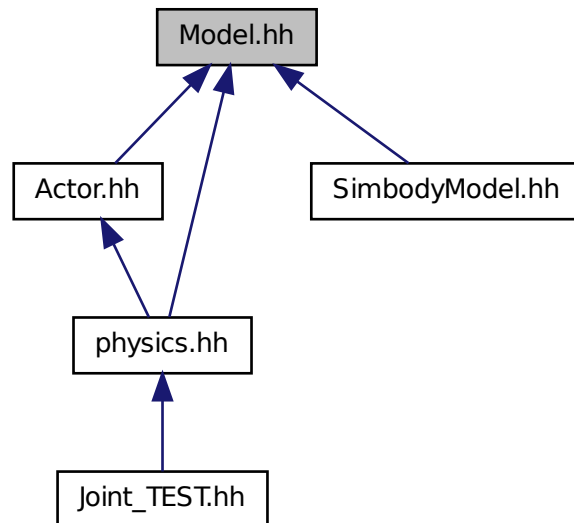
11.91 Model.hh File Reference

```
#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 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.

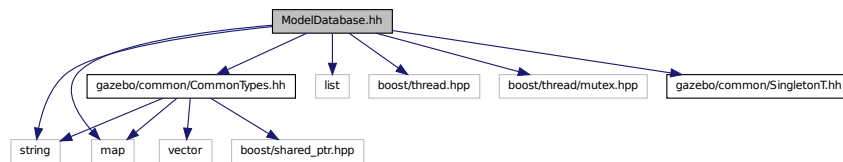
Namespaces

- namespace **boost**
- namespace **gazebo**
 - Forward declarations for the common classes.*
- namespace **gazebo::physics**
 - namespace for physics*

11.92 ModelDatabase.hh File Reference

```
#include <string>
#include <map>
#include <list>
#include <boost/thread.hpp>
#include <boost/thread/mutex.hpp>
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/CommonTypes.hh"
```

Include dependency graph for ModelDatabase.hh:



Classes

- class **gazebo::common::ModelDatabase**
 - Connects to model database, and has utility functions to find models.*

Namespaces

- namespace **gazebo**
 - Forward declarations for the common classes.*
- namespace **gazebo::common**
 - Common namespace.*

Macros

- #define **GZ_MODEL_DB_MANIFEST_FILENAME** "database.config"
 - The file name of model database XML configuration.*
- #define **GZ_MODEL_MANIFEST_FILENAME** "model.config"
 - The file name of model XML configuration.*

11.92.1 Macro Definition Documentation

11.92.1.1 #define GZ_MODEL_DB_MANIFEST_FILENAME "database.config"

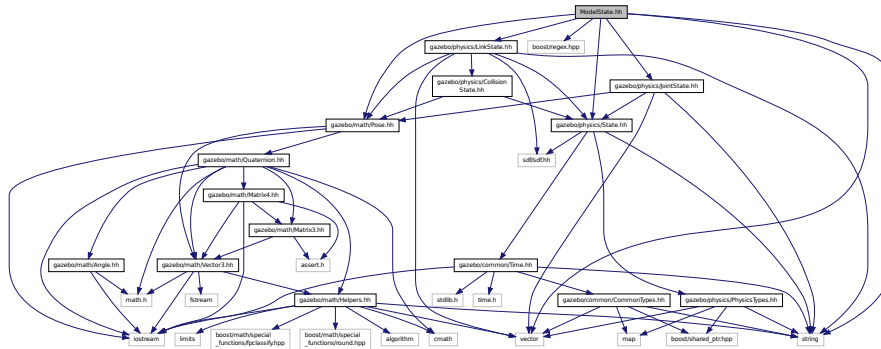
The file name of model database XML configuration.

11.92.1.2 #define GZ_MODEL_MANIFEST_FILENAME "model.config"

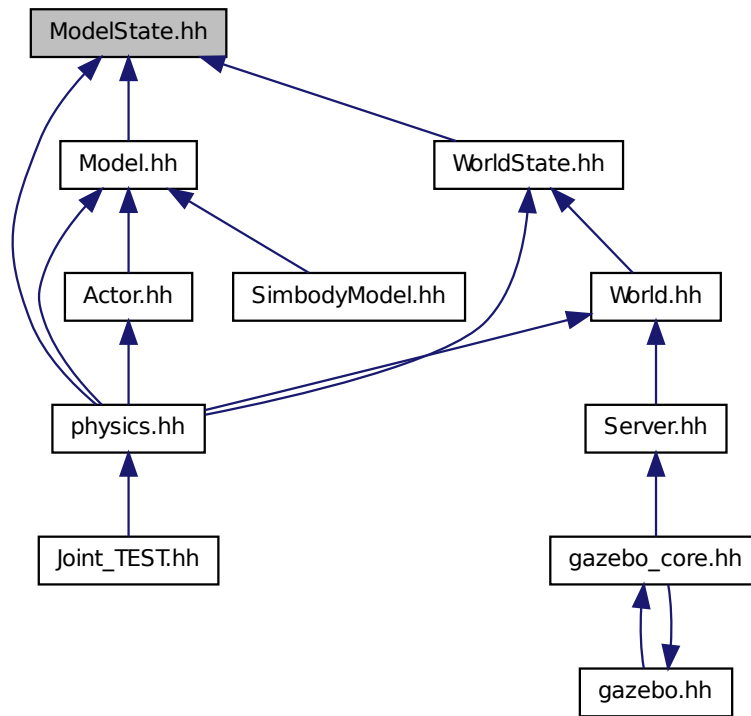
The file name of model XML configuration.

11.93 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 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. 537) object.

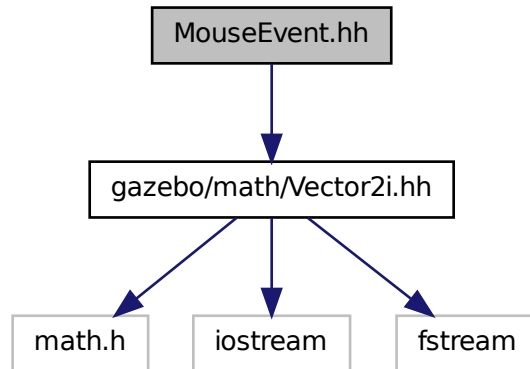
Namespaces

- namespace `gazebo`
Forward declarations for the common classes.
- namespace `gazebo::physics`
namespace for physics

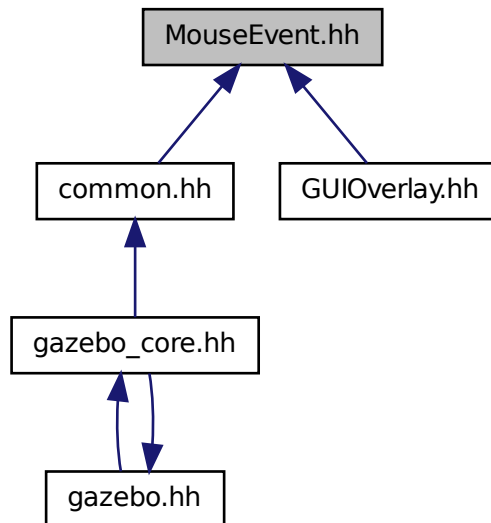
11.94 MouseEvent.hh File Reference

```
#include "gazebo/math/Vector2i.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.

11.95 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 dependency graph for MovableText.hh:



Classes

- class **gazebo::rendering::MovableText**

Movable text.

Namespaces

- namespace **boost**
- namespace **gazebo**

Forward declarations for the common classes.

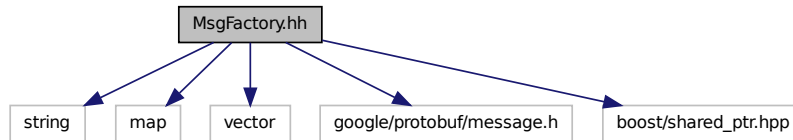
- namespace **gazebo::rendering**

Rendering namespace.

11.96 MsgFactory.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include <google/protobuf/message.h>
#include <boost/shared_ptr.hpp>
```

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**)()

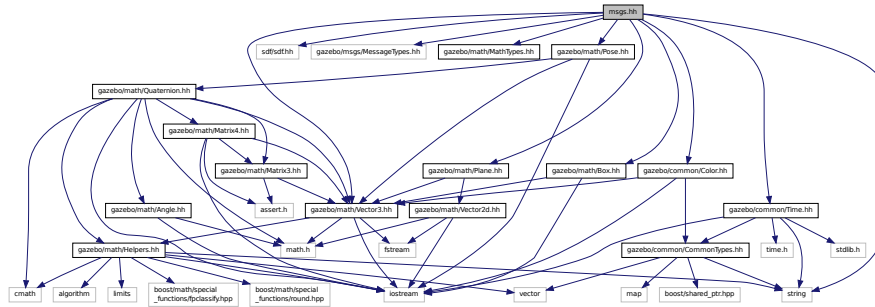
11.97 msgs.hh File Reference

```

#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/Color.hh"
#include "gazebo/common/Time.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

- `msgs::Vector3d gazebo::msgs::Convert` (const `math::Vector3` &_v)
*Convert a **math::Vector3** (p. 1004) to a `msgs::Vector3d`.*
- `msgs::Quaternion gazebo::msgs::Convert` (const `math::Quaternion` &_q)
*Convert a **math::Quaternion** (p. 675) to a `msgs::Quaternion`.*
- `msgs::Pose gazebo::msgs::Convert` (const `math::Pose` &_p)
*Convert a **math::Pose** (p. 648) to a `msgs::Pose`.*
- `msgs::Color gazebo::msgs::Convert` (const `common::Color` &_c)
*Convert a **common::Color** (p. 226) to a `msgs::Color`.*
- `msgs::Time gazebo::msgs::Convert` (const `common::Time` &_t)
*Convert a **common::Time** (p. 944) to a `msgs::Time`.*
- `msgs::PlaneGeom gazebo::msgs::Convert` (const `math::Plane` &_p)
*Convert a **math::Plane** (p. 640) to a `msgs::PlaneGeom`.*
- `math::Vector3 gazebo::msgs::Convert` (const `msgs::Vector3d` &_v)
*Convert a `msgs::Vector3d` to a **math::Vector**.*
- `math::Quaternion gazebo::msgs::Convert` (const `msgs::Quaternion` &_q)
*Convert a `msgs::Quaternion` to a **math::Quaternion** (p. 675).*
- `math::Pose gazebo::msgs::Convert` (const `msgs::Pose` &_p)

- Convert a `msgs::Pose` to a `math::Pose` (p. 648).*

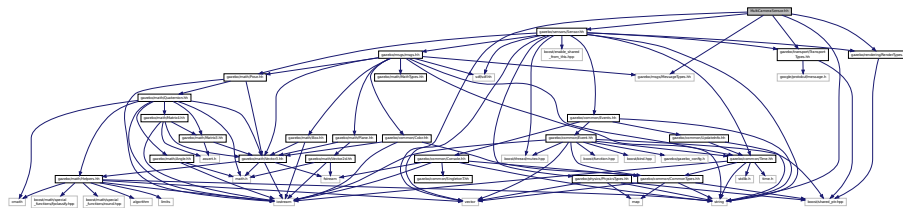
 - common::Color **gazebo::msgs::Convert** (const msgs::Color &_c)
 - Convert a `msgs::Color` to a `common::Color` (p. 226).*
 - common::Time **gazebo::msgs::Convert** (const msgs::Time &_t)
 - Convert a `msgs::Time` to a `common::Time` (p. 944).*
 - math::Plane **gazebo::msgs::Convert** (const msgs::PlaneGeom &_p)
 - Convert a `msgs::PlaneGeom` to a `common::Plane`.*
- msgs::Request * **gazebo::msgs::CreateRequest** (const std::string &_request, const std::string &_data="")
 - Create a request message.*
- msgs::Fog **gazebo::msgs::FogFromSDF** (sdf::ElementPtr _sdf)
 - Create a `msgs::Fog` from a fog SDF element.*
- msgs::Geometry **gazebo::msgs::GeometryFromSDF** (sdf::ElementPtr _sdf)
 - Create a `msgs::Geometry` from a geometry SDF element.*
- msgs::Header * **gazebo::msgs::GetHeader** (google::protobuf::Message &_message)
 - Get the header from a protobuf message.*
- msgs::GUI **gazebo::msgs::GUIFromSDF** (sdf::ElementPtr _sdf)
 - Create a `msgs::GUI` from a GUI SDF element.*
- void **gazebo::msgs::Init** (google::protobuf::Message &_message, const std::string &_id="")
 - Initialize a message.*
- msgs::Light **gazebo::msgs::LightFromSDF** (sdf::ElementPtr _sdf)
 - Create a `msgs::Light` from a light SDF element.*
- msgs::MeshGeom **gazebo::msgs::MeshFromSDF** (sdf::ElementPtr _sdf)
 - Create a `msgs::MeshGeom` from a mesh SDF element.*
- msgs::Scene **gazebo::msgs::SceneFromSDF** (sdf::ElementPtr _sdf)
 - Create a `msgs::Scene` from a scene SDF element.*
- void **gazebo::msgs::Set** (common::Image &_img, const msgs::Image &_msg)
 - Convert a `msgs::Image` to a `common::Image` (p. 389).*
- void **gazebo::msgs::Set** (msgs::Image *_msg, const common::Image &_i)
 - Set a `msgs::Image` from a `common::Image` (p. 389).*
- void **gazebo::msgs::Set** (msgs::Vector3d *_pt, const math::Vector3 &_v)
 - Set a `msgs::Vector3d` from a `math::Vector3` (p. 1004).*
- void **gazebo::msgs::Set** (msgs::Vector2d *_pt, const math::Vector2d &_v)
 - Set a `msgs::Vector2d` from a `math::Vector3` (p. 1004).*
- void **gazebo::msgs::Set** (msgs::Quaternion *_q, const math::Quaternion &_v)
 - Set a `msgs::Quaternion` from a `math::Quaternion` (p. 675).*
- void **gazebo::msgs::Set** (msgs::Pose *_p, const math::Pose &_v)
 - Set a `msgs::Pose` from a `math::Pose` (p. 648).*
- void **gazebo::msgs::Set** (msgs::Color *_c, const common::Color &_v)
 - Set a `msgs::Color` from a `common::Color` (p. 226).*
- void **gazebo::msgs::Set** (msgs::Time *_t, const common::Time &_v)
 - Set a `msgs::Time` from a `common::Time` (p. 944).*
- void **gazebo::msgs::Set** (msgs::PlaneGeom *_p, const math::Plane &_v)
 - Set a `msgs::Plane` from a `math::Plane` (p. 640).*
- void **gazebo::msgs::Stamp** (msgs::Header *_header)
 - Time stamp a header.*
- void **gazebo::msgs::Stamp** (msgs::Time *_time)
 - Set the time in a time message.*

- msgs::TrackVisual **gazebo::msgs::TrackVisualFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::TrackVisual from a track visual SDF element.
- msgs::Visual **gazebo::msgs::VisualFromSDF** (sdf::ElementPtr _sdf)
Create a msgs::Visual from a visual SDF element.

11.98 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 dependency graph for MultiCameraSensor.hh:



Classes

- class **gazebo::sensors::MultiCameraSensor**
Multiple camera sensor.

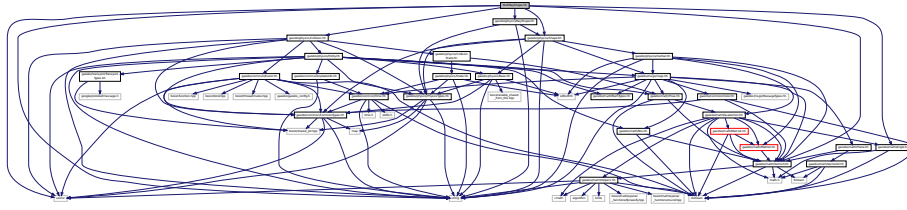
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

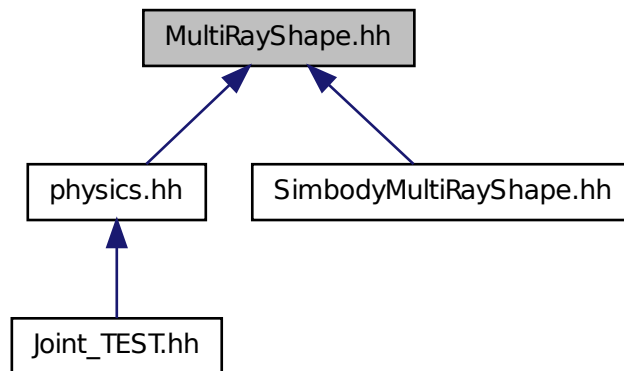
11.99 MultiRayShape.hh File Reference

```
#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 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

11.100 Node.hh File Reference

```
#include <tbb/task.h>
```

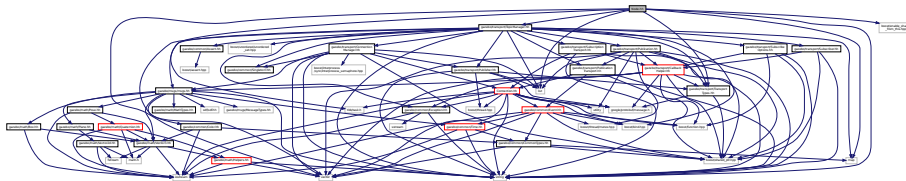


```

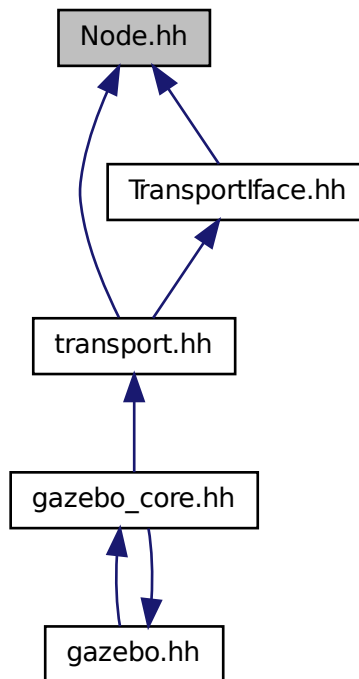
#include <boost/enable_shared_from_this.hpp>
#include <map>
#include <list>
#include <string>
#include <vector>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/transport/TopicManager.hh"

```

Include dependency graph for Node.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Node**

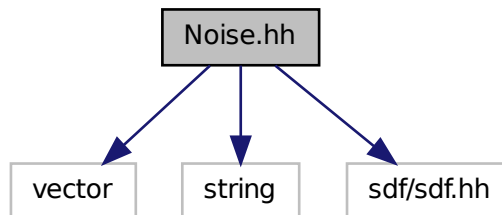
A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

11.101 Noise.hh File Reference

```
#include <vector>
#include <string>
#include <sdf/sdf.hh>
Include dependency graph for Noise.hh:
```



Classes

- class **gazebo::sensors::Noise**
Noise (p. 603) models for sensor output signals.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.102 ogre_gazebo.h File Reference

```
#include <OGRE/Ogre.h>
```

```

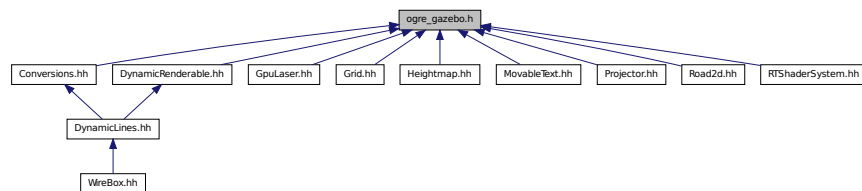
#include <OGRE/OgreImageCodec.h>
#include <OGRE/OgreMovableObject.h>
#include <OGRE/OgreRenderable.h>
#include <OGRE/OgrePlugin.h>
#include <OGRE/OgreDataStream.h>
#include <OGRE/OgreLogManager.h>
#include <OGRE/OgreWindowEventUtilities.h>
#include <OGRE/OgreSceneQuery.h>
#include <OGRE/OgreRoot.h>
#include <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/OgreFontManager.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 dependency graph for ogre_gazebo.h:



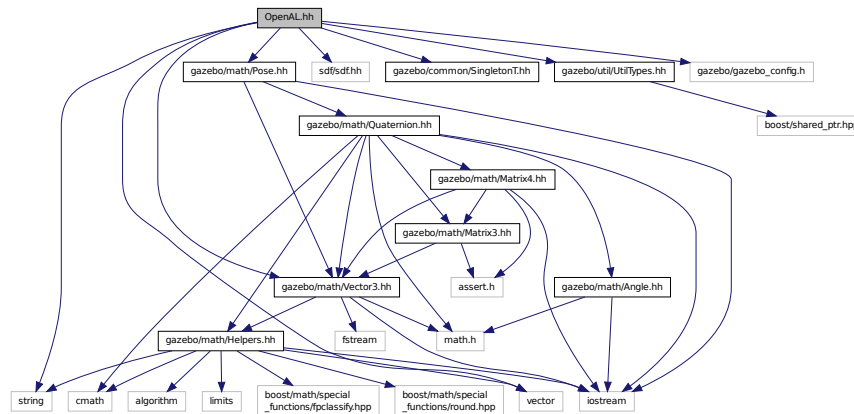
This graph shows which files directly or indirectly include this file:



11.103 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 dependency graph for OpenAL.hh:



Classes

- class **gazebo::util::OpenAL**
3D audio setup and playback.
- class **gazebo::util::OpenALSink**
OpenAL (p. 609) Listener.
- class **gazebo::util::OpenALSource**
OpenAL (p. 609) Source.

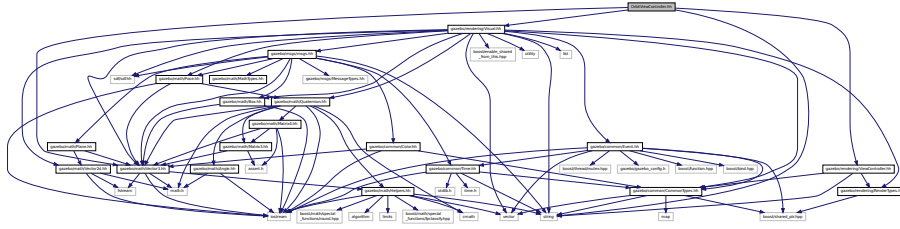
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::util**

11.104 OrbitViewController.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/rendering/ViewController.hh"
#include "gazebo/math/Vector3.hh"
```

Include dependency graph for OrbitViewController.hh:



Classes

- class **gazebo::rendering::OrbitViewController**

Orbit view controller.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

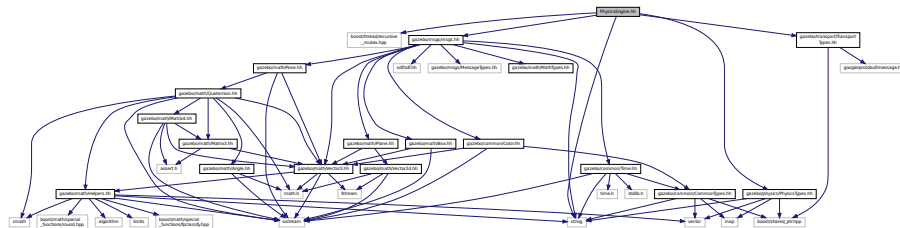
- namespace **gazebo::rendering**

Rendering namespace.

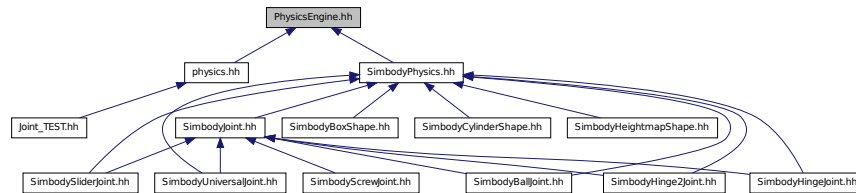
11.105 PhysicsEngine.hh File Reference

```
#include <boost/thread/recursive_mutex.hpp>
#include <string>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for PhysicsEngine.hh:



This graph shows which files directly or indirectly include this file:



Classes

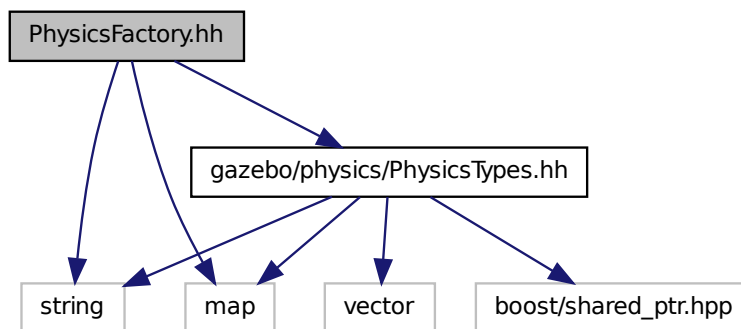
- class **gazebo::physics::PhysicsEngine**
Base (p. 153) class for a physics engine.

Namespaces

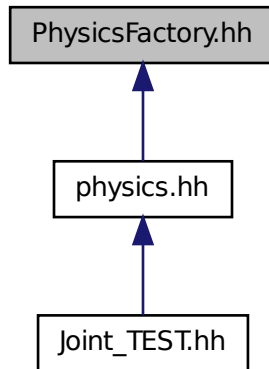
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.106 PhysicsFactory.hh File Reference

```
#include <string>
#include <map>
#include "gazebo/physics/PhysicsTypes.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

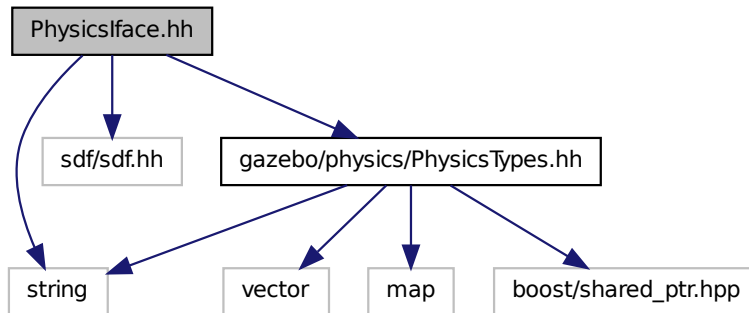
- typedef PhysicsEnginePtr(* **gazebo::physics::PhysicsFactoryFn**)(WorldPtr world)

11.107 PhysicsIface.hh File Reference

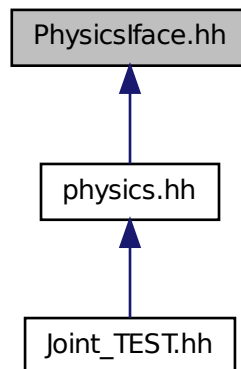
```

#include <string>
#include <sdf/sdf.hh>
#include "gazebo/physics/PhysicsTypes.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

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

Create a world given a name.

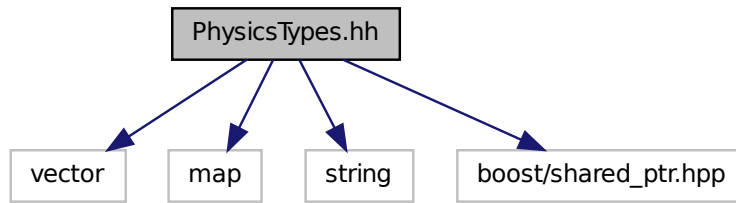
- bool **gazebo::physics::fini** ()
*Finalize transport by calling **gazebo::transport::fini** (p. 87).*
- WorldPtr **gazebo::physics::get_world** (const std::string &_name="")
Returns a pointer to a world by name.
- uint32_t **gazebo::physics::getUniqueld** ()
Get a unique ID.
- void **gazebo::physics::init_world** (WorldPtr _world)
Init world given a pointer to it.
- void **gazebo::physics::init_worlds** ()
initialize multiple worlds stored in static variable gazebo::g_worlds
- bool **gazebo::physics::load** ()
*Setup **gazebo::SystemPlugin** (p. 942)'s and call **gazebo::transport::init** (p. 88).*
- void **gazebo::physics::load_world** (WorldPtr _world, sdf::ElementPtr _sdf)
Load world from sdf::Element pointer.
- void **gazebo::physics::load_worlds** (sdf::ElementPtr _sdf)
load multiple worlds from single sdf::Element pointer
- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)
*Pause world by calling **World::SetPaused** (p. 1080).*
- void **gazebo::physics::pause_worlds** (bool pause)
pause multiple worlds stored in static variable gazebo::g_worlds
- void **gazebo::physics::remove_worlds** ()
remove multiple worlds stored in static variable gazebo::g_worlds
- void **gazebo::physics::run_world** (WorldPtr _world, unsigned int _iterations=0)
*Run world by calling **World::Run()** (p. 1080) given a pointer to it.*
- void **gazebo::physics::run_worlds** (unsigned int _iterations=0)
Run multiple worlds stored in static variable gazebo::g_worlds.
- void **gazebo::physics::stop_world** (WorldPtr _world)
*Stop world by calling **World::Stop()** (p. 1081) given a pointer to it.*
- void **gazebo::physics::stop_worlds** ()
stop multiple worlds stored in static variable gazebo::g_worlds
- bool **gazebo::physics::worlds_running** ()
Return true if any world is running.

11.108 PhysicsTypes.hh File Reference

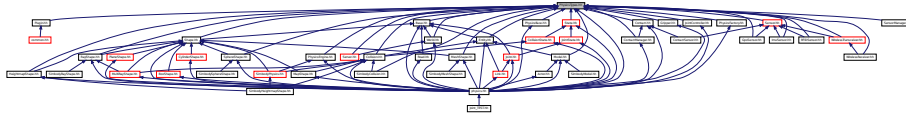
default namespace for gazebo

```
#include <vector>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```

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**

11.108.1 Detailed Description

default namespace for gazebo

11.108.2 Macro Definition Documentation

11.108.2.1 `#define GZ_ALL_COLLIDE 0x0FFFFFFF`

Default collision bitmask.

Collision objects will collide with everything.

11.108.2.2 `#define GZ_FIXED_COLLIDE 0x00000001`

Collision object will collide only with fixed objects.

11.108.2.3 `#define GZ_GHOST_COLLIDE 0x10000000`

Collides with everything else but other ghost.

11.108.2.4 `#define GZ_NONE_COLLIDE 0x00000000`

Collision object will collide with nothing.

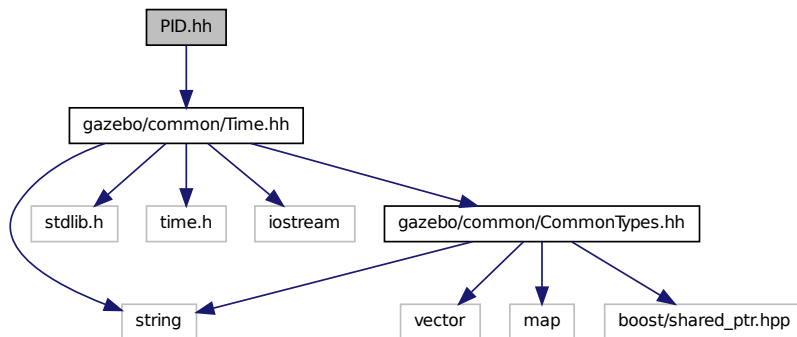
11.108.2.5 `#define GZ_SENSOR_COLLIDE 0x00000002`

Collision object will collide only with sensors.

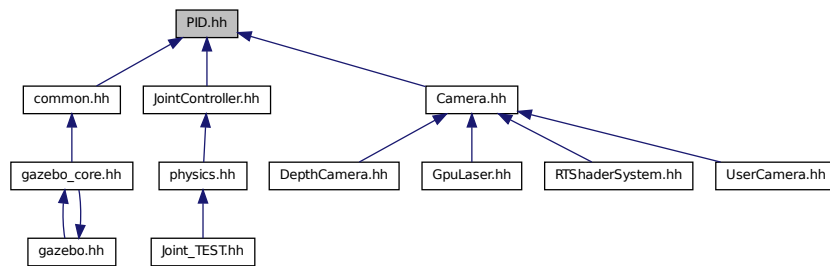
11.109 PID.hh File Reference

```
#include "gazebo/common/Time.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. 636) controller class.*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

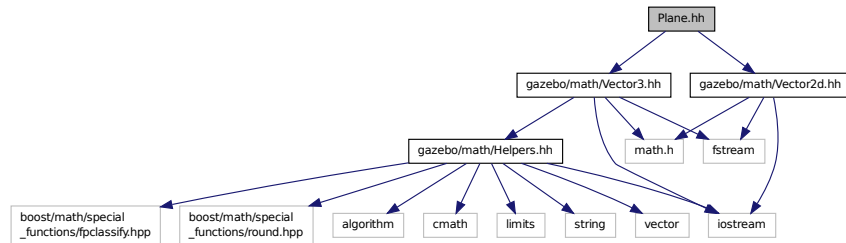
- namespace **gazebo::common**

Common namespace.

11.110 Plane.hh File Reference

```
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2d.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**

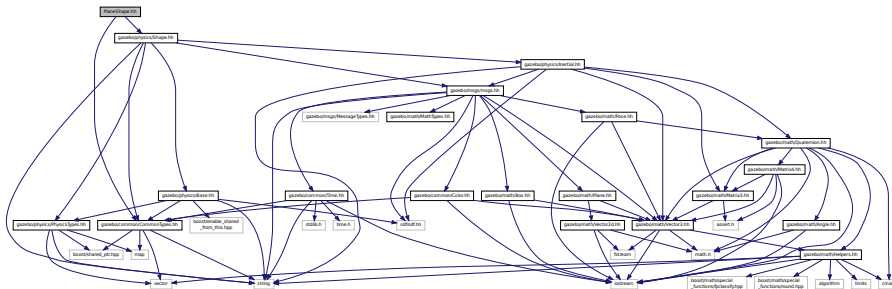
Math namespace.

11.111 PlaneShape.hh File Reference

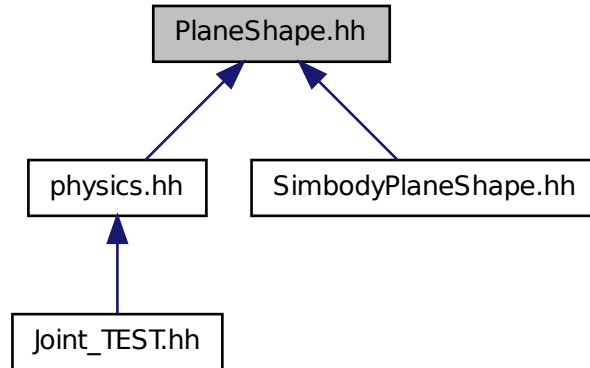
```
#include "gazebo/common/CommonTypes.hh"
```

```
#include "gazebo/physics/Shape.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. 213) for an infinite plane.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

11.112 Plugin.hh File Reference

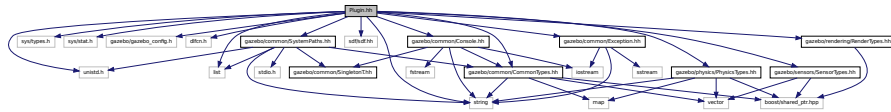
```
#include <unistd.h>
```

```

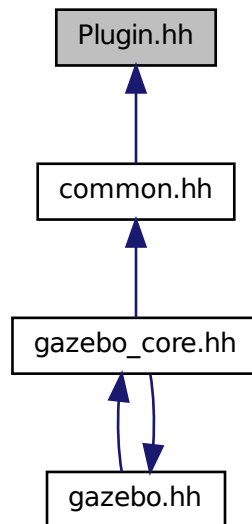
#include <sys/types.h>
#include <sys/stat.h>
#include <gazebo/gazebo_config.h>
#include <dlfcn.h>
#include <list>
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/SystemPaths.hh"
#include "gazebo/common/Console.hh"
#include "gazebo/common/Exception.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/SensorTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"

```

Include dependency graph for Plugin.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::ModelPlugin**

- A plugin with access to **physics::Model** (p. 537).
- 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. 1070).

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.

11.112.1 Macro Definition Documentation

11.112.1.1 #define GZ_REGISTER_MODEL_PLUGIN(classname)

Value:

```
extern "C" gazebo::ModelPlugin *RegisterPlugin();    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

11.112.1.2 #define GZ_REGISTER_SENSOR_PLUGIN(*classname*)

Value:

```
extern "C" gazebo::SensorPlugin *RegisterPlugin();    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

11.112.1.3 #define GZ_REGISTER_SYSTEM_PLUGIN(*classname*)

Value:

```
extern "C" gazebo::SystemPlugin *RegisterPlugin();    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

11.112.1.4 #define GZ_REGISTER_VISUAL_PLUGIN(*classname*)

Value:

```
extern "C" gazebo::VisualPlugin *RegisterPlugin();    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

11.112.15 #define GZ_REGISTER_WORLD_PLUGIN(*classname*)**Value:**

```
extern "C" gazebo::WorldPlugin *RegisterPlugin();    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.

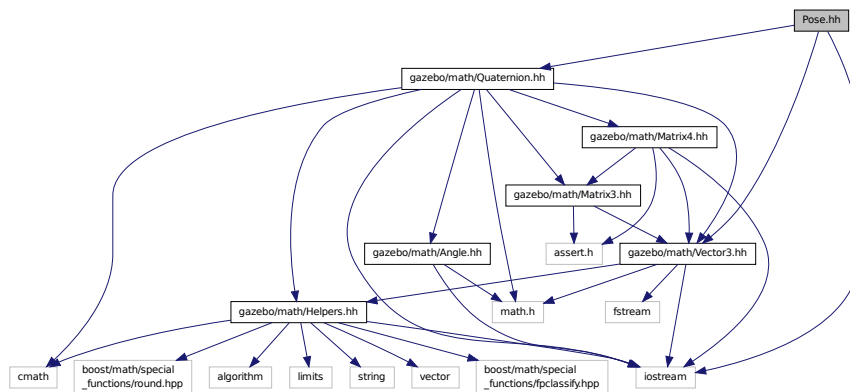
Returns

the name of the registered plugin

11.113 Pose.hh File Reference

```
#include <iostream>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Quaternion.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.

11.114 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 dependency graph for Projector.hh:



Classes

- class **gazebo::rendering::Projector**
Projects a material onto surface, light a light projector.

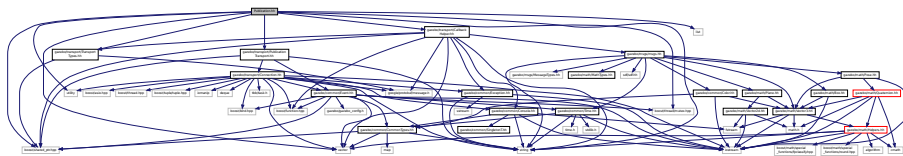
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

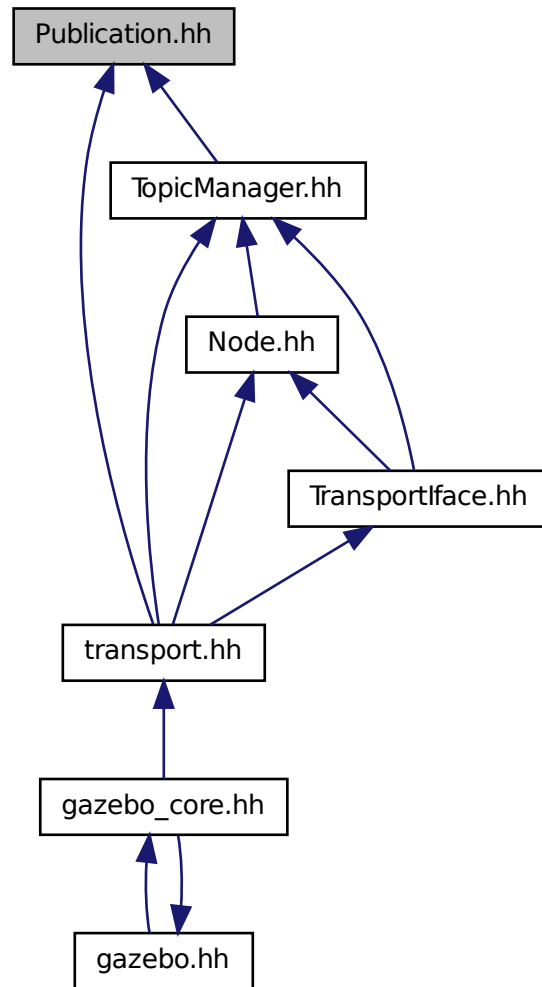
11.115 Publication.hh File Reference

```
#include <utility>
```

```
#include <boost/shared_ptr.hpp>
#include <boost/thread/mutex.hpp>
#include <list>
#include <string>
#include <vector>
#include "gazebo/transport/CallbackHelper.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/transport/PublicationTransport.hh"
Include dependency graph for Publication.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

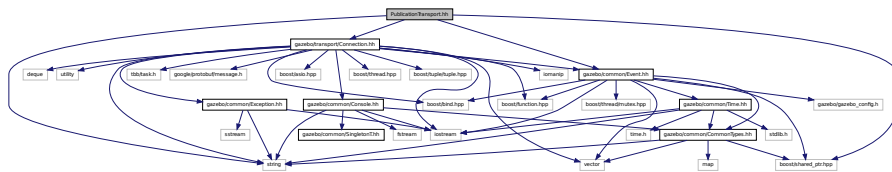
- class **gazebo::transport::Publication**
A publication for a topic.

Namespaces

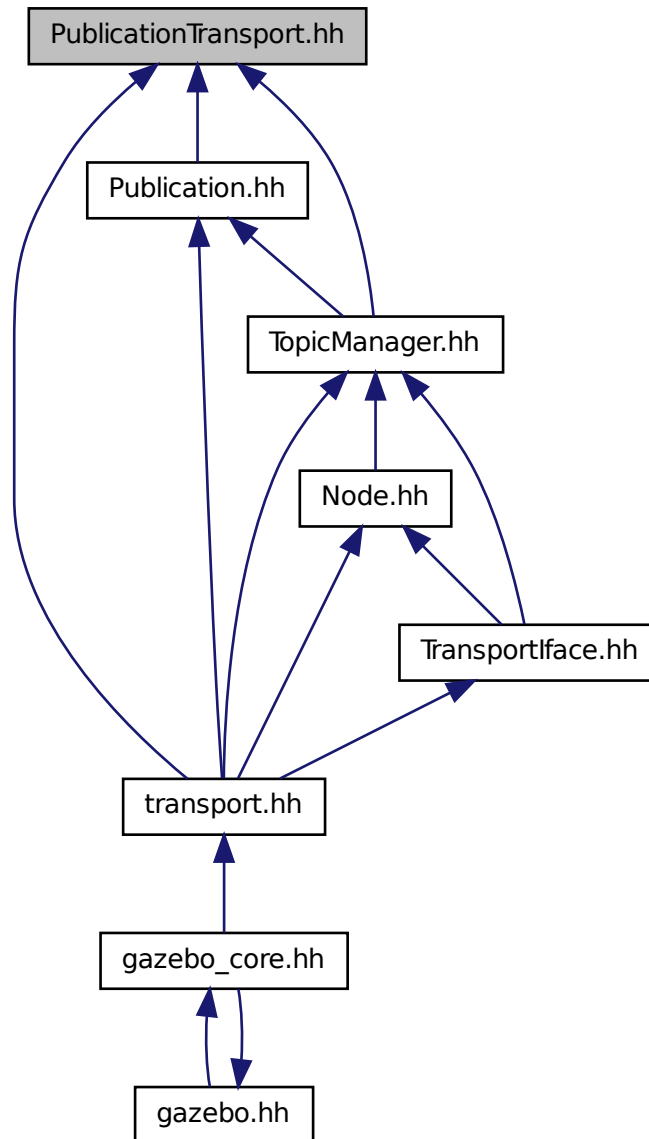
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

11.116 PublicationTransport.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <string>
#include "gazebo/transport/Connection.hh"
#include "gazebo/common/Event.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**

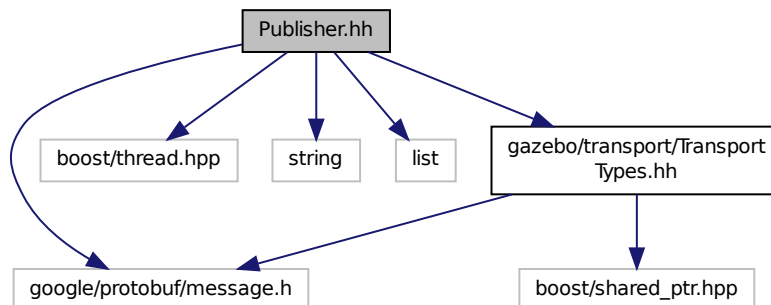
Forward declarations for the common classes.

- namespace **gazebo::transport**

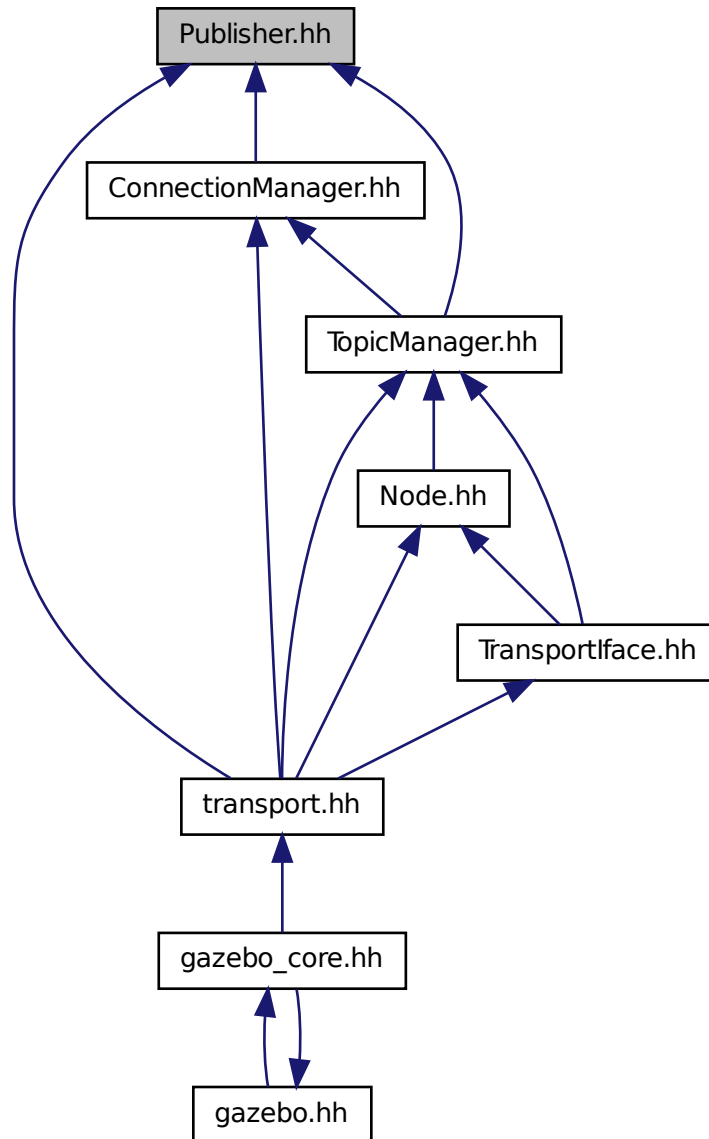
11.117 Publisher.hh File Reference

```
#include <google/protobuf/message.h>
#include <boost/thread.hpp>
#include <string>
#include <list>
#include "gazebo/transport/TransportTypes.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.

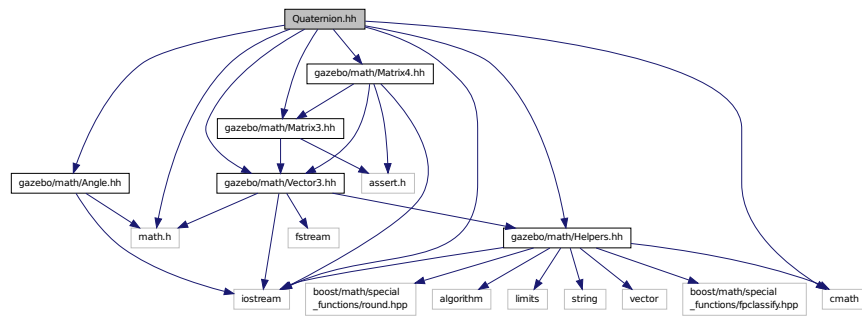
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

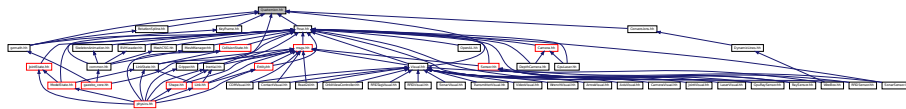
11.118 Quaternion.hh File Reference

```
#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 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

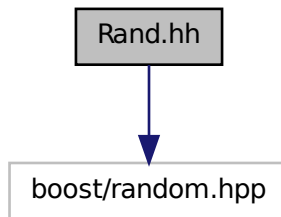
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**

Math namespace.

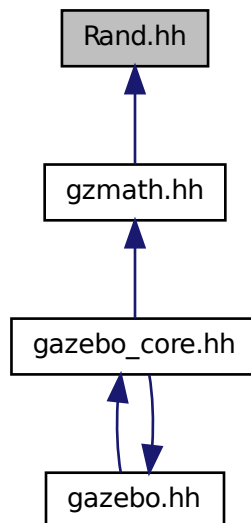
11.119 Rand.hh File Reference

```
#include <boost/random.hpp>
```

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.

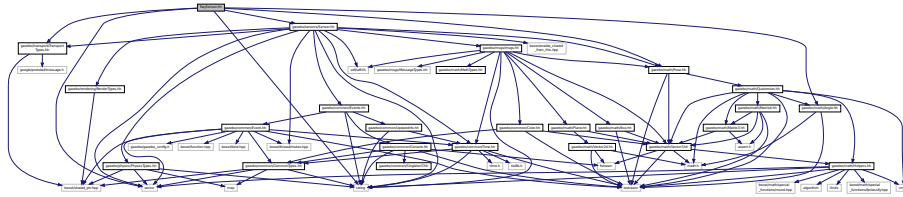
Typedefs

- 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**

11.120 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 dependency graph for RaySensor.hh:



Classes

- class **gazebo::sensors::RaySensor**
Sensor (p. 751) with one or more rays.

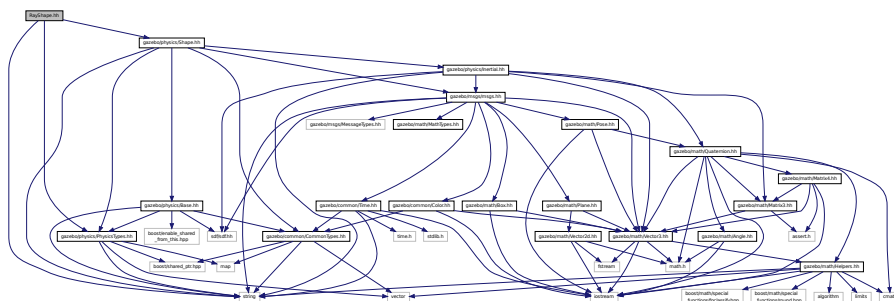
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

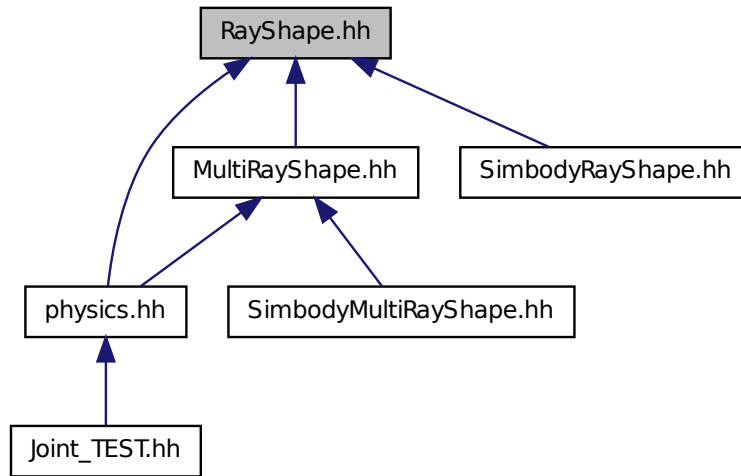
11.121 RayShape.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.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. 153) class for Ray collision geometry.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

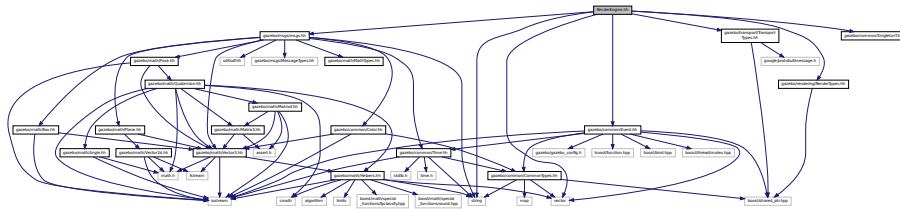
11.122 RenderEngine.hh File Reference

```

#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 dependency graph for RenderEngine.hh:



Classes

- class **gazebo::rendering::RenderEngine**
Adaptor to Ogre3d.

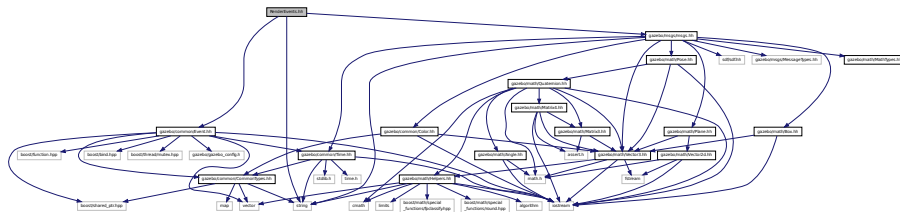
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.123 RenderEvents.hh File Reference

```
#include <string>
#include "gazebo/common/Event.hh"
#include "gazebo/msgs/msgs.hh"
```

Include dependency graph for RenderEvents.hh:



Classes

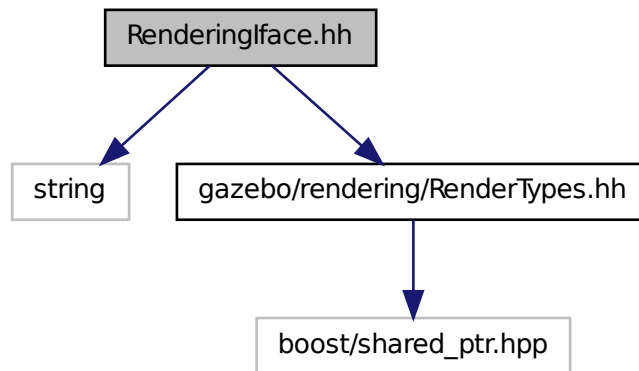
- class **gazebo::rendering::Events**
Base class for rendering events.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.124 Renderingface.hh File Reference

```
#include <string>
#include "gazebo/rendering/RenderTypes.hh"
Include dependency graph for Renderingface.hh:
```



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

Functions

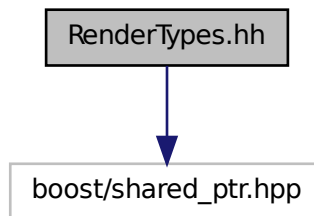
- rendering::ScenePtr **gazebo::rendering::create_scene** (const std::string &_name, bool _enableVisualizations, bool _isServer=false)
*create **rendering::Scene** (p. 728) by name.*
- bool **gazebo::rendering::fini** ()
teardown rendering engine.
- rendering::ScenePtr **gazebo::rendering::get_scene** (const std::string &_name="")
*get pointer to **rendering::Scene** (p. 728) by name.*

- bool **gazebo::rendering::init** ()
init rendering engine.
- bool **gazebo::rendering::load** ()
load rendering engine.
- void **gazebo::rendering::remove_scene** (const std::string &_name)
*remove a **rendering::Scene** (p. 728) by name*

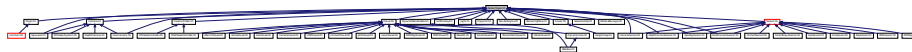
11.125 RenderTypes.hh File Reference

```
#include <boost/shared_ptr.hpp>
```

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.

11.125.1 Macro Definition Documentation

11.125.1.1 #define GZ_VISIBILITY_ALL 0xFFFFFFFF

Render everything visibility mask.

11.125.1.2 #define GZ_VISIBILITY_GUI 0x00000001

Render GUI visuals mask.

11.125.1.3 #define GZ_VISIBILITY_SELECTABLE 0x00000002

Render visuals that are selectable mask.

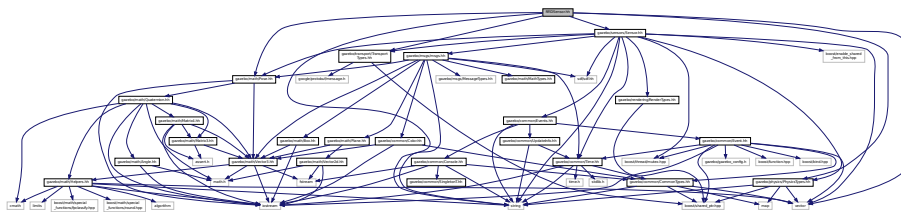
11.125.1.4 #define GZ_VISIBILITY_SELECTION 0x10000000

Renders only objects that can be selected.

11.126 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 dependency graph for RFIDSensor.hh:



Classes

- class **gazebo::sensors::RFIDSensor**
Sensor (p. 751) class for RFID type of sensor.

Namespaces

- namespace **gazebo**

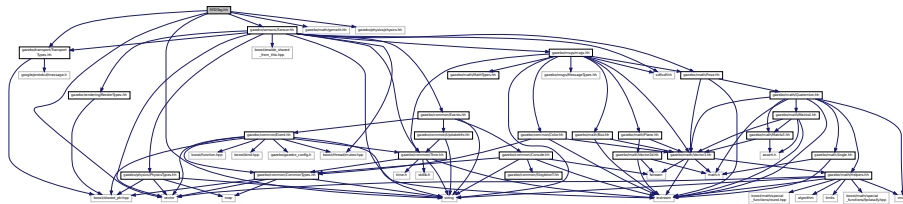
Forward declarations for the common classes.

- namespace **gazebo::sensors**

Sensors namespace.

11.127 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 dependency graph for RFIDTag.hh:
```



Classes

- class **gazebo::sensors::RFIDTag**

RFIDTag (p. 712) to interact with *RFIDTagSensors*.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

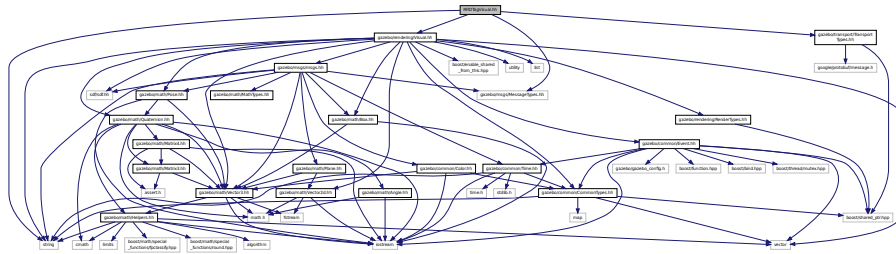
- namespace **gazebo::sensors**

Sensors namespace.

11.128 RFIDTagVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for RFIDTagVisual.hh:



Classes

- class **gazebo::rendering::RFIDTagVisual**
Visualization for RFID tags sensor.

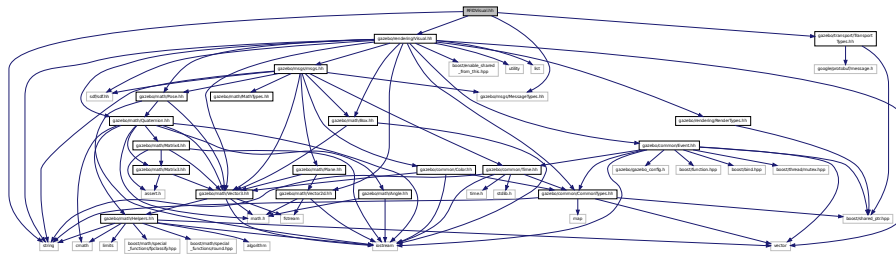
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.129 RFIDVisual.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for RFIDVisual.hh:



Classes

- class **gazebo::rendering::RFIDVisual**
Visualization for RFID sensor.

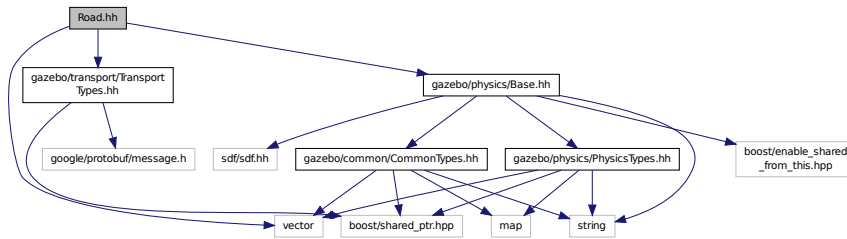
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

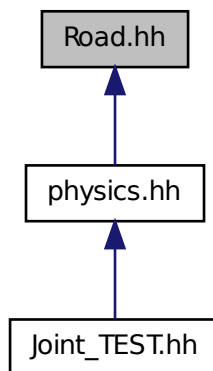
11.130 Road.hh File Reference

```
#include <vector>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/Base.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. 718) from SDF

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.131 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 dependency graph for Road2d.hh:



Classes

- class **gazebo::rendering::Road2d**

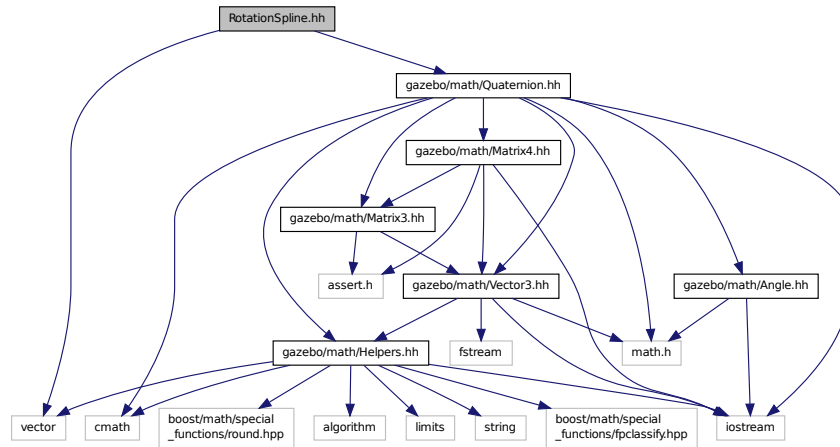
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

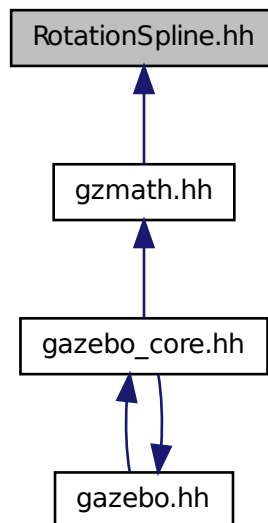
11.132 RotationSpline.hh File Reference

```
#include <vector>
#include "gazebo/math/Quaternion.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. 906) for rotations.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::math**

Math namespace.

11.133 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 dependency graph for RTShaderSystem.hh:



Classes

- class **gazebo::rendering::RTShaderSystem**

*Implements **Ogre** (p. 123)'s Run-Time Shader system.*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

Rendering namespace.

11.134 Scene.hh File Reference

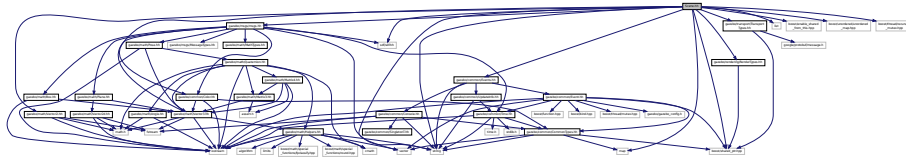
```
#include <vector>
```

```

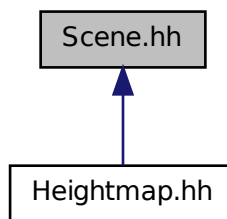
#include <map>
#include <string>
#include <list>
#include <boost/enable_shared_from_this.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/unordered/unordered_map.hpp>
#include <boost/thread/recursive_mutex.hpp>
#include <sdf/sdf.hh>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/common/Color.hh"
#include "gazebo/math/Vector2i.hh"

```

Include 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

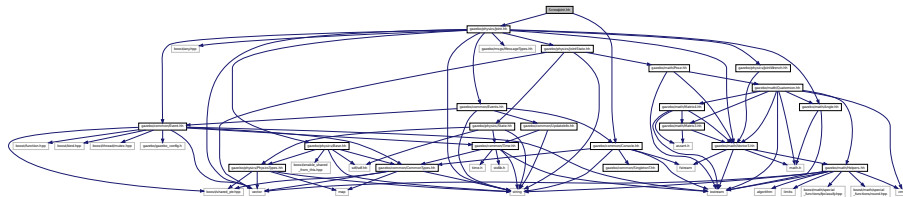
- namespace **boost**
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**

Rendering namespace.

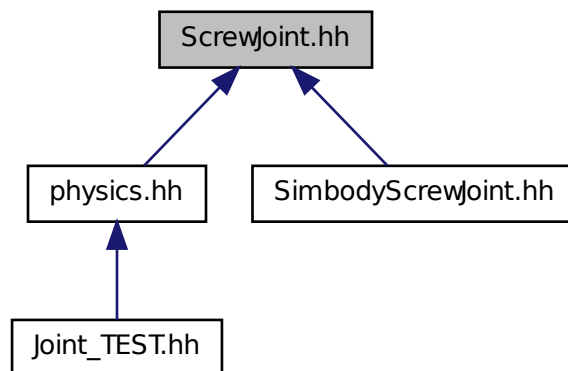
- namespace **Ogre**
- namespace **SkyX**

11.135 ScrewJoint.hh File Reference

```
#include "gazebo/physics/Joint.hh"
#include "gazebo/common/Console.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

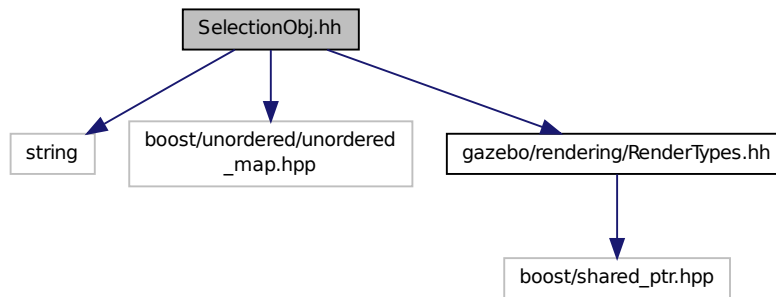
- namespace **gazebo**
Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

11.136 SelectionObj.hh File Reference

```
#include <string>
#include <boost/unordered/unordered_map.hpp>
#include "gazebo/rendering/RenderTypes.hh"
Include dependency graph for SelectionObj.hh:
```



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.

11.137 Sensor.hh File Reference

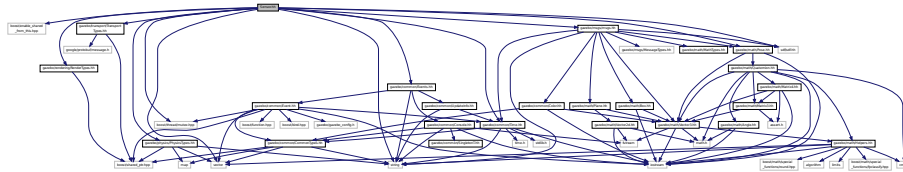
```
#include <boost/enable_shared_from_this.hpp>
```

```

#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/msgs/msgs.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/transport/TransportTypes.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

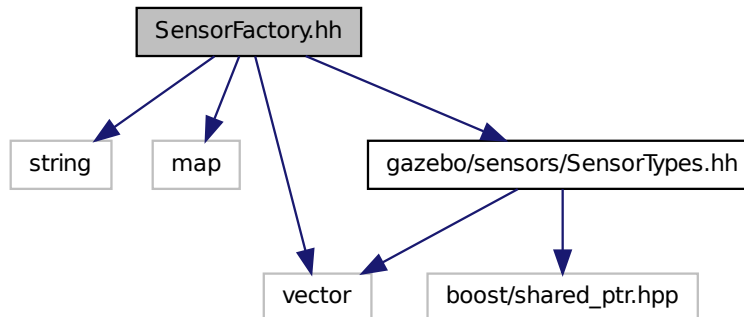
- enum **gazebo::sensors::SensorCategory** { **gazebo::sensors::IMAGE** = 0, **gazebo::sensors::RAY** = 1, **gazebo::sensors::OTHER** = 2, **gazebo::sensors::CATEGORY_COUNT** = 3 }

SensorClass is used to categorize sensors.

11.138 SensorFactory.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "gazebo/sensors/SensorTypes.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.

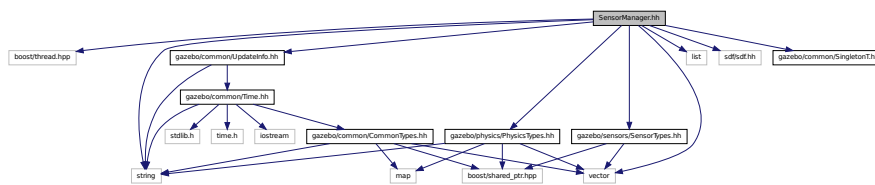
Typedefs

- typedef Sensor ***(** gazebo::sensors::SensorFactoryFn*)()**

11.139 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 dependency graph for SensorManager.hh:



Classes

- class **gazebo::sensors::SensorManager**

Class to manage and update all sensors.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

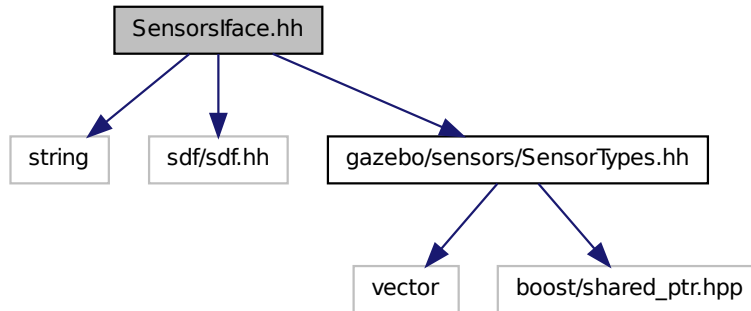
- namespace **gazebo::sensors**

Sensors namespace.

11.140 SensorsIface.hh File Reference

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/sensors/SensorTypes.hh"
```


Include dependency graph for Sensorsface.hh:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

Functions

- std::string **gazebo::sensors::create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName) **GAZEBO_DEPRECATED**(1.10)
Deprecated.
- 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.
- bool **gazebo::sensors::fini** ()
shutdown the sensor generation loop.
- SensorPtr **gazebo::sensors::get_sensor** (const std::string &_name)
Get a sensor using by name.
- bool **gazebo::sensors::init** ()
initialize the sensor generation loop.
- bool **gazebo::sensors::load** ()
Load the sensor library.
- void **gazebo::sensors::remove_sensor** (const std::string &_sensorName)
Remove a sensor by name.
- bool **gazebo::sensors::remove_sensors** ()
Remove all sensors.
- void **gazebo::sensors::run_once** (bool _force=false)
Run the sensor generation one step.
- void **gazebo::sensors::run_threads** ()

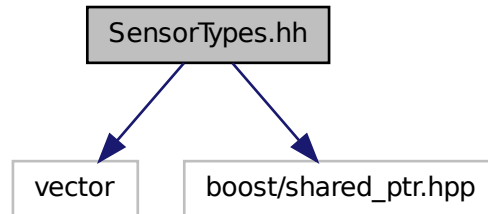
Run sensors in a threads. This is a non-blocking call.

- void **gazebo::sensors::stop** ()
Stop the sensor generation loop.

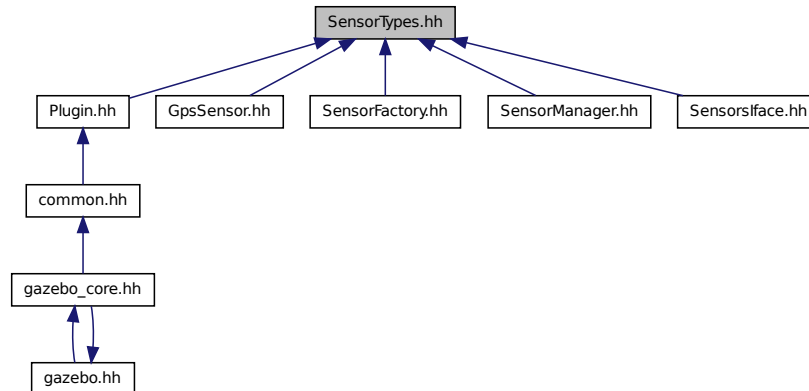
11.141 SensorTypes.hh File Reference

Forward declarations and typedefs for sensors.

```
#include <vector>
#include <boost/shared_ptr.hpp>
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
< GpsSensor > **gazebo::sensors::GpsSensorPtr**
- typedef std::vector
< GpuRaySensorPtr > **gazebo::sensors::GpuRaySensor_V**
- typedef boost::shared_ptr
< GpuRaySensor > **gazebo::sensors::GpuRaySensorPtr**
- typedef std::vector< ImuSensorPtr > **gazebo::sensors::ImuSensor_V**
- typedef boost::shared_ptr
< ImuSensor > **gazebo::sensors::ImuSensorPtr**
- 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**

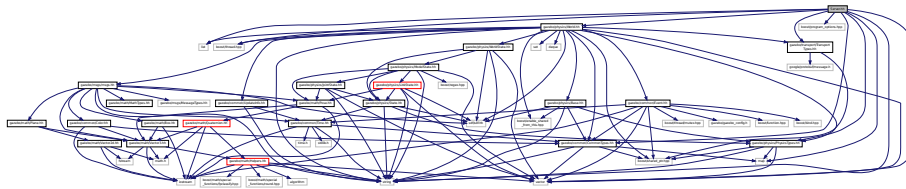
11.141.1 Detailed Description

Forward declarations and typedefs for sensors.

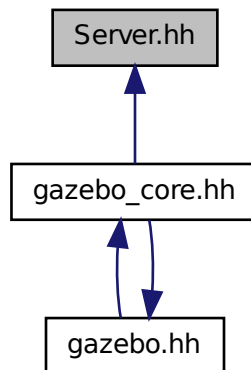
11.142 Server.hh File Reference

```
#include <string>
#include <vector>
#include <list>
#include <map>
#include <boost/program_options.hpp>
#include <boost/thread.hpp>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/World.hh"
```

Include dependency graph for Server.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class `gazebo::Server`

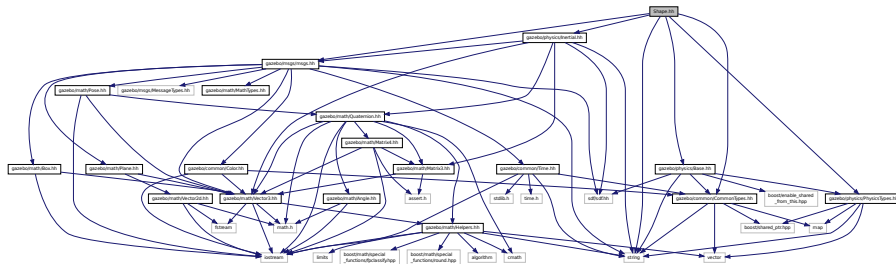
Namespaces

- namespace `boost`
- namespace `gazebo`

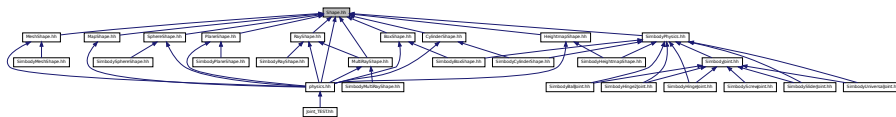
Forward declarations for the common classes.

11.143 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 dependency graph for Shape.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `gazebo::physics::Shape`
Base (p. 153) class for all shapes.

Namespaces

- namespace `gazebo`
Forward declarations for the common classes.

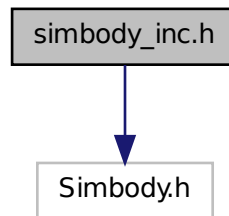
- namespace **gazebo::physics**

namespace for physics

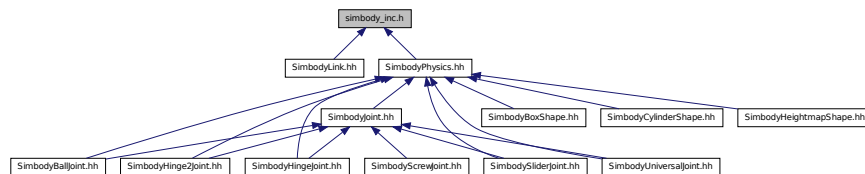
11.144 simbody_inc.h File Reference

```
#include <Simbody.h>
```

Include dependency graph for simbody_inc.h:



This graph shows which files directly or indirectly include this file:



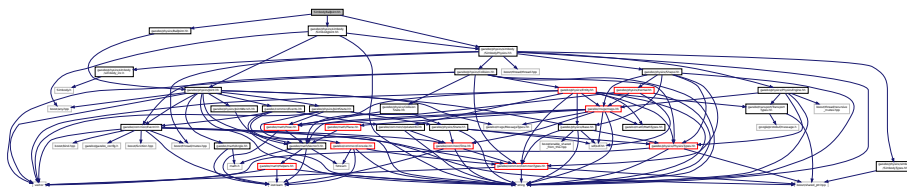
11.145 SimbodyBallJoint.hh File Reference

```
#include "gazebo/physics/BallJoint.hh"
```

```
#include "gazebo/physics/simbody/SimbodyJoint.hh"
```

```
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
```

Include dependency graph for SimbodyBallJoint.hh:



Classes

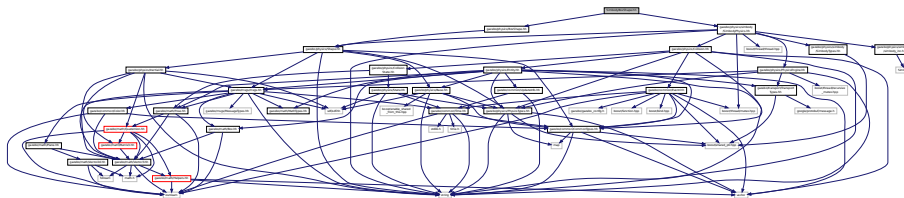
- class **gazebo::physics::SimbodyBallJoint**
SimbodyBallJoint (p. 778) class models a ball joint in Simbody.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.146 SimbodyBoxShape.hh File Reference

```
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/physics/BoxShape.hh"
Include dependency graph for SimbodyBoxShape.hh:
```



Classes

- class **gazebo::physics::SimbodyBoxShape**
Simbody box collision.

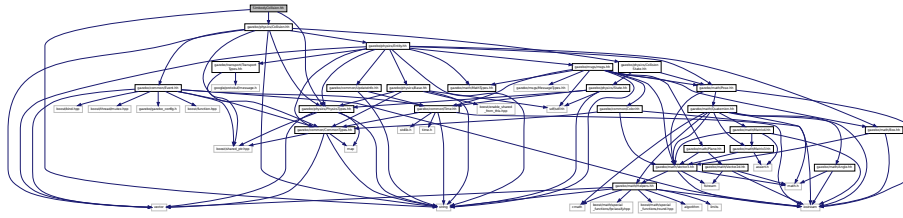
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.147 SimbodyCollision.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Collision.hh"
```

Include dependency graph for SimbodyCollision.hh:



Classes

- class **gazebo::physics::SimbodyCollision**
Simbody collisions.

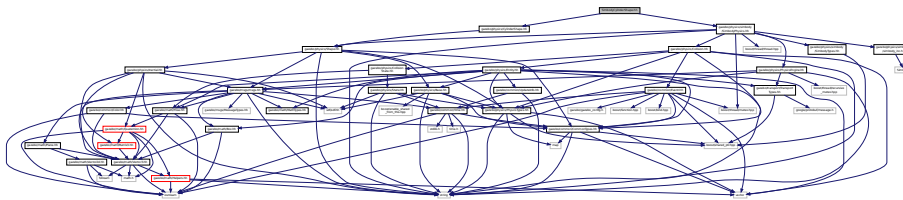
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics
- namespace **SimTK**

11.148 SimbodyCylinderShape.hh File Reference

```
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/physics/CylinderShape.hh"
```

Include dependency graph for SimbodyCylinderShape.hh:



Classes

- class **gazebo::physics::SimbodyCylinderShape**
Cylinder collision.

Namespaces

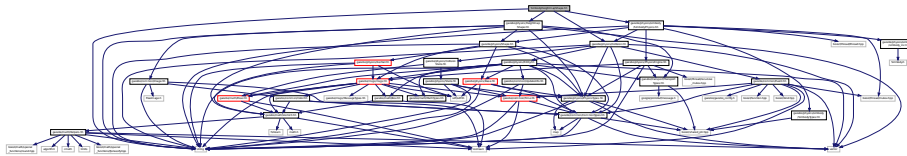
- namespace **gazebo**
Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

11.149 SimbodyHeightmapShape.hh File Reference

```
#include <string>
#include "gazebo/physics/HeightmapShape.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/physics/Collision.hh"
Include dependency graph for SimbodyHeightmapShape.hh:
```



Classes

- class **gazebo::physics::SimbodyHeightmapShape**

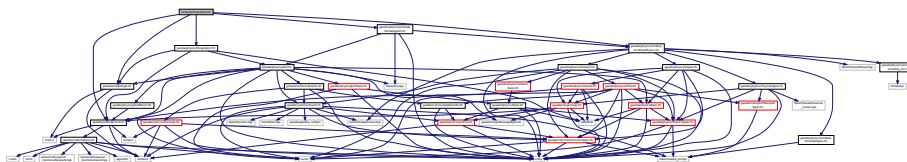
Height map collision.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.150 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 dependency graph for SimbodyHinge2Joint.hh:
```



Classes

- class **gazebo::physics::SimbodyHinge2Joint**

A two axis hinge joint.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

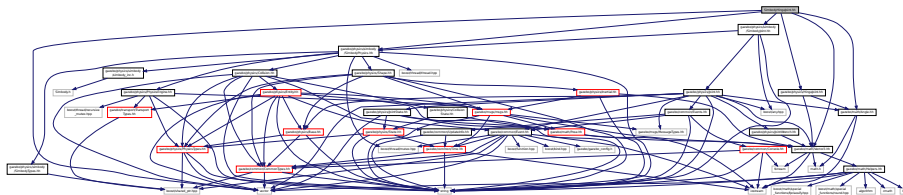
- namespace **gazebo::physics**

namespace for physics

11.151 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 dependency graph for SimbodyHingeJoint.hh:



Classes

- class **gazebo::physics::SimbodyHingeJoint**

A single axis hinge joint.

Namespaces

- namespace **gazebo**

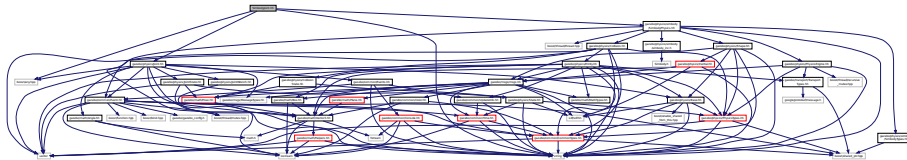
Forward declarations for the common classes.

- namespace **gazebo::physics**

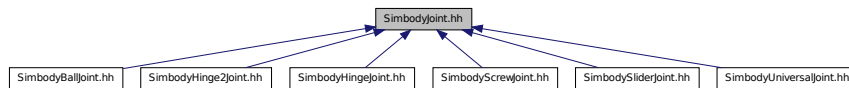
namespace for physics

11.152 SimbodyJoint.hh File Reference

```
#include <boost/any.hpp>
#include <string>
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
#include "gazebo/physics/Joint.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. 153) class for all joints.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

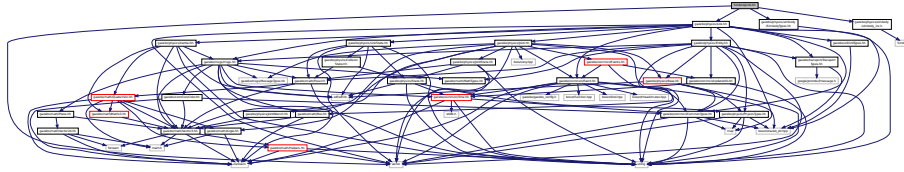
- namespace **gazebo::physics**

namespace for physics

11.153 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 dependency graph for SimbodyLink.hh:



Classes

- class **gazebo::physics::SimbodyLink**
Simbody Link (p. 455) class.

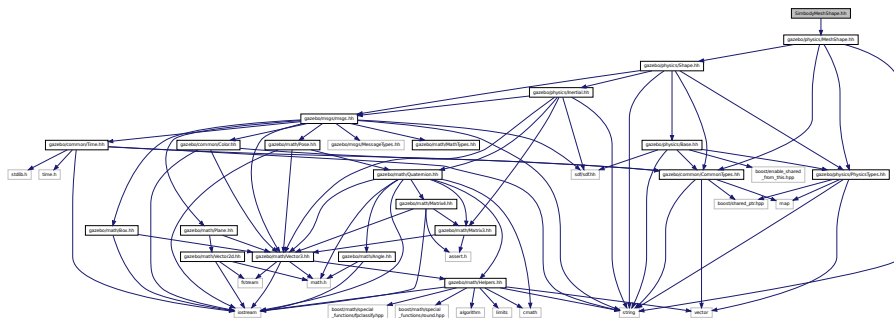
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.154 SimbodyMeshShape.hh File Reference

```
#include "gazebo/physics/MeshShape.hh"
```

Include dependency graph for SimbodyMeshShape.hh:



Classes

- class **gazebo::physics::SimbodyMeshShape**
Triangle mesh collision.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

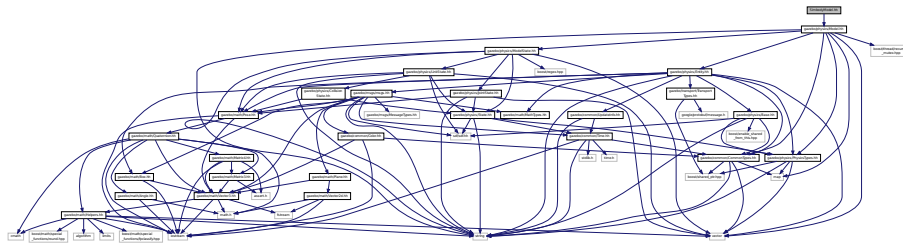
- namespace **gazebo::physics**

namespace for physics

11.155 SimbodyModel.hh File Reference

```
#include "gazebo/physics/Model.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.

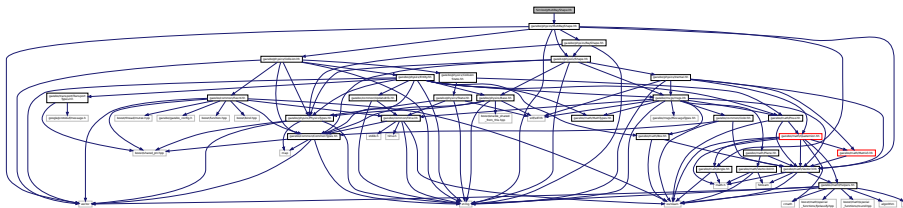
- namespace **gazebo::physics**

namespace for physics

11.156 SimbodyMultiRayShape.hh File Reference

```
#include "gazebo/physics/MultiRayShape.hh"
```

Include dependency graph for SimbodyMultiRayShape.hh:



Classes

- class **gazebo::physics::SimbodyMultiRayShape**
*Simbody specific version of **MultiRayShape** (p. 578).*

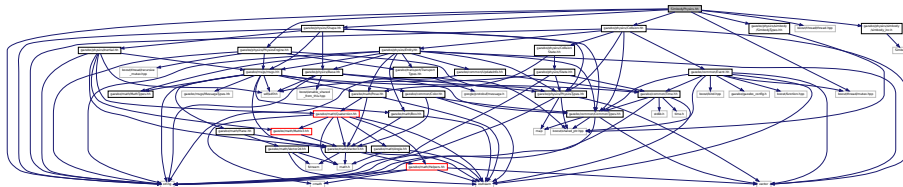
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

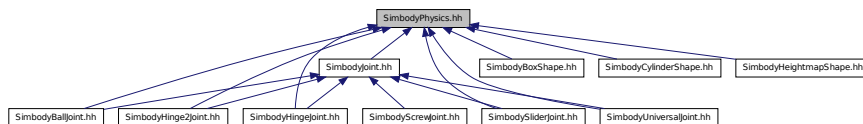
11.157 SimbodyPhysics.hh File Reference

```
#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 dependency graph for SimbodyPhysics.hh:



This graph shows which files directly or indirectly include this file:



Classes

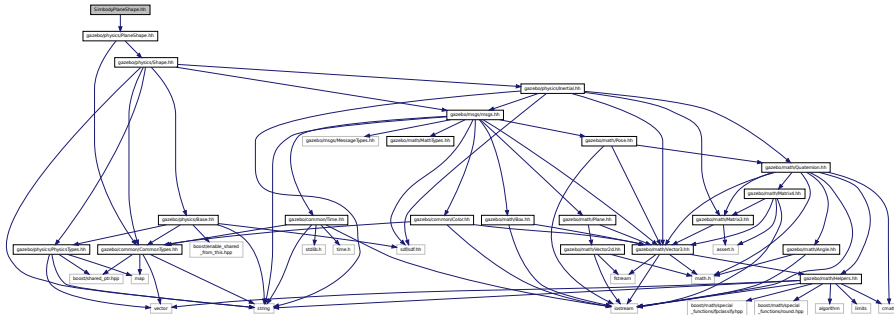
- class **gazebo::physics::SimbodyPhysics**
Simbody physics engine.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.158 SimbodyPlaneShape.hh File Reference

```
#include "gazebo/physics/PlaneShape.hh"
Include dependency graph for SimbodyPlaneShape.hh:
```



Classes

- class **gazebo::physics::SimbodyPlaneShape**
Simbody collision for an infinite plane.

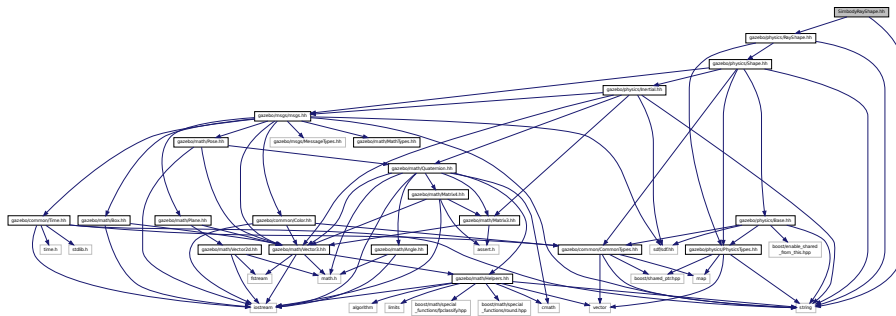
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.159 SimbodyRayShape.hh File Reference

```
#include <string>
#include "gazebo/physics/RayShape.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**

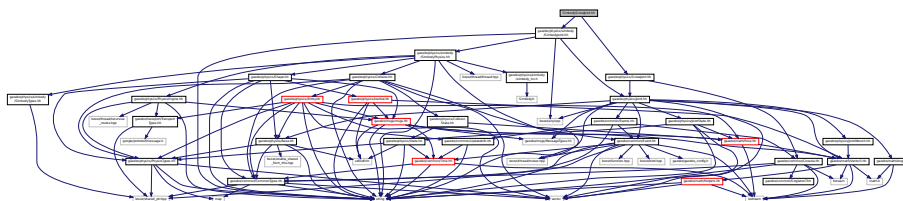
namespace for physics

11.160 SimbodyScrewJoint.hh File Reference

```
#include "gazebo/physics/simbody/SimbodyJoint.hh"
```

```
#include "gazebo/physics/ScrewJoint.hh"
```

Include dependency graph for SimbodyScrewJoint.hh:



Classes

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

A screw joint.

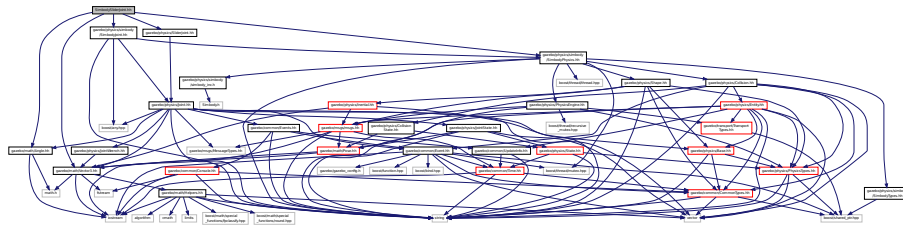
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.161 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 dependency graph for SimbodySliderJoint.hh:



Classes

- class **gazebo::physics::SimbodySliderJoint**
A slider joint.

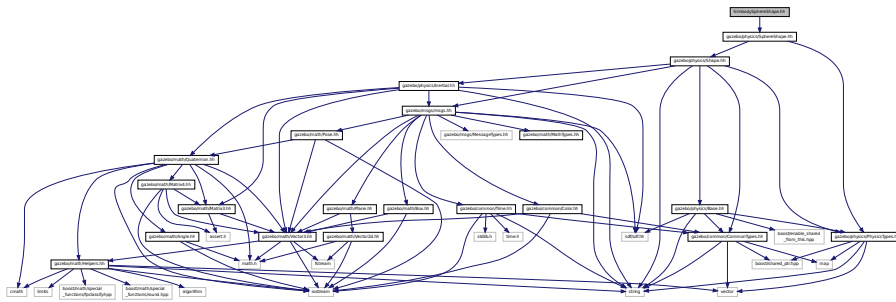
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.162 SimbodySphereShape.hh File Reference

```
#include "gazebo/physics/SphereShape.hh"
```

Include dependency graph for SimbodySphereShape.hh:



Classes

- class **gazebo::physics::SimbodySphereShape**
Simbody sphere collision.

Namespaces

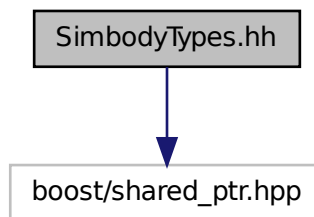
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.163 SimbodyTypes.hh File Reference

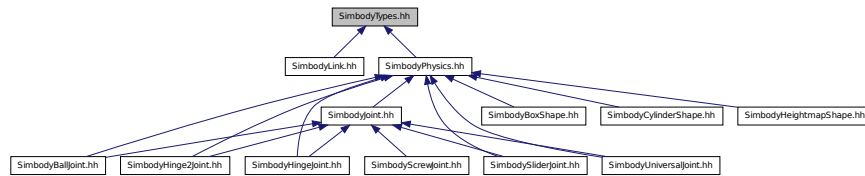
Simbody wrapper forward declarations and typedefs.

```
#include <boost/shared_ptr.hpp>
```

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

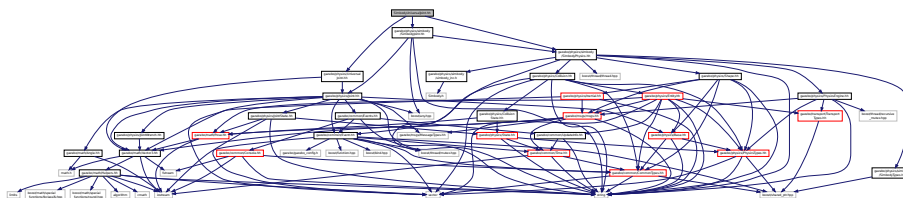
- typedef boost::shared_ptr
< SimbodyCollision > **gazebo::physics::SimbodyCollisionPtr**
- typedef boost::shared_ptr
< SimbodyLink > **gazebo::physics::SimbodyLinkPtr**
- typedef boost::shared_ptr
< SimbodyModel > **gazebo::physics::SimbodyModelPtr**
- typedef boost::shared_ptr
< SimbodyPhysics > **gazebo::physics::SimbodyPhysicsPtr**
- typedef boost::shared_ptr
< SimbodyRayShape > **gazebo::physics::SimbodyRayShapePtr**

11.163.1 Detailed Description

Simbody wrapper forward declarations and typedefs.

11.164 SimbodyUniversalJoint.hh File Reference

```
#include "gazebo/physics/UniversalJoint.hh"
#include "gazebo/physics/simbody/SimbodyJoint.hh"
#include "gazebo/physics/simbody/SimbodyPhysics.hh"
Include dependency graph for SimbodyUniversalJoint.hh:
```



Classes

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

A simbody universal joint class.

Namespaces

- namespace **gazebo**

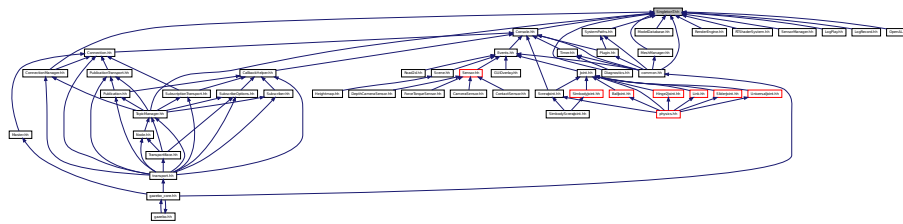
Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

11.165 SingletonT.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

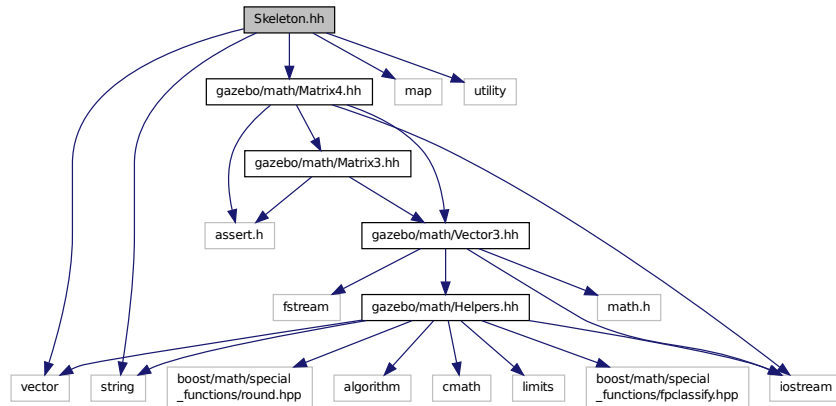
- class **SingletonT< T >**

Singleton template class.

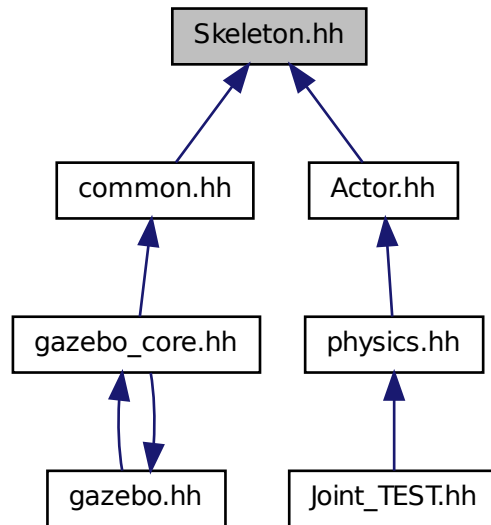
11.166 Skeleton.hh File Reference

```
#include <vector>
#include <string>
#include <map>
#include <utility>
#include "gazebo/math/Matrix4.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. 598) *Skeleton.hh* (p. 1268) *common/common.hh*
- class **gazebo::common::Skeleton**
A skeleton.

- class **gazebo::common::SkeletonNode**

A skeleton node.

Namespaces

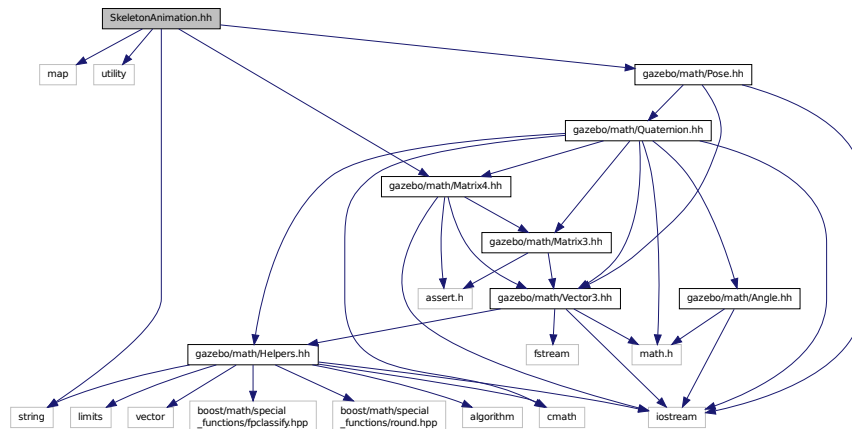
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

Typedefs

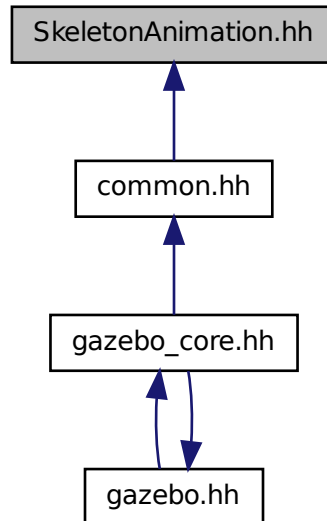
- typedef `std::map< unsigned int, SkeletonNode * >` **gazebo::common::NodeMap**
- typedef `std::map< unsigned int, SkeletonNode * >::iterator` **gazebo::common::NodeMapIter**
- typedef `std::map< double, std::vector< NodeTransform > >` **gazebo::common::RawNodeAnim**
- typedef `std::vector< std::vector< std::pair< std::string, double > > >` **gazebo::common::RawNodeWeights**
- typedef `std::map< std::string, RawNodeAnim >` **gazebo::common::RawSkeletonAnim**

11.167 SkeletonAnimation.hh File Reference

```
#include <map>
#include <utility>
#include <string>
#include "gazebo/math/Matrix4.hh"
#include "gazebo/math/Pose.hh"
Include dependency graph for SkeletonAnimation.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::NodeAnimation**
Node animation.
- class **gazebo::common::SkeletonAnimation**
***Skeleton** (p. 866) animation.*

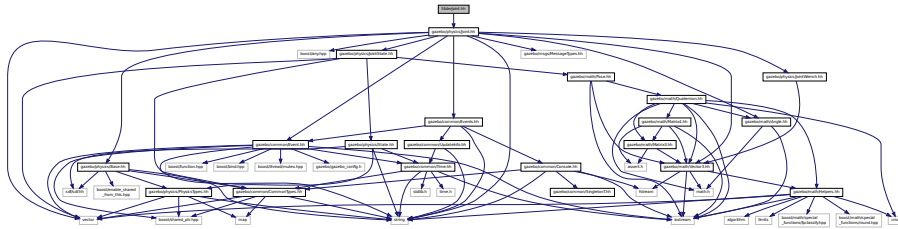
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

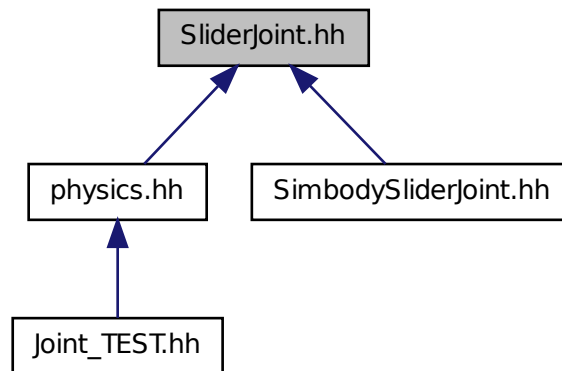
11.168 SliderJoint.hh File Reference

```
#include "gazebo/physics/Joint.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

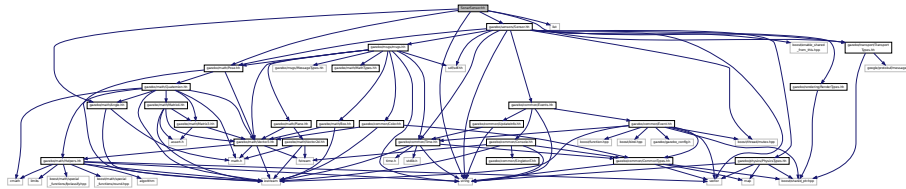
11.169 SonarSensor.hh File Reference

```
#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 dependency graph for SonarSensor.hh:



Classes

- class **gazebo::sensors::SonarSensor**
Sensor (p. 751) with sonar cone.

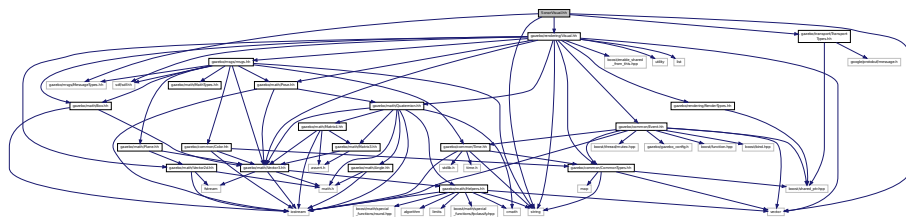
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.170 SonarVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
#include "gazebo msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for SonarVisual.hh:



Classes

- class **gazebo::rendering::SonarVisual**
Visualization for sonar data.

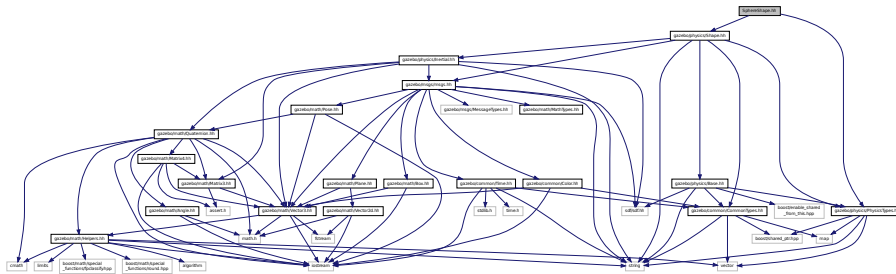
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

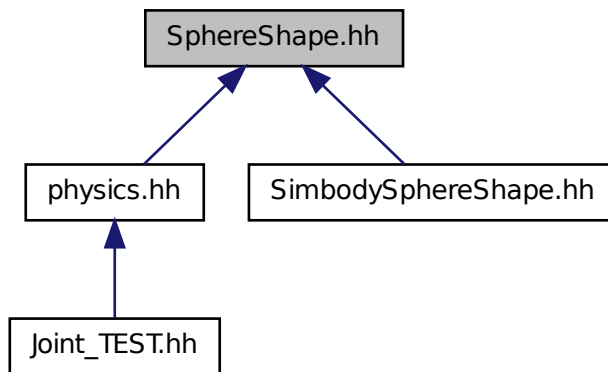
11.171 SphereShape.hh File Reference

```
#include "gazebo/physics/Shape.hh"
#include "gazebo/physics/PhysicsTypes.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.

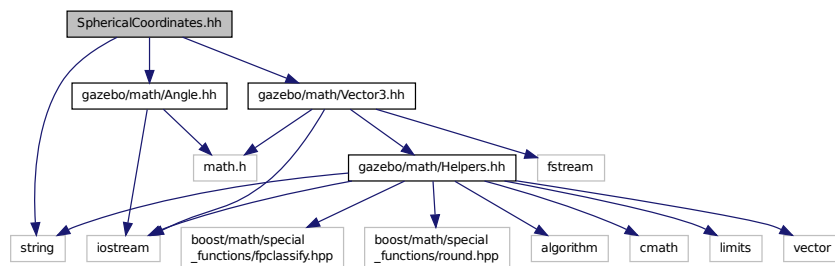
- namespace **gazebo::physics**

namespace for physics

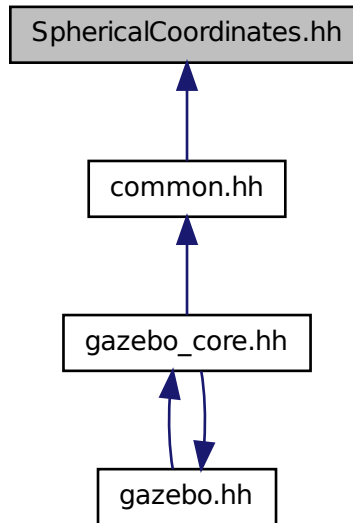
11.172 SphericalCoordinates.hh File Reference

```
#include <string>
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.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.

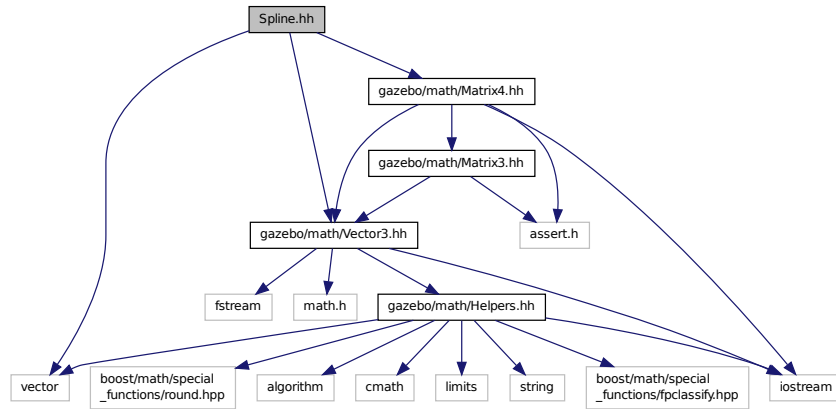
- namespace **gazebo::common**

Common namespace.

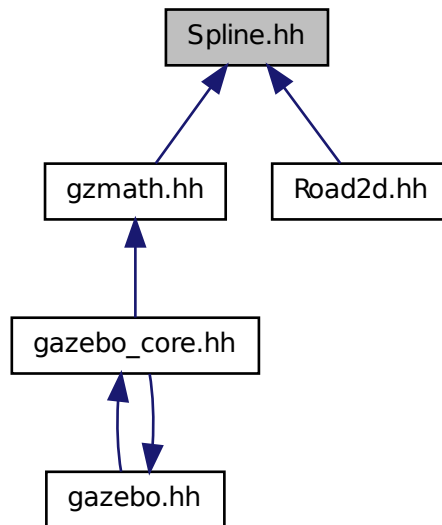
11.173 Spline.hh File Reference

```
#include <vector>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Matrix4.hh"
```

Include dependency graph for Spline.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Spline**

Splines.

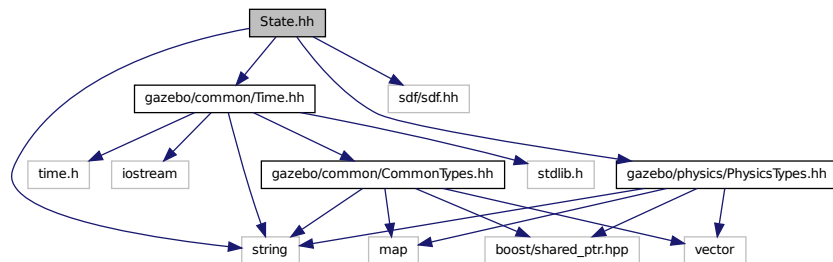
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

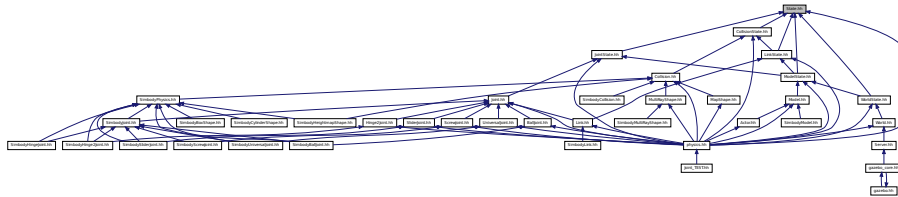
11.174 State.hh File Reference

```
#include <string>
#include <sdf/sdf.hh>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/common/Time.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. 910) of an entity.

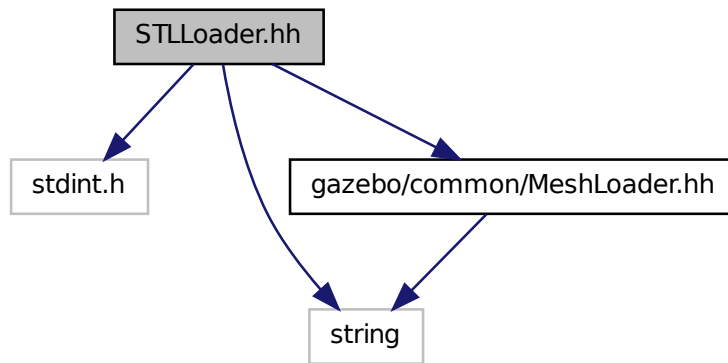
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

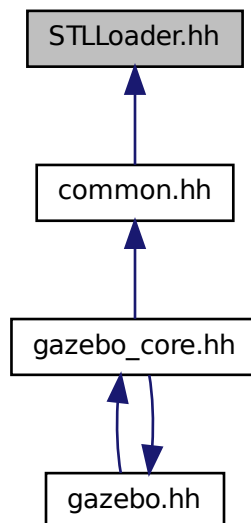
11.175 STLLoader.hh File Reference

```
#include <stdint.h>
#include <string>
#include "gazebo/common/MeshLoader.hh"
```

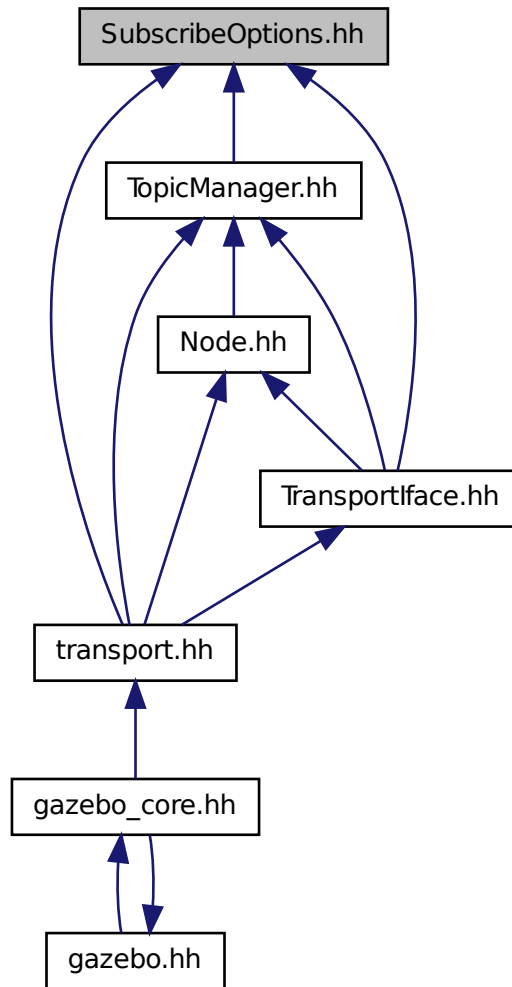
Include dependency graph for STLLoader.hh:



This graph shows which files directly or indirectly include this file:



This graph shows which files directly or indirectly include this file:



Classes

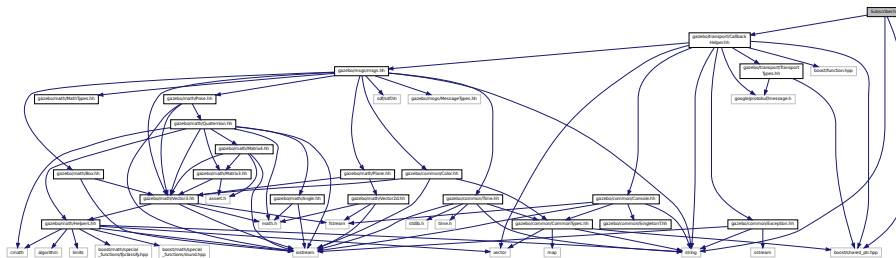
- class **`gazebo::transport::SubscribeOptions`**
Options for a subscription.

Namespaces

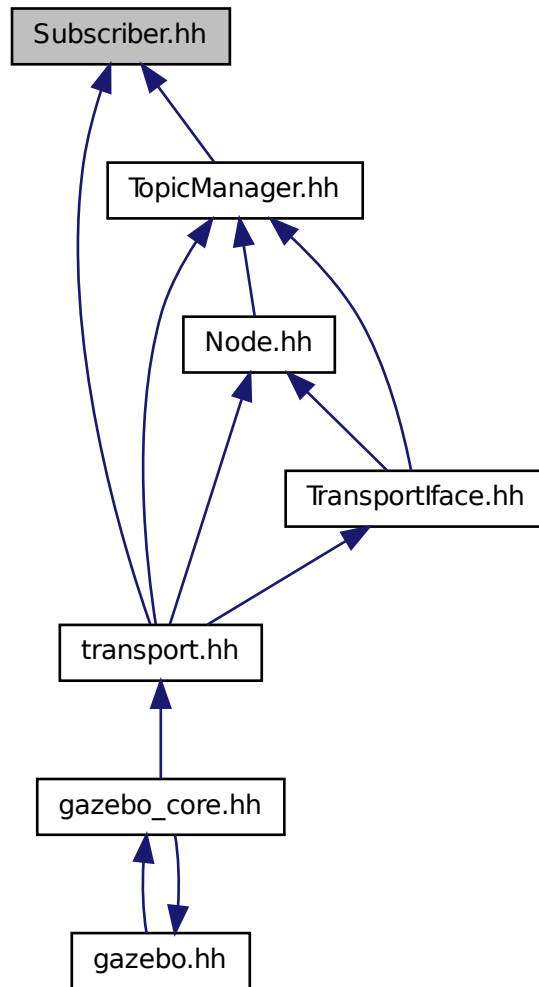
- namespace **`gazebo`**
Forward declarations for the common classes.
- namespace **`gazebo::transport`**

11.177 Subscriber.hh File Reference

```
#include <string>
#include <boost/shared_ptr.hpp>
#include "gazebo/transport/CallbackHelper.hh"
Include dependency graph for Subscriber.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

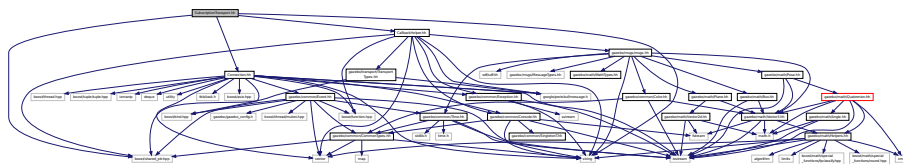
- class **gazebo::transport::Subscriber**
A subscriber to a topic.

Namespaces

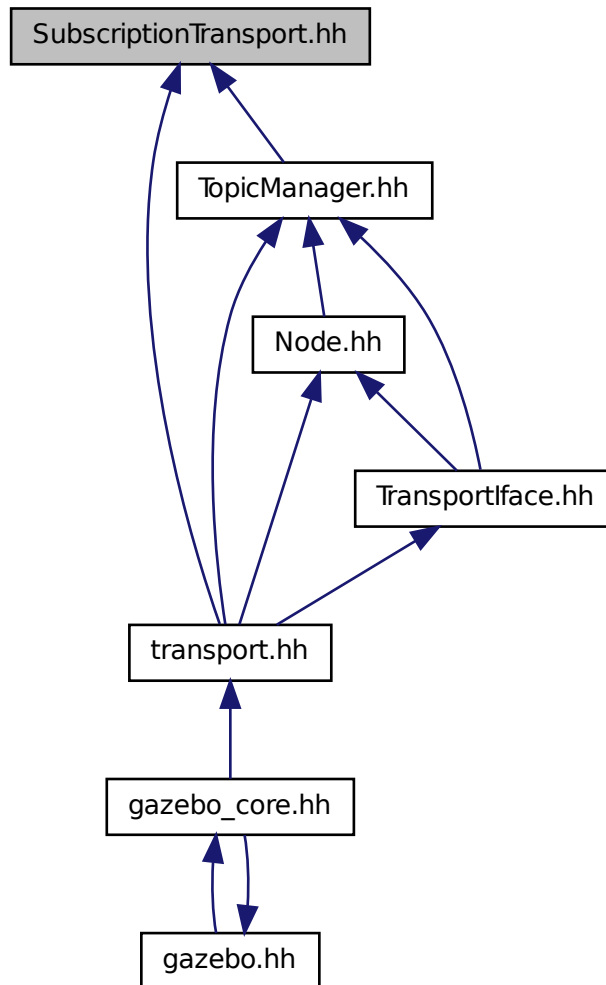
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

11.178 SubscriptionTransport.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <string>
#include "Connection.hh"
#include "CallbackHelper.hh"
Include dependency graph for SubscriptionTransport.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

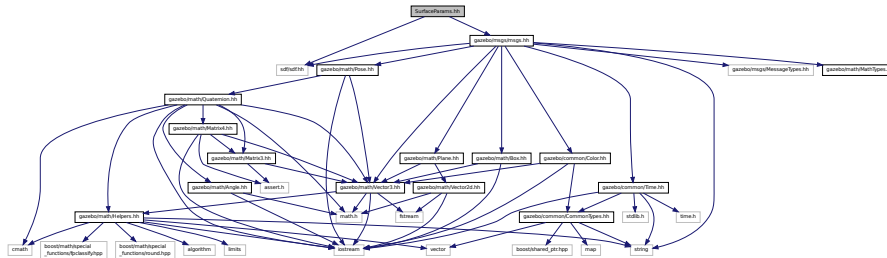
- class **gazebo::transport::SubscriptionTransport**
transport/transport.hh

Namespaces

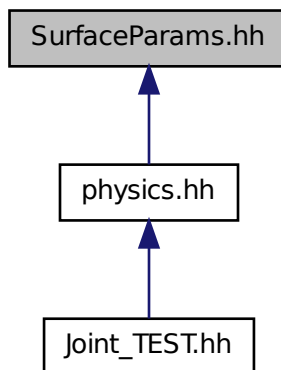
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

11.179 SurfaceParams.hh File Reference

```
#include <sdf/sdf.hh>  
#include "gazebomsgs/msgs.hh"  
Include dependency graph for SurfaceParams.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::SurfaceParams**
SurfaceParams (p. 933) defines various Surface contact parameters.

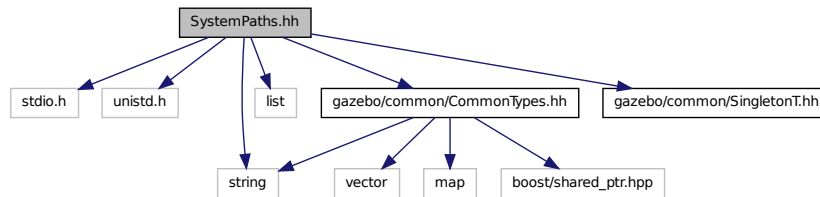
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

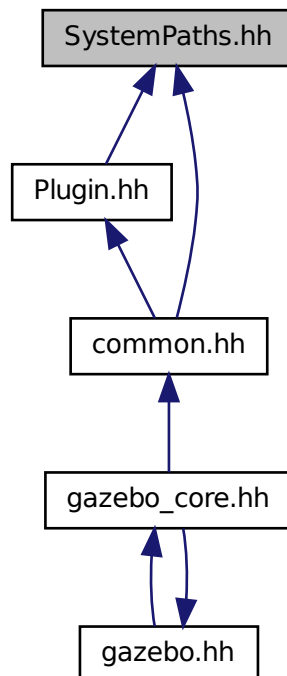
11.180 SystemPaths.hh File Reference

```
#include <stdio.h>
#include <unistd.h>
#include <string>
#include <list>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/SingletonT.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**

11.180.1 Macro Definition Documentation

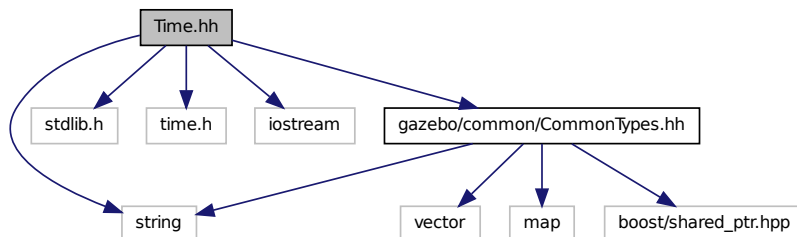
11.180.1.1 #define GetCurrentDir getcwd

11.180.1.2 #define LINUX

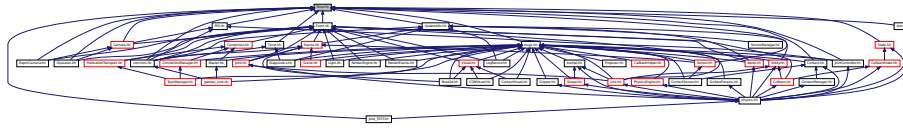
11.181 Time.hh File Reference

```
#include <string>
#include <stdlib.h>
#include <time.h>
#include <iostream>
#include "gazebo/common/CommonTypes.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. 944) class, can be used to hold wall- or sim-time.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

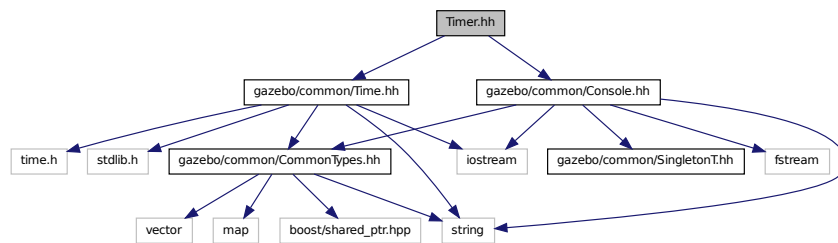
Common namespace.

11.182 Timer.hh File Reference

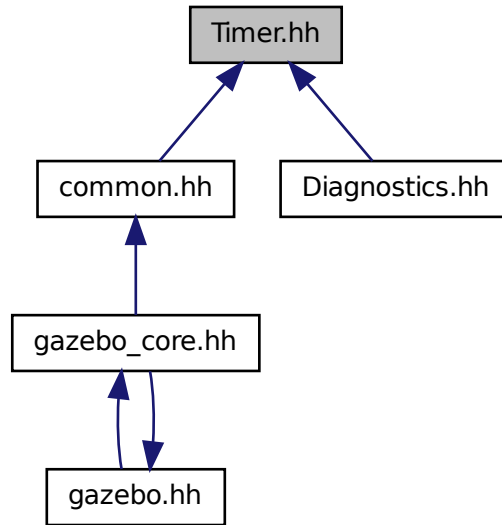
```
#include "gazebo/common/Console.hh"
```

```
#include "gazebo/common/Time.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.

11.183 TopicManager.hh File Reference

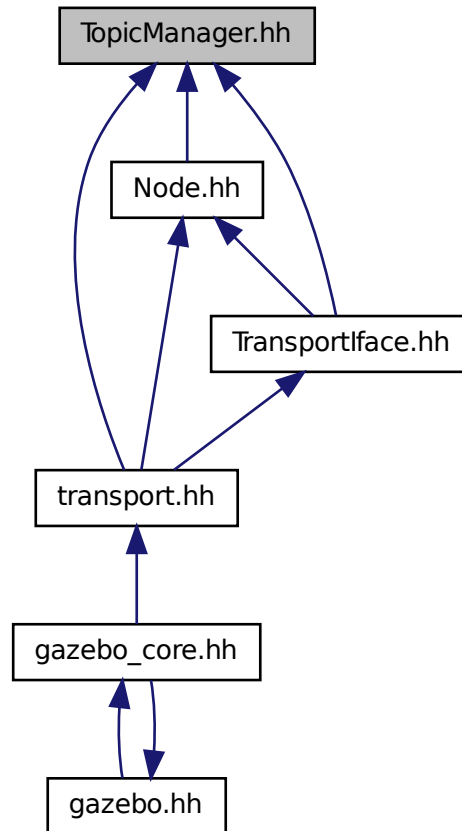
```
#include <boost/bind.hpp>
```

```
#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 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.

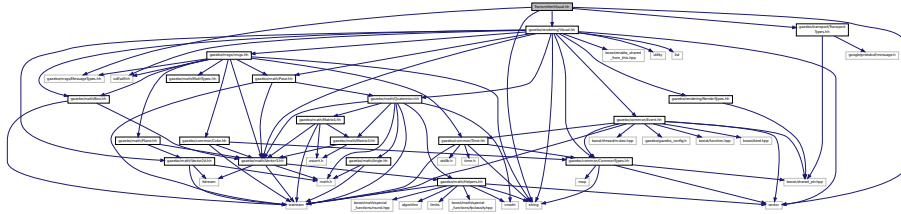
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

11.184 TransmitterVisual.hh File Reference

```
#include <string>
```

```
#include <vector>
#include "gazebo/rendering/Visual.hh"
#include "gazebo_msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
Include dependency graph for TransmitterVisual.hh:
```



Classes

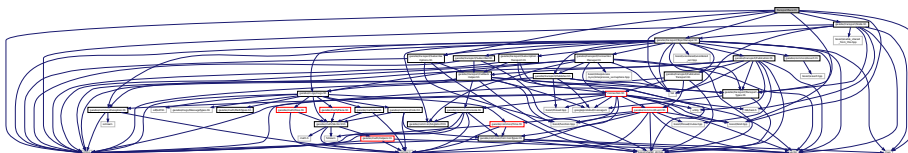
- class **gazebo::rendering::TransmitterVisual**
Visualization for the wireless propagation data.

Namespaces

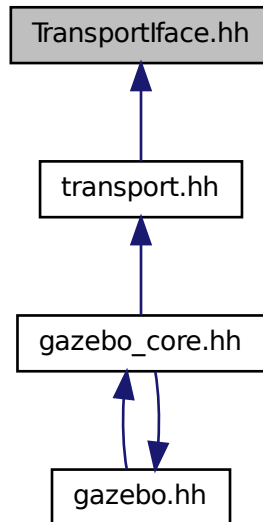
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.185 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

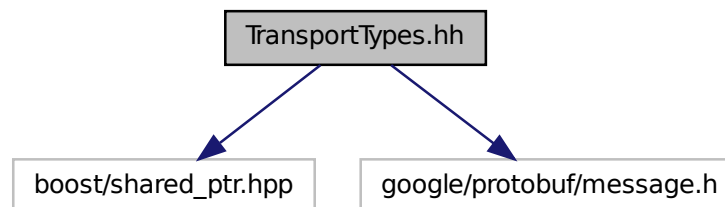
- void **gazebo::transport::clear_buffers** ()
Clear any remaining communication buffers.
- void **gazebo::transport::fini** ()
Cleanup the transport component.
- bool **gazebo::transport::get_master_uri** (std::string &_master_host, unsigned int &_master_port)
Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.
- void **gazebo::transport::get_topic_namespaces** (std::list< std::string > &_namespaces)
Return all the namespace (world names) on the master.
- std::map< std::string, std::list< std::string > > **gazebo::transport::getAdvertisedTopics** ()
Get a list of all the topics and their message types.
- std::list< std::string > **gazebo::transport::getAdvertisedTopics** (const std::string &_msgType)
Get a list of all the unique advertised topic names.
- bool **gazebo::transport::getMinimalComms** ()
Get whether minimal comms has been enabled.

- `std::string gazebo::transport::getTopicMsgType` (const std::string &_topicName)
Get the message typename that is published on the given topic.
- `bool gazebo::transport::init` (const std::string &_master_host="", unsigned int _master_port=0)
Initialize the transport system.
- `bool gazebo::transport::is_stopped` ()
Is the transport system stopped?
- `void gazebo::transport::pause_incoming` (bool _pause)
Pause or unpaue incoming messages.
- `template<typename M > void gazebo::transport::publish` (const std::string &_topic, const google::protobuf::Message &_message)
A convenience function for a one-time publication of a message.
- `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.
- `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.
- `void gazebo::transport::requestNoReply` (NodePtr _node, const std::string &_request, const std::string &_data="")
Send a request and don't wait for a response.
- `void gazebo::transport::run` ()
Run the transport component.
- `void gazebo::transport::setMinimalComms` (bool _enabled)
Set whether minimal comms should be used.
- `void gazebo::transport::stop` ()
Stop the transport component from running.

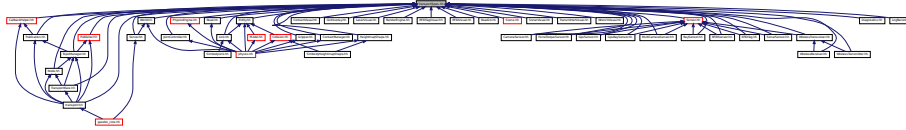
11.186 TransportTypes.hh File Reference

Forward declarations for transport.

```
#include <boost/shared_ptr.hpp>
#include <google/protobuf/message.h>
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

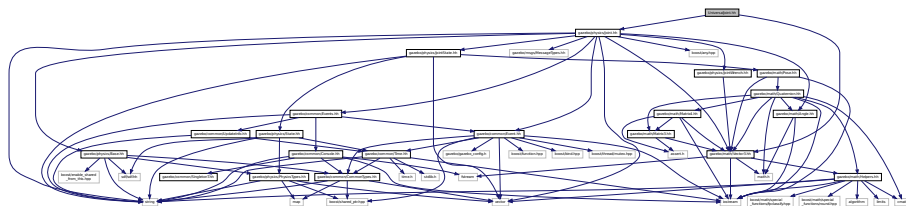
- typedef boost::shared_ptr
< google::protobuf::Message > **gazebo::transport::MessagePtr**
- typedef boost::shared_ptr< Node > **gazebo::transport::NodePtr**
- typedef boost::shared_ptr
< Publication > **gazebo::transport::PublicationPtr**
- typedef boost::shared_ptr
< PublicationTransport > **gazebo::transport::PublicationTransportPtr**
- typedef boost::shared_ptr
< Publisher > **gazebo::transport::PublisherPtr**
- typedef boost::shared_ptr
< Subscriber > **gazebo::transport::SubscriberPtr**
- typedef boost::shared_ptr
< SubscriptionTransport > **gazebo::transport::SubscriptionTransportPtr**

11.186.1 Detailed Description

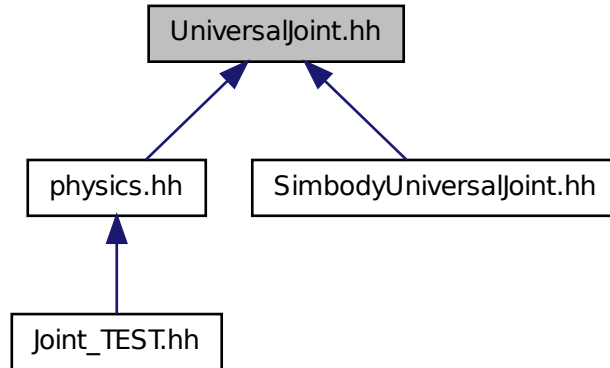
Forward declarations for transport.

11.187 UniversalJoint.hh File Reference

```
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/Joint.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.

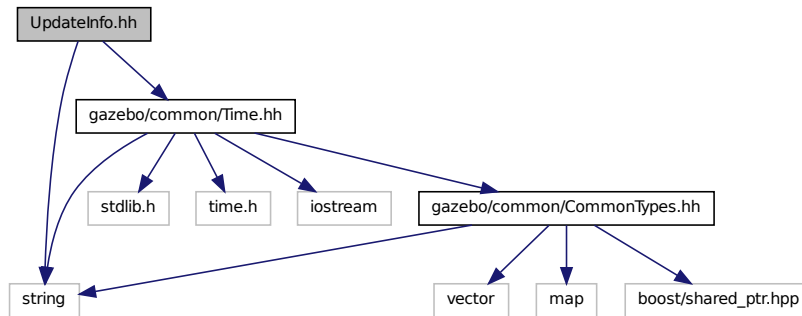
- namespace **gazebo::physics**

namespace for physics

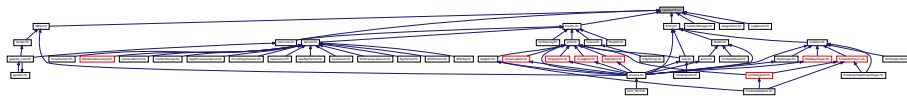
11.188 UpdateInfo.hh File Reference

```
#include <string>  
#include "gazebo/common/Time.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.

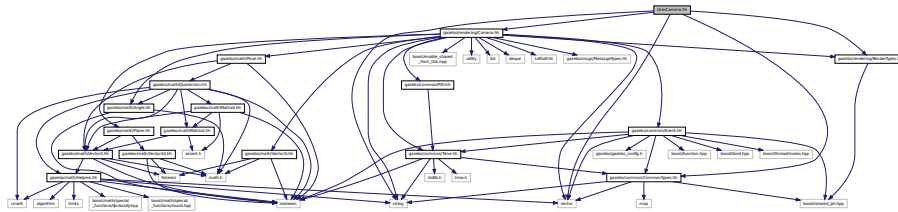
11.189 UserCamera.hh File Reference

```

#include <string>
#include <vector>
#include "gazebo/rendering/Camera.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/common/CommonTypes.hh"

```

Include dependency graph for UserCamera.hh:



Classes

- class **gazebo::rendering::UserCamera**
A camera used for user visualization of a scene.

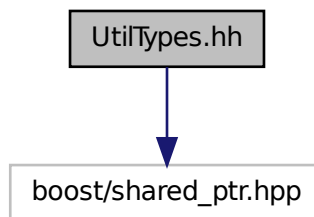
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

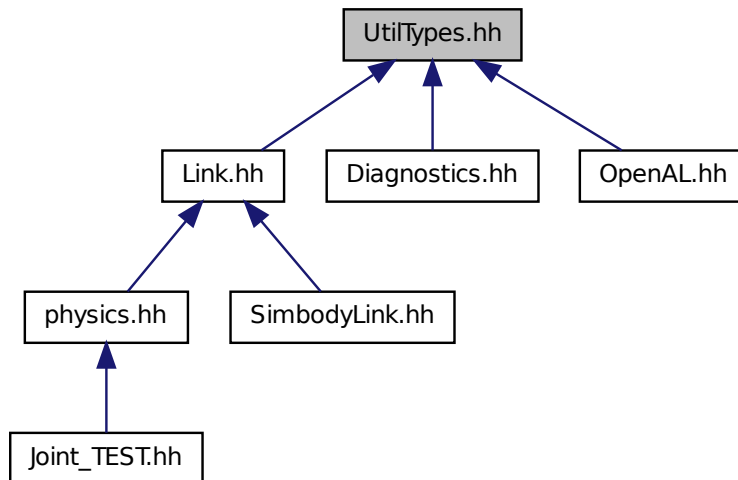
11.190 UtilTypes.hh File Reference

```
#include <boost/shared_ptr.hpp>
```

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**

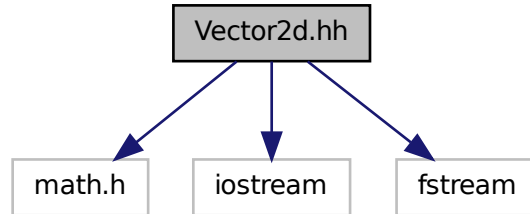
11.191 Vector2d.hh File Reference

```

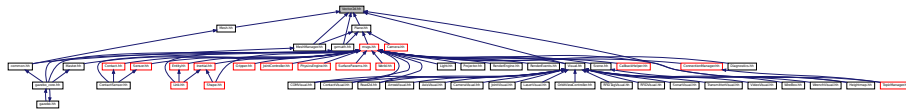
#include <math.h>
#include <iostream>
#include <fstream>

```

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.

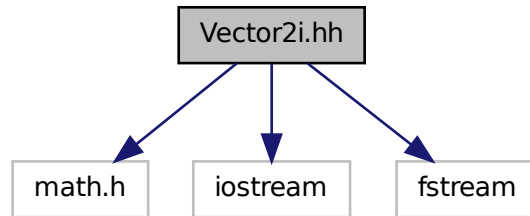
- namespace **gazebo::math**

Math namespace.

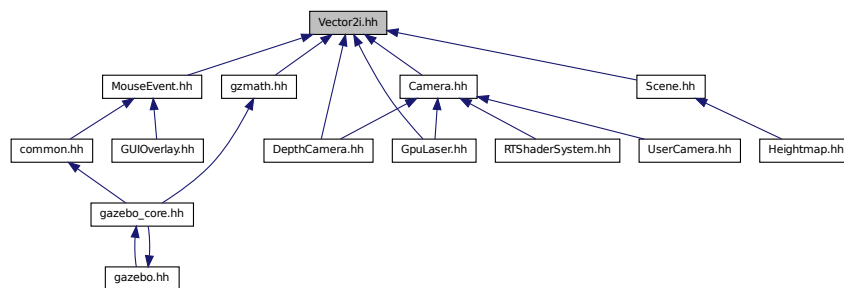
11.192 Vector2i.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
```

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.

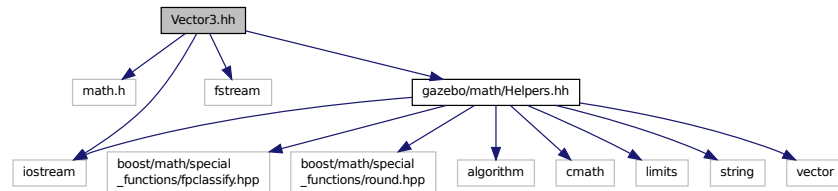
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

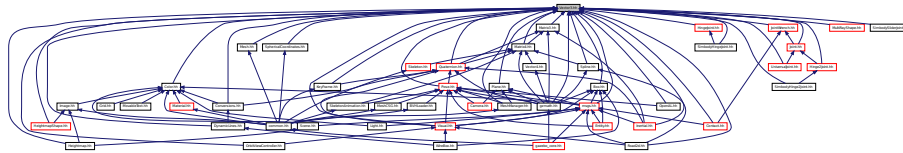
11.193 Vector3.hh File Reference

```
#include <math.h>
```

```
#include <iostream>
#include <fstream>
#include "gazebo/math/Helpers.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. 1004) class represents the generic vector containing 3 elements.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

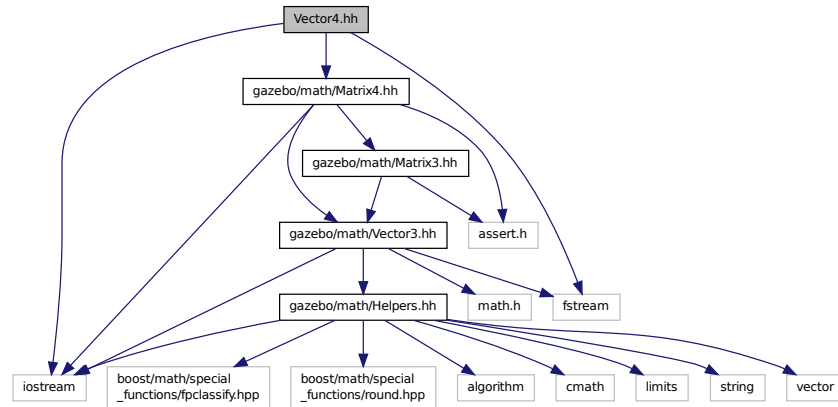
- namespace **gazebo::math**

Math namespace.

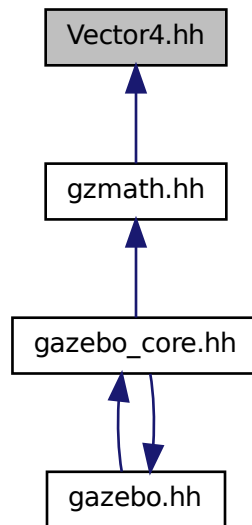
11.194 Vector4.hh File Reference

```
#include <iostream>
#include <fstream>
#include "gazebo/math/Matrix4.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.

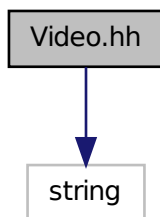
- namespace **gazebo::math**

Math namespace.

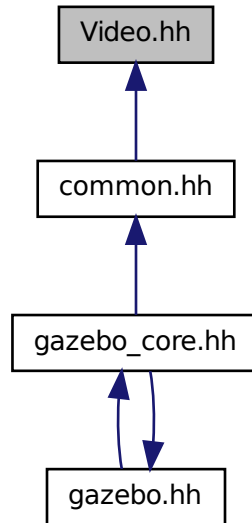
11.195 Video.hh File Reference

```
#include <string>
```

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.

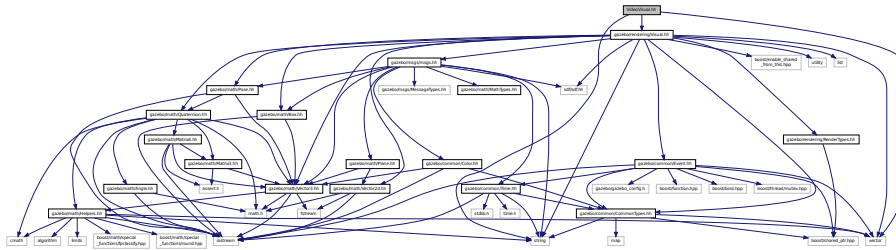
- namespace **gazebo::common**

Common namespace.

11.196 VideoVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for VideoVisual.hh:



Classes

- class **gazebo::rendering::VideoVisual**
A visual element that displays a video as a texture.

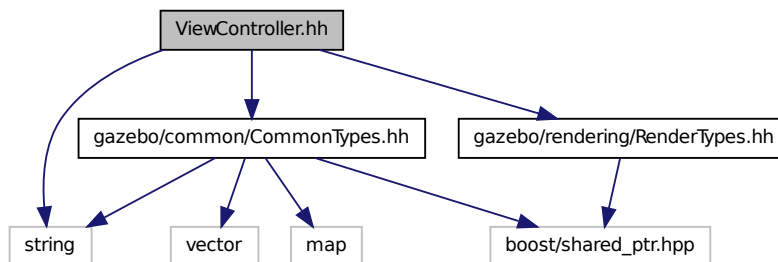
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::rendering**
Rendering namespace.

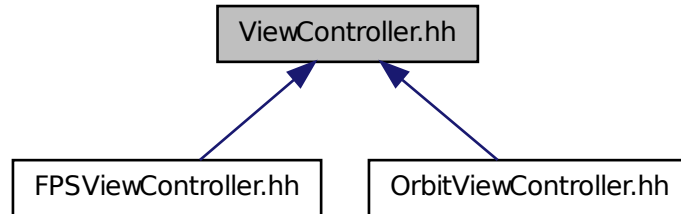
11.197 ViewController.hh File Reference

```
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/rendering/RenderTypes.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.

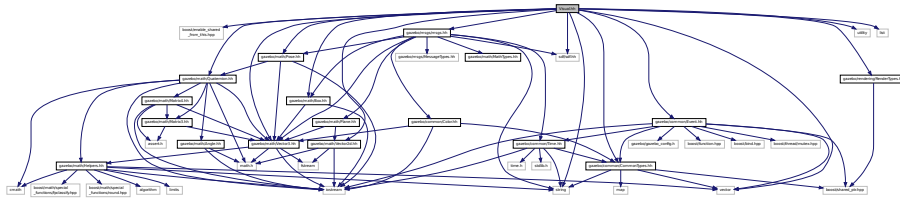
- namespace **gazebo::rendering**

Rendering namespace.

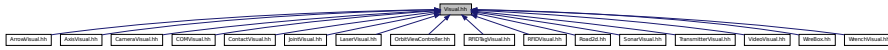
11.198 Visual.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#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 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**

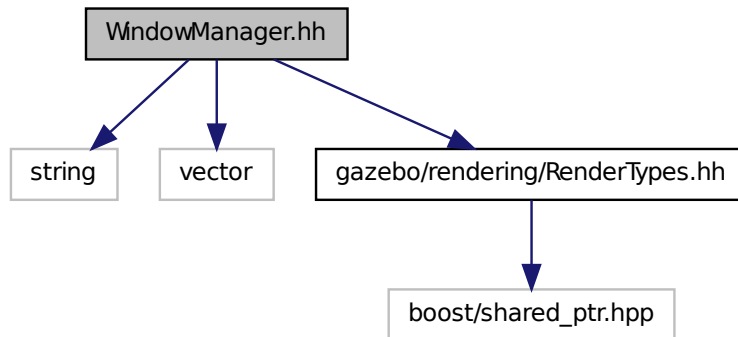
Rendering namespace.

- namespace **Ogre**

11.199 WindowManager.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/RenderTypes.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**

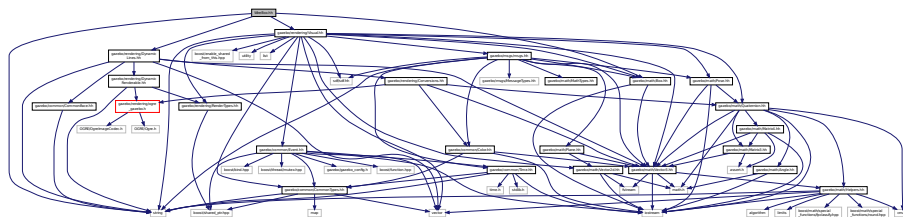
11.200 WireBox.hh File Reference

```

#include <string>
#include "gazebo/math/Box.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/rendering/DynamicLines.hh"

```

Include dependency graph for WireBox.hh:



Classes

- class **gazebo::rendering::WireBox**
Draws a wireframe box.

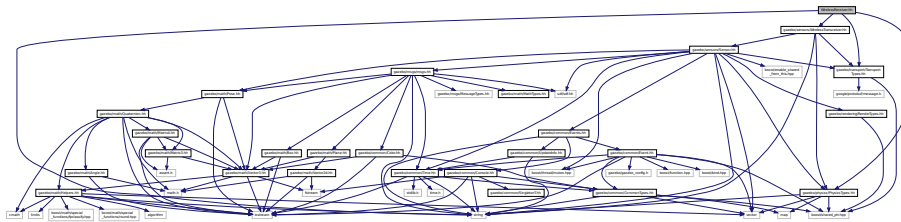
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.201 WirelessReceiver.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/WirelessTransceiver.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for WirelessReceiver.hh:



Classes

- class **gazebo::sensors::WirelessReceiver**
Sensor (p. 751) class for receiving wireless signals.

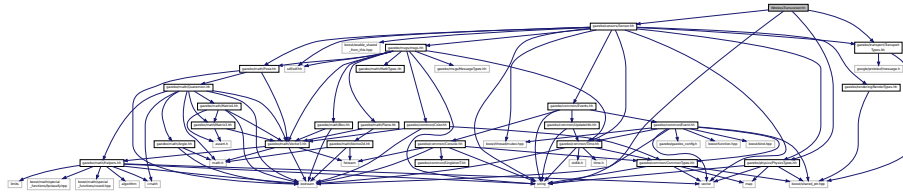
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

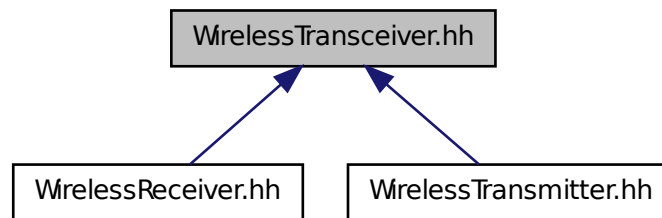
11.202 WirelessTransceiver.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/transport/TransportTypes.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. 751) class for receiving wireless signals.

Namespaces

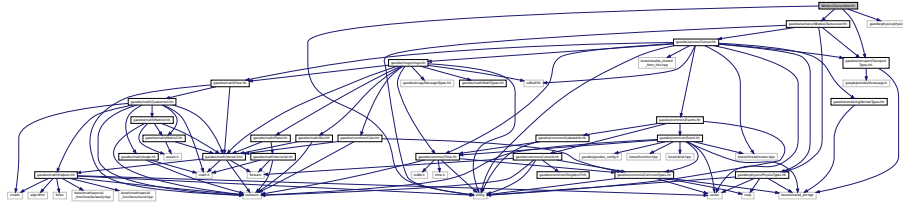
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.203 WirelessTransmitter.hh File Reference

```

#include <string>
#include "gazebo/physics/physics.hh"
#include "gazebo/sensors/WirelessTransceiver.hh"
#include "gazebo/transport/TransportTypes.hh"
  
```


Include dependency graph for WirelessTransmitter.hh:



Classes

- class **gazebo::sensors::WirelessTransmitter**
Transmitter to send wireless signals.

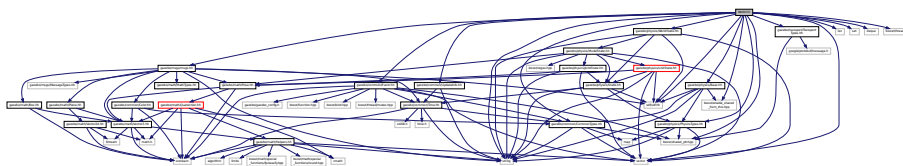
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

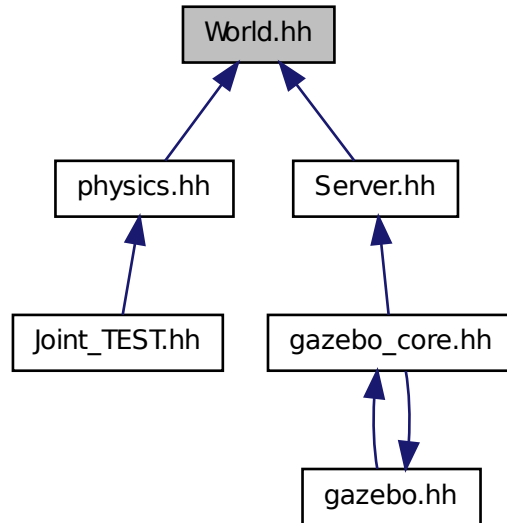
11.204 World.hh File Reference

```
#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 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

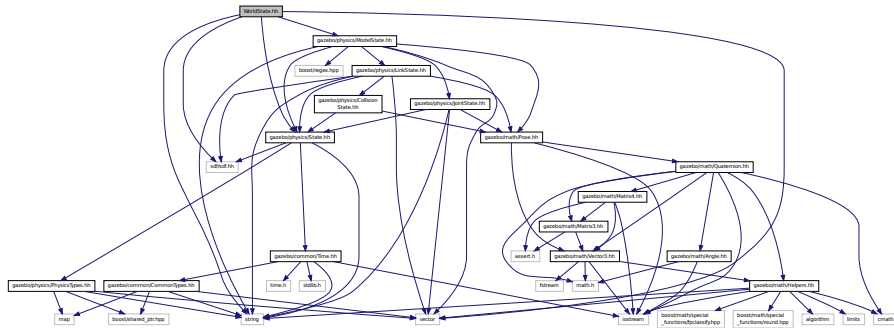
11.205 WorldState.hh File Reference

```

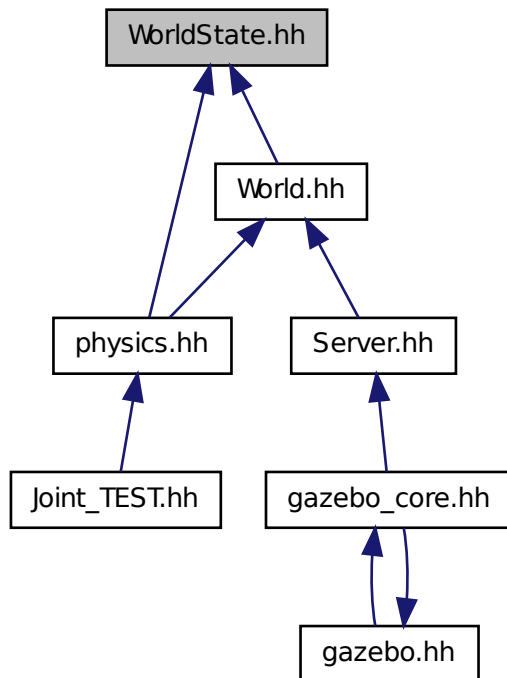
#include <string>
#include <vector>
#include <sdf/sdf.hh>
#include "gazebo/physics/State.hh"
#include "gazebo/physics/ModelState.hh"

```

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. 1070) object.

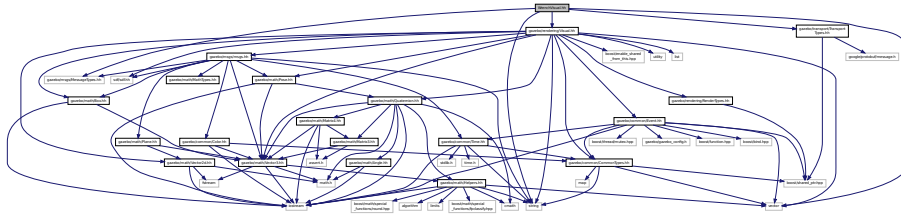
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.206 WrenchVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
#include "gazebo_msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.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.

Index

- ~Actor
 - gazebo::physics::Actor, 128
- ~Angle
 - gazebo::math::Angle, 133
- ~Animation
 - gazebo::common::Animation, 141
- ~ArrowVisual
 - gazebo::rendering::ArrowVisual, 145
- ~AssertionInternalError
 - gazebo::common::AssertionInternalError, 147
- ~AudioDecoder
 - gazebo::common::AudioDecoder, 147
- ~AxisVisual
 - gazebo::rendering::AxisVisual, 150
- ~BVHLoader
 - gazebo::common::BVHLoader, 173
- ~BallJoint
 - gazebo::physics::BallJoint, 152
- ~Base
 - gazebo::physics::Base, 157
- ~Box
 - gazebo::math::Box, 166
- ~BoxShape
 - gazebo::physics::BoxShape, 171
- ~COMVisual
 - gazebo::rendering::COMVisual, 239
- ~CallbackHelper
 - gazebo::transport::CallbackHelper, 175
- ~Camera
 - gazebo::rendering::Camera, 185
- ~CameraSensor
 - gazebo::sensors::CameraSensor, 207
- ~CameraVisual
 - gazebo::rendering::CameraVisual, 211
- ~ColladaLoader
 - gazebo::common::ColladaLoader, 212
- ~Collision
 - gazebo::physics::Collision, 216
- ~CollisionState
 - gazebo::physics::CollisionState, 224
- ~Color
 - gazebo::common::Color, 229
- ~Connection
 - gazebo::event::Connection, 240
 - gazebo::transport::Connection, 243
- ~Contact
 - gazebo::physics::Contact, 254
- ~ContactManager
 - gazebo::physics::ContactManager, 257
- ~ContactSensor
 - gazebo::sensors::ContactSensor, 262
- ~ContactVisual
 - gazebo::rendering::ContactVisual, 266
- ~CylinderShape
 - gazebo::physics::CylinderShape, 270
- ~DepthCamera
 - gazebo::rendering::DepthCamera, 274
- ~DepthCameraSensor
 - gazebo::sensors::DepthCameraSensor, 278
- ~DiagnosticTimer
 - gazebo::util::DiagnosticTimer, 284
- ~DynamicLines
 - gazebo::rendering::DynamicLines, 287
- ~DynamicRenderable
 - gazebo::rendering::DynamicRenderable, 291
- ~Entity
 - gazebo::physics::Entity, 296
- ~Event
 - gazebo::event::Event, 307
- ~EventT
 - Events, 43
- ~Exception
 - gazebo::common::Exception, 332
- ~FPSViewController
 - gazebo::rendering::FPSViewController, 339
- ~ForceTorqueSensor
 - gazebo::sensors::ForceTorqueSensor, 335
- ~GUIOverlay
 - gazebo::rendering::GUIOverlay, 371
- ~GazeboGenerator
 - google::protobuf::compiler::cpp::GazeboGenerator, 341
- ~GpsSensor
 - gazebo::sensors::GpsSensor, 342
- ~GpuLaser
 - gazebo::rendering::GpuLaser, 347
- ~GpuRaySensor
 - gazebo::sensors::GpuRaySensor, 356
- ~Grid
 - gazebo::rendering::Grid, 366

- ~Gripper
 - gazebo::physics::Gripper, 369
- ~GzTerrainMatGen
 - gazebo::rendering::GzTerrainMatGen, 375
- ~Heightmap
 - gazebo::rendering::Heightmap, 377
- ~HeightmapShape
 - gazebo::physics::HeightmapShape, 383
- ~Hinge2Joint
 - gazebo::physics::Hinge2Joint, 387
- ~HingeJoint
 - gazebo::physics::HingeJoint, 388
- ~IOManager
 - gazebo::transport::IOManager, 410
- ~Image
 - gazebo::common::Image, 391
- ~ImuSensor
 - gazebo::sensors::ImuSensor, 396
- ~Inertial
 - gazebo::physics::Inertial, 401
- ~InternalError
 - gazebo::common::InternalError, 409
- ~Joint
 - gazebo::physics::Joint, 415
- ~JointState
 - gazebo::physics::JointState, 438
- ~JointVisual
 - gazebo::rendering::JointVisual, 442
- ~KeyFrame
 - gazebo::common::KeyFrame, 446
- ~LaserVisual
 - gazebo::rendering::LaserVisual, 448
- ~Light
 - gazebo::rendering::Light, 450
- ~Link
 - gazebo::physics::Link, 460
- ~LinkState
 - gazebo::physics::LinkState, 478
- ~MapShape
 - gazebo::physics::MapShape, 494
- ~Master
 - gazebo::Master, 497
- ~Material
 - gazebo::common::Material, 501
- ~Matrix3
 - gazebo::math::Matrix3, 509
- ~Matrix4
 - gazebo::math::Matrix4, 514
- ~Mesh
 - gazebo::common::Mesh, 521
- ~MeshCSG
 - gazebo::common::MeshCSG, 527
- ~MeshLoader
 - gazebo::common::MeshLoader, 528
- ~MeshShape
 - gazebo::physics::MeshShape, 535
- ~Model
 - gazebo::physics::Model, 541
- ~ModelPlugin
 - gazebo::ModelPlugin, 554
- ~ModelState
 - gazebo::physics::ModelState, 557
- ~MovableText
 - gazebo::rendering::MovableText, 568
- ~MultiCameraSensor
 - gazebo::sensors::MultiCameraSensor, 575
- ~MultiRayShape
 - gazebo::physics::MultiRayShape, 581
- ~Node
 - gazebo::transport::Node, 588
- ~NodeAnimation
 - gazebo::common::NodeAnimation, 595
- ~NodeTransform
 - gazebo::common::NodeTransform, 600
- ~Noise
 - gazebo::sensors::Noise, 604
- ~NumericAnimation
 - gazebo::common::NumericAnimation, 607
- ~NumericKeyFrame
 - gazebo::common::NumericKeyFrame, 608
- ~OpenALSink
 - gazebo::util::OpenALSink, 612
- ~OpenALSource
 - gazebo::util::OpenALSource, 613
- ~OrbitViewController
 - gazebo::rendering::OrbitViewController, 618
- ~PID
 - gazebo::common::PID, 637
- ~PhysicsEngine
 - gazebo::physics::PhysicsEngine, 624
- ~Plane
 - gazebo::math::Plane, 641
- ~PlaneShape
 - gazebo::physics::PlaneShape, 644
- ~PluginT
 - gazebo::PluginT, 647
- ~Pose
 - gazebo::math::Pose, 651
- ~PoseAnimation
 - gazebo::common::PoseAnimation, 659
- ~PoseKeyFrame
 - gazebo::common::PoseKeyFrame, 661
- ~Projector
 - gazebo::rendering::Projector, 663
- ~Publication
 - gazebo::transport::Publication, 665
- ~PublicationTransport
 - gazebo::transport::PublicationTransport, 670

- ~Publisher
 - gazebo::transport::Publisher, 672
- ~Quaternion
 - gazebo::math::Quaternion, 679
- ~RFIDSensor
 - gazebo::sensors::RFIDSensor, 711
- ~RFIDTag
 - gazebo::sensors::RFIDTag, 714
- ~RFIDTagVisual
 - gazebo::rendering::RFIDTagVisual, 716
- ~RFIDVisual
 - gazebo::rendering::RFIDVisual, 717
- ~RaySensor
 - gazebo::sensors::RaySensor, 695
- ~RayShape
 - gazebo::physics::RayShape, 702
- ~Road
 - gazebo::physics::Road, 719
- ~Road2d
 - gazebo::rendering::Road2d, 720
- ~RotationSpline
 - gazebo::math::RotationSpline, 721
- ~SM2Profile
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 890
- ~STLLoader
 - gazebo::common::STLLoader, 916
- ~Scene
 - gazebo::rendering::Scene, 733
- ~ScrewJoint
 - gazebo::physics::ScrewJoint, 747
- ~SelectionObj
 - Rendering, 76
- ~Sensor
 - gazebo::sensors::Sensor, 754
- ~SensorPlugin
 - gazebo::SensorPlugin, 768
- ~Server
 - gazebo::Server, 770
- ~Shape
 - gazebo::physics::Shape, 777
- ~SimbodyBallJoint
 - gazebo::physics::SimbodyBallJoint, 780
- ~SimbodyBoxShape
 - gazebo::physics::SimbodyBoxShape, 785
- ~SimbodyCollision
 - gazebo::physics::SimbodyCollision, 787
- ~SimbodyCylinderShape
 - gazebo::physics::SimbodyCylinderShape, 790
- ~SimbodyHeightmapShape
 - gazebo::physics::SimbodyHeightmapShape, 792
- ~SimbodyHinge2Joint
 - gazebo::physics::SimbodyHinge2Joint, 794
- ~SimbodyHingeJoint
 - gazebo::physics::SimbodyHingeJoint, 800
- ~SimbodyJoint
 - gazebo::physics::SimbodyJoint, 807
- ~SimbodyLink
 - gazebo::physics::SimbodyLink, 816
- ~SimbodyMeshShape
 - gazebo::physics::SimbodyMeshShape, 824
- ~SimbodyModel
 - gazebo::physics::SimbodyModel, 826
- ~SimbodyMultiRayShape
 - gazebo::physics::SimbodyMultiRayShape, 828
- ~SimbodyPhysics
 - gazebo::physics::SimbodyPhysics, 831
- ~SimbodyPlaneShape
 - gazebo::physics::SimbodyPlaneShape, 839
- ~SimbodyRayShape
 - gazebo::physics::SimbodyRayShape, 841
- ~SimbodyScrewJoint
 - gazebo::physics::SimbodyScrewJoint, 845
- ~SimbodySliderJoint
 - gazebo::physics::SimbodySliderJoint, 851
- ~SimbodySphereShape
 - gazebo::physics::SimbodySphereShape, 857
- ~SimbodyUniversalJoint
 - gazebo::physics::SimbodyUniversalJoint, 860
- ~SingletonT
 - SingletonT, 866
- ~Skeleton
 - gazebo::common::Skeleton, 868
- ~SkeletonAnimation
 - gazebo::common::SkeletonAnimation, 874
- ~SkeletonNode
 - gazebo::common::SkeletonNode, 880
- ~SliderJoint
 - gazebo::physics::SliderJoint, 887
- ~SonarSensor
 - gazebo::sensors::SonarSensor, 892
- ~SonarVisual
 - gazebo::rendering::SonarVisual, 896
- ~SpawnJointOptions
 - Joint_TEST::SpawnJointOptions, 897
- ~SphereShape
 - gazebo::physics::SphereShape, 900
- ~SphericalCoordinates
 - gazebo::common::SphericalCoordinates, 903
- ~Spline
 - gazebo::math::Spline, 907
- ~State
 - gazebo::physics::State, 912
- ~SubMesh
 - gazebo::common::SubMesh, 919
- ~Subscriber
 - gazebo::transport::Subscriber, 930
- ~SubscriptionTransport

- gazebo::transport::SubscriptionTransport, 931
- ~SurfaceParams
 - gazebo::physics::SurfaceParams, 934
- ~SystemPlugin
 - gazebo::SystemPlugin, 943
- ~Time
 - gazebo::common::Time, 949
- ~Timer
 - gazebo::common::Timer, 966
- ~TransmitterVisual
 - gazebo::rendering::TransmitterVisual, 976
- ~UniversalJoint
 - gazebo::physics::UniversalJoint, 977
- ~UserCamera
 - gazebo::rendering::UserCamera, 981
- ~Vector2d
 - gazebo::math::Vector2d, 989
- ~Vector2i
 - gazebo::math::Vector2i, 998
- ~Vector3
 - gazebo::math::Vector3, 1008
- ~Vector4
 - gazebo::math::Vector4, 1021
- ~Video
 - gazebo::common::Video, 1028
- ~VideoVisual
 - gazebo::rendering::VideoVisual, 1030
- ~ViewController
 - gazebo::rendering::ViewController, 1032
- ~Visual
 - gazebo::rendering::Visual, 1040
- ~WindowManager
 - gazebo::rendering::WindowManager, 1057
- ~WireBox
 - gazebo::rendering::WireBox, 1060
- ~WirelessReceiver
 - gazebo::sensors::WirelessReceiver, 1062
- ~WirelessTransceiver
 - gazebo::sensors::WirelessTransceiver, 1064
- ~WirelessTransmitter
 - gazebo::sensors::WirelessTransmitter, 1068
- ~World
 - gazebo::physics::World, 1073
- ~WorldPlugin
 - gazebo::WorldPlugin, 1082
- ~WorldState
 - gazebo::physics::WorldState, 1085
- ~WrenchVisual
 - gazebo::rendering::WrenchVisual, 1091
- _setupGeometry
 - gazebo::rendering::MovableText, 569
- _updateColors
 - gazebo::rendering::MovableText, 569
- a
 - gazebo::common::Color, 237
- ABGR
 - gazebo::common::Color, 229
- ACTOR
 - gazebo::physics::Base, 156
- ADD
 - gazebo::common::Material, 500
- ARGB
 - gazebo::common::Color, 229
- AcceptCallback
 - gazebo::transport::Connection, 243
- active
 - gazebo::physics::Actor, 129
 - gazebo::sensors::Sensor, 761
- Actor
 - gazebo::physics::Actor, 127
- Actor.hh, 1093
- Actor_V
 - gazebo::physics, 109
- ActorPtr
 - gazebo::physics, 109
- Add
 - gazebo::util::LogRecord, 488
- add_plugin
 - gazebo, 95
- add_search_path_suffix
 - Common, 36
- AddAnimation
 - gazebo::common::Skeleton, 868
- AddCallback
 - gazebo::transport::PublicationTransport, 670
- AddChild
 - gazebo::common::SkeletonNode, 880
 - gazebo::physics::Base, 157
- AddChildJoint
 - gazebo::physics::Link, 460
- AddContact
 - gazebo::physics::Collision, 216
- addEntity
 - gazebo::event::Events, 320
- AddForce
 - gazebo::physics::Link, 460
 - gazebo::physics::SimbodyLink, 816
- AddForceAtRelativePosition
 - gazebo::physics::Link, 460
 - gazebo::physics::SimbodyLink, 816
- AddForceAtWorldPosition
 - gazebo::physics::Link, 461
 - gazebo::physics::SimbodyLink, 817
- AddGazeboPaths
 - gazebo::common::SystemPaths, 939
- AddIndex
 - gazebo::common::SubMesh, 919

- AddJoint
 - gazebo::physics::JointController, 435
- AddKeyFrame
 - gazebo::common::NodeAnimation, 595
 - gazebo::common::SkeletonAnimation, 874
- AddMaterial
 - gazebo::common::Mesh, 521
- AddMesh
 - gazebo::common::MeshManager, 530
- AddModelPaths
 - gazebo::common::SystemPaths, 939
- AddNode
 - gazebo::transport::TopicManager, 969
- AddNodeAssignment
 - gazebo::common::SubMesh, 919
- AddNodeToProcess
 - gazebo::transport::TopicManager, 970
- AddNormal
 - gazebo::common::SubMesh, 920
- AddOgrePaths
 - gazebo::common::SystemPaths, 939
- AddParentJoint
 - gazebo::physics::Link, 461
- AddPluginPaths
 - gazebo::common::SystemPaths, 940
- AddPoint
 - gazebo::math::RotationSpline, 722
 - gazebo::math::Spline, 907
 - gazebo::rendering::DynamicLines, 287, 288
- AddPublisher
 - gazebo::transport::Publication, 666
- AddRawTransform
 - gazebo::common::SkeletonNode, 880
- AddRay
 - gazebo::physics::MultiRayShape, 581
 - gazebo::physics::SimbodyMultiRayShape, 828
- AddRelativeForce
 - gazebo::physics::Link, 461
 - gazebo::physics::SimbodyLink, 817
- AddRelativeTorque
 - gazebo::physics::Link, 461
 - gazebo::physics::SimbodyLink, 817
- AddResourcePath
 - gazebo::rendering::RenderEngine, 708
- AddScene
 - gazebo::rendering::RTShaderSystem, 726
- AddSearchPathSuffix
 - gazebo::common::SystemPaths, 940
- AddSubMesh
 - gazebo::common::Mesh, 521
- AddSubscription
 - gazebo::transport::Publication, 666
- AddTag
 - gazebo::sensors::RFIDSensor, 711
- addTechnique
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 890
- AddTexCoord
 - gazebo::common::SubMesh, 920
- AddTime
 - gazebo::common::Animation, 141
- AddTorque
 - gazebo::physics::Link, 461
 - gazebo::physics::SimbodyLink, 817
- AddTransport
 - gazebo::transport::Publication, 666
- AddType
 - gazebo::physics::Base, 157
- AddVertNodeWeight
 - gazebo::common::Skeleton, 868
- AddVertex
 - gazebo::common::SubMesh, 920
- AddVisual
 - gazebo::rendering::Scene, 733
- Advertise
 - gazebo::transport::ConnectionManager, 249
 - gazebo::transport::Node, 589
 - gazebo::transport::TopicManager, 970
- alt
 - gazebo::common::MouseEvent, 565
- ambient
 - gazebo::common::Material, 506
- anchorLink
 - gazebo::physics::Joint, 428
- anchorPos
 - gazebo::physics::Joint, 428
- anchorPose
 - gazebo::physics::Joint, 429
- Angle
 - gazebo::math::Angle, 133
- Angle.hh, 1094
 - GZ_DTOR, 1096
 - GZ_NORMALIZE, 1096
 - GZ_RTOD, 1096
- angularAccel
 - gazebo::physics::Link, 475
- animState
 - gazebo::rendering::Camera, 203
- Animation
 - gazebo::common::Animation, 141
- animation
 - gazebo::physics::Entity, 304
- Animation.hh, 1097
- AnimationComplete
 - gazebo::rendering::Camera, 186
 - gazebo::rendering::UserCamera, 981
- animationConnection
 - gazebo::physics::Entity, 304

- AnimationPtr
 - gazebo::common, 98
- animationStartPose
 - gazebo::physics::Entity, 304
- animations
 - gazebo::common::SkeletonAnimation, 876
- anims
 - gazebo::common::Skeleton, 872
- Apply
 - gazebo::sensors::Noise, 604
- ApplyDamping
 - gazebo::physics::Joint, 415
- applyDamping
 - gazebo::physics::Joint, 429
- ApplyShadows
 - gazebo::rendering::RTShaderSystem, 726
- AreConnected
 - gazebo::physics::Joint, 415
 - gazebo::physics::SimbodyJoint, 807
- ArrowVisual
 - gazebo::rendering::ArrowVisual, 144
- ArrowVisual.hh, 1098
- ArrowVisualPtr
 - gazebo::rendering, 114
- Assert.hh, 1099
 - GZ_ASSERT, 1100
- AssertionInternalError
 - gazebo::common::AssertionInternalError, 146
- AsyncRead
 - gazebo::transport::Connection, 243
- Attach
 - gazebo::physics::Joint, 416
 - Rendering, 76
- AttachAxes
 - gazebo::rendering::Visual, 1040
- AttachCameraToImage
 - gazebo::rendering::GUIOverlay, 371, 372
- AttachEntity
 - gazebo::rendering::RTShaderSystem, 726
- AttachLineVertex
 - gazebo::rendering::Visual, 1040
- AttachMesh
 - gazebo::rendering::Visual, 1041
- AttachObject
 - gazebo::rendering::Visual, 1041
- AttachStaticModel
 - gazebo::physics::Link, 462
 - gazebo::physics::Model, 541
- AttachToVisual
 - gazebo::rendering::Camera, 186
- AttachToVisualImpl
 - gazebo::rendering::Camera, 186, 187
 - gazebo::rendering::UserCamera, 981
- AttachViewport
 - gazebo::rendering::RTShaderSystem, 726
- AttachVisual
 - gazebo::rendering::Visual, 1041
- attachedModels
 - gazebo::physics::Model, 551
- attachedModelsOffset
 - gazebo::physics::Link, 475
 - gazebo::physics::Model, 551
- Attribute
 - gazebo::physics::Joint, 415
- AudioDecoder
 - gazebo::common::AudioDecoder, 147
- AudioDecoder.hh, 1101
- autoCalc
 - gazebo::math::RotationSpline, 724
 - gazebo::math::Spline, 910
- autoStart
 - gazebo::physics::Actor, 129
- axis
 - Joint_TEST::SpawnJointOptions, 897
- AxisVisual
 - gazebo::rendering::AxisVisual, 150
- AxisVisual.hh, 1102
- AxisVisualPtr
 - gazebo::rendering, 114
- b
 - gazebo::common::Color, 237
- BALL_JOINT
 - gazebo::physics::Base, 156
- BASE
 - gazebo::physics::Base, 156
- BAYER_GBRG8
 - gazebo::common::Image, 391
- BAYER_GRBG8
 - gazebo::common::Image, 391
- BAYER_RGGB8
 - gazebo::common::Image, 391
- BAYER_RGGR8
 - gazebo::common::Image, 391
- BGR_INT16
 - gazebo::common::Image, 391
- BGR_INT32
 - gazebo::common::Image, 391
- BGR_INT8
 - gazebo::common::Image, 391
- BGRA
 - gazebo::common::Color, 229
- BGRA_INT8
 - gazebo::common::Image, 391
- BLEND_COUNT
 - gazebo::common::Material, 500
- BLINN
 - gazebo::common::Material, 501

- BOX_SHAPE
 - gazebo::physics::Base, 156
- BVHLoader
 - gazebo::common::BVHLoader, 173
- BVHLoader.hh, 1108
 - X_POSITION, 1110
 - X_ROTATION, 1110
 - Y_POSITION, 1110
 - Y_ROTATION, 1110
 - Z_POSITION, 1110
 - Z_ROTATION, 1110
- BallJoint
 - gazebo::physics::BallJoint, 152
- BallJoint.hh, 1102
- Base
 - gazebo::physics::Base, 157
- Base.hh, 1103
- Base64.hh, 1104
 - Base64Decode, 1106
 - Base64Encode, 1106
- Base64Decode
 - Base64.hh, 1106
- Base64Encode
 - Base64.hh, 1106
- Base_V
 - gazebo::physics, 109
- BasePtr
 - gazebo::physics, 109
- bayerFrameBuffer
 - gazebo::rendering::Camera, 203
- bindShapeTransform
 - gazebo::common::Skeleton, 872
- Black
 - gazebo::common::Color, 237
- BlendMode
 - gazebo::common::Material, 500
- blendMode
 - gazebo::common::Material, 506
- BlendModeStr
 - gazebo::common::Material, 506
- Blue
 - gazebo::common::Color, 237
- body1Force
 - gazebo::physics::JointWrench, 443
- body1Torque
 - gazebo::physics::JointWrench, 444
- body2Force
 - gazebo::physics::JointWrench, 444
- body2Torque
 - gazebo::physics::JointWrench, 444
- bonePosePub
 - gazebo::physics::Actor, 129
- BooleanOperation
 - gazebo::common::MeshCSG, 526
- boost, 93
- bounce
 - gazebo::physics::SurfaceParams, 935
- bounceThreshold
 - gazebo::physics::SurfaceParams, 935
- Box
 - gazebo::math::Box, 165, 166
- Box.hh, 1106
- BoxShape
 - gazebo::physics::BoxShape, 171
- BoxShape.hh, 1107
- BoxShapePtr
 - gazebo::physics, 109
- build
 - gazebo::common::Animation, 143
- BuildInterpolationSplines
 - gazebo::common::PoseAnimation, 659
- BuildNodeMap
 - gazebo::common::Skeleton, 869
- button
 - gazebo::common::MouseEvent, 565
- ButtonCallback
 - gazebo::rendering::GUIOverlay, 372
- Buttons
 - gazebo::common::MouseEvent, 564
- buttons
 - gazebo::common::MouseEvent, 565
- CATEGORY_COUNT
 - gazebo::sensors, 119
- CFM
 - gazebo::physics::Joint, 415
- COLLISION
 - gazebo::physics::Base, 156
- COMVisual
 - gazebo::rendering::COMVisual, 238
- COMVisual.hh, 1122
- COMVisualPtr
 - gazebo::rendering, 114
- COR3_MAX
 - STLLoader.hh, 1280
- CYLINDER_SHAPE
 - gazebo::physics::Base, 156
- CacheForceTorque
 - gazebo::physics::Joint, 416
 - gazebo::physics::SimbodyJoint, 807
- CallbackHelper
 - gazebo::transport::CallbackHelper, 175
- CallbackHelper.hh, 1110
- CallbackHelperPtr
 - Transport, 87
- CallbackHelperT
 - gazebo::transport::CallbackHelperT, 177
- Camera

- gazebo::rendering::Camera, 185
- camera
 - gazebo::rendering::Camera, 203
 - gazebo::rendering::ViewController, 1034
- Camera.hh, 1112
- cameraCount
 - gazebo::rendering::GpuLaser, 352
- cameraElem
 - gazebo::sensors::GpuRaySensor, 364
- CameraPtr
 - gazebo::rendering, 114
- CameraSensor
 - gazebo::sensors::CameraSensor, 207
- CameraSensor.hh, 1113
- CameraSensor_V
 - gazebo::sensors, 118
- CameraSensorPtr
 - gazebo::sensors, 118
- CameraVisual
 - gazebo::rendering::CameraVisual, 211
- CameraVisual.hh, 1113
- CameraVisualPtr
 - gazebo::rendering, 114
- Cancel
 - gazebo::transport::Connection, 243
- captureData
 - gazebo::rendering::Camera, 203
- captureDataOnce
 - gazebo::rendering::Camera, 203
- cegui.h, 1114
- Center
 - gazebo::common::Mesh, 521
 - gazebo::common::SubMesh, 921
- cfm
 - gazebo::physics::SurfaceParams, 935
- cgVisuals
 - gazebo::physics::Link, 475
- CheckAndTruncateForce
 - gazebo::physics::Joint, 416
- chfov
 - gazebo::rendering::GpuLaser, 352
- childLink
 - gazebo::physics::Joint, 429
- childLinkPose
 - Joint_TEST::SpawnJointOptions, 897
- children
 - gazebo::common::SkeletonNode, 885
 - gazebo::physics::Base, 164
- childrenEnd
 - gazebo::physics::Base, 164
- clamp
 - Math, 48
- Classes for physics and dynamics, 63
 - create_world, 67
 - EntityTypename, 69
 - fini, 67
 - GZ_REGISTER_PHYSICS_ENGINE, 66
 - get_world, 67
 - getUniqueld, 67
 - init_world, 67
 - init_worlds, 68
 - load, 68
 - load_world, 68
 - load_worlds, 68
 - pause_world, 68
 - pause_worlds, 68
 - PhysicsFactoryFn, 67
 - remove_worlds, 69
 - run_world, 69
 - run_worlds, 69
 - stop_world, 69
 - stop_worlds, 69
 - worlds_running, 69
- Clear
 - gazebo::math::RotationSpline, 722
 - gazebo::math::Spline, 907
 - gazebo::physics::ContactManager, 257
 - gazebo::physics::World, 1073
 - gazebo::rendering::DynamicLines, 288
 - gazebo::rendering::RTShaderSystem, 727
 - gazebo::rendering::Scene, 733
- clear_buffers
 - Transport, 87
- ClearBuffers
 - gazebo::transport::TopicManager, 970
- ClearGazeboPaths
 - gazebo::common::SystemPaths, 940
- ClearModelPaths
 - gazebo::common::SystemPaths, 940
- ClearOgrePaths
 - gazebo::common::SystemPaths, 940
- ClearParent
 - gazebo::rendering::Visual, 1041
- ClearPluginPaths
 - gazebo::common::SystemPaths, 940
- Clone
 - gazebo::rendering::Visual, 1041
- CloneVisual
 - gazebo::rendering::Scene, 733
- coeffs
 - gazebo::math::Spline, 910
- ColladaLoader
 - gazebo::common::ColladaLoader, 212
- ColladaLoader.hh, 1115
- collideWithoutContact
 - gazebo::physics::SurfaceParams, 935
- collideWithoutContactBitmask
 - gazebo::physics::SurfaceParams, 935

- Collision
 - gazebo::physics::Collision, 216
- Collision.hh, 1116
- collision1
 - gazebo::physics::Contact, 255
- collision2
 - gazebo::physics::Contact, 255
- Collision_V
 - gazebo::physics, 109
- collisionNames
 - gazebo::physics::ContactPublisher, 260
- collisionParent
 - gazebo::physics::Shape, 778
- CollisionPtr
 - gazebo::physics, 109
- CollisionState
 - gazebo::physics::CollisionState, 224
- CollisionState.hh, 1117
- collisions
 - gazebo::physics::ContactPublisher, 260
- Color
 - gazebo::common::Color, 229
- Color.hh, 1117
- ColorErr
 - Common, 37
- ColorMsg
 - Common, 37
- Common, 31
 - add_search_path_suffix, 36
 - ColorErr, 37
 - ColorMsg, 37
 - DownloadDependencies, 37
 - find_file, 37, 38
 - find_file_path, 38
 - Fini, 38
 - GetDBConfig, 38
 - GetModelConfig, 38
 - GetModelFile, 38
 - GetModelName, 39
 - GetModelPath, 39
 - GetModels, 39
 - GetQuiet, 40
 - GetURI, 40
 - gzclr_end, 35
 - gzclr_start, 35
 - gzdbg, 35
 - gzerr, 35
 - gzlog, 35
 - gzmsg, 35
 - gzthrow, 36
 - gzwarn, 36
 - HasModel, 40
 - Init, 40
 - IsInitialized, 40
 - load, 40
 - Log, 41
 - MODEL_PLUGIN, 36
 - NullStream, 36
 - PixelFormatNames, 41
 - PluginType, 36
 - SENSOR_PLUGIN, 36
 - SYSTEM_PLUGIN, 36
 - SetQuiet, 41
 - Start, 41
 - VISUAL_PLUGIN, 36
 - WORLD_PLUGIN, 36
- Commonface.hh, 1118
- CommonTypes.hh, 1120
 - GAZEBO_DEPRECATED, 1122
 - GAZEBO_FORCEINLINE, 1122
 - NULL, 1122
- ComputeScopedName
 - gazebo::physics::Base, 157
- Connect
 - Events, 43
 - gazebo::transport::Connection, 243
- ConnectAddEntity
 - gazebo::event::Events, 313
- ConnectCreateEntity
 - gazebo::event::Events, 313
- ConnectCreateScene
 - gazebo::rendering::Events, 308
- ConnectDeleteEntity
 - gazebo::event::Events, 313
- ConnectDiagTimerStart
 - gazebo::event::Events, 313
- ConnectDiagTimerStop
 - gazebo::event::Events, 314
- ConnectEnabled
 - gazebo::physics::Link, 462
- ConnectJointUpdate
 - gazebo::physics::Joint, 416
- ConnectNewDepthFrame
 - gazebo::rendering::DepthCamera, 274
- ConnectNewImageFrame
 - gazebo::rendering::Camera, 187
- ConnectNewLaserFrame
 - gazebo::rendering::GpuLaser, 347
 - gazebo::sensors::GpuRaySensor, 357
- ConnectNewLaserScans
 - gazebo::physics::MultiRayShape, 582
- ConnectNewRGBPointCloud
 - gazebo::rendering::DepthCamera, 274
- ConnectPause
 - gazebo::event::Events, 314
- ConnectPostRender
 - gazebo::event::Events, 314
- ConnectPreRender

- gazebo::event::Events, 315
- ConnectPubToSub
 - gazebo::transport::TopicManager, 970
- ConnectRemoveScene
 - gazebo::rendering::Events, 309
- ConnectRender
 - gazebo::event::Events, 315
- ConnectSetSelectedEntity
 - gazebo::event::Events, 315
- ConnectSigInt
 - gazebo::event::Events, 315
- ConnectStep
 - gazebo::event::Events, 316
- ConnectStop
 - gazebo::event::Events, 316
- ConnectSubToPub
 - gazebo::transport::TopicManager, 970
- ConnectSubscribers
 - gazebo::transport::TopicManager, 970
- ConnectToRemoteHost
 - gazebo::transport::ConnectionManager, 249
- ConnectToShutdown
 - gazebo::transport::Connection, 243
- ConnectUpdate
 - gazebo::sensors::ForceTorqueSensor, 335
 - gazebo::sensors::SonarSensor, 892
- ConnectUpdated
 - gazebo::sensors::Sensor, 755
- ConnectWorldCreated
 - gazebo::event::Events, 316
- ConnectWorldUpdateBegin
 - gazebo::event::Events, 317
- ConnectWorldUpdateEnd
 - gazebo::event::Events, 317
- Connection
 - gazebo::event::Connection, 240
 - gazebo::transport::Connection, 243
- Connection.hh, 1123
 - HEADER_LENGTH, 1125
- Connection_V
 - gazebo::event, 99
- ConnectionCount
 - Events, 44
- ConnectionManager.hh, 1125
- ConnectionPtr
 - gazebo::event, 99
 - gazebo::transport, 121
- connections
 - gazebo::physics::Entity, 304
 - gazebo::rendering::Camera, 204
 - gazebo::sensors::Sensor, 761
- Console.hh, 1127
- constraint
 - gazebo::physics::SimbodyJoint, 812
- Contact
 - gazebo::physics::Contact, 254
- contact
 - gazebo::physics::SimbodyPhysics, 837
- Contact.hh, 1128
 - MAX_COLLIDE_RETURNS, 1129
 - MAX_CONTACT_JOINTS, 1129
- contactFiducial
 - gazebo::physics::RayShape, 705
- contactLen
 - gazebo::physics::RayShape, 705
- ContactManager
 - gazebo::physics::ContactManager, 257
- contactManager
 - gazebo::physics::PhysicsEngine, 633
- ContactManager.hh, 1130
- ContactPtr
 - gazebo::physics, 109
- contactRetro
 - gazebo::physics::RayShape, 705
- ContactSensor
 - gazebo::sensors::ContactSensor, 262
- ContactSensor.hh, 1131
- ContactSensor_V
 - gazebo::sensors, 118
- ContactSensorPtr
 - gazebo::sensors, 118
- ContactVisual
 - gazebo::rendering::ContactVisual, 266
- ContactVisual.hh, 1131
- ContactVisualPtr
 - gazebo::rendering, 114
- contacts
 - gazebo::physics::ContactPublisher, 260
- control
 - gazebo::common::MouseEvent, 565
- Conversions.hh, 1132
- Convert
 - gazebo::common::SphericalCoordinates, 903
 - gazebo::rendering::Conversions, 267, 268
 - Messages, 54–57
- ConvertPixelFormat
 - gazebo::common::Image, 391
- CoordPoseSolve
 - gazebo::math::Pose, 651
- CoordPositionAdd
 - gazebo::math::Pose, 651, 652
- CoordPositionSub
 - gazebo::math::Pose, 652
- CoordRotationAdd
 - gazebo::math::Pose, 652
- CoordRotationSub
 - gazebo::math::Pose, 652
- CopyNormals

- gazebo::common::SubMesh, 921
- CopyVertices
 - gazebo::common::SubMesh, 921
- Correct
 - gazebo::math::Pose, 653
 - gazebo::math::Quaternion, 679
 - gazebo::math::Vector3, 1008
- count
 - gazebo::physics::Contact, 255
- Create
 - gazebo::PluginT, 647
- create_scene
 - Rendering, 77
- create_sensor
 - Sensors, 82
- create_world
 - Classes for physics and dynamics, 67
- CreateBoolean
 - gazebo::common::MeshCSG, 527
- CreateBox
 - gazebo::common::MeshManager, 530
- CreateCamera
 - gazebo::common::MeshManager, 530
 - gazebo::rendering::Scene, 733
- CreateCollision
 - gazebo::physics::PhysicsEngine, 624
 - gazebo::physics::SimbodyPhysics, 831
- CreateCone
 - gazebo::common::MeshManager, 531
- CreateCylinder
 - gazebo::common::MeshManager, 531
- CreateDepthCamera
 - gazebo::rendering::Scene, 733
- CreateDepthTexture
 - gazebo::rendering::DepthCamera, 274
- CreateDynamicLine
 - gazebo::rendering::Visual, 1041
- CreateFilter
 - gazebo::physics::ContactManager, 257, 258
- CreateGpuLaser
 - gazebo::rendering::Scene, 734
- CreateGrid
 - gazebo::rendering::Scene, 734
- CreateJoint
 - gazebo::physics::PhysicsEngine, 625
 - gazebo::physics::SimbodyPhysics, 831
- CreateKeyFrame
 - gazebo::common::NumericAnimation, 607
 - gazebo::common::PoseAnimation, 659
- CreateLaserTexture
 - gazebo::rendering::GpuLaser, 347
- CreateLink
 - gazebo::physics::PhysicsEngine, 625
 - gazebo::physics::SimbodyPhysics, 831
- CreateModel
 - gazebo::physics::PhysicsEngine, 625
 - gazebo::physics::SimbodyPhysics, 832
- CreatePlane
 - gazebo::common::MeshManager, 531
 - gazebo::physics::PlaneShape, 644
 - gazebo::physics::SimbodyPlaneShape, 839
- CreateRenderTexture
 - gazebo::rendering::Camera, 188
- CreateRequest
 - Messages, 57
- CreateScene
 - gazebo::rendering::RenderEngine, 708
- createScene
 - gazebo::rendering::Events, 309
- CreateSensor
 - gazebo::sensors::SensorManager, 765
- CreateShape
 - gazebo::physics::PhysicsEngine, 625
 - gazebo::physics::SimbodyPhysics, 832
- CreateSink
 - gazebo::util::OpenAL, 610
- CreateSource
 - gazebo::util::OpenAL, 610
- CreateSphere
 - gazebo::common::MeshManager, 532
- CreateTube
 - gazebo::common::MeshManager, 532
- CreateUserCamera
 - gazebo::rendering::Scene, 734
- CreateVertexDeclaration
 - gazebo::rendering::DynamicRenderable, 291
- CreateWindow
 - gazebo::rendering::GUIOverlay, 372
 - gazebo::rendering::WindowManager, 1058
- Cross
 - gazebo::math::Vector2d, 990
 - gazebo::math::Vector2i, 998
 - gazebo::math::Vector3, 1008
- cvfov
 - gazebo::rendering::GpuLaser, 352
- CylinderShape
 - gazebo::physics::CylinderShape, 270
- CylinderShape.hh, 1133
- CylinderShapePtr
 - gazebo::physics, 109
- d
 - gazebo::math::Plane, 642
- DEFERRED
 - gazebo::rendering::RenderEngine, 707
- DIAG_TIMER_LAP
 - Utility, 91
- DIAG_TIMER_START

- Utility, 91
- DIAG_TIMER_STOP
 - Utility, 91
- DIFFERENCE
 - gazebo::common::MeshCSG, 527
- damper
 - gazebo::physics::SimbodyJoint, 812
- dampingCoefficient
 - gazebo::physics::Joint, 429
- DebugPrint
 - gazebo::physics::PhysicsEngine, 625
 - gazebo::physics::SimbodyPhysics, 832
- DebugString
 - gazebo::physics::Contact, 254
- DecCount
 - gazebo::transport::IOManager, 410
- Decode
 - gazebo::common::AudioDecoder, 148
- DecodeTopicName
 - gazebo::transport::Node, 589
- defaultVpParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperCg, 772
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperGLSL, 774
- defxAB
 - gazebo::physics::SimbodyJoint, 812
- Degree
 - gazebo::math::Angle, 134
- DeleteDynamicLine
 - gazebo::rendering::Visual, 1042
- deleteEntity
 - gazebo::event::Events, 320
- DepthCamera
 - gazebo::rendering::DepthCamera, 274
- DepthCamera.hh, 1134
- DepthCameraPtr
 - gazebo::rendering, 114
- DepthCameraSensor
 - gazebo::sensors::DepthCameraSensor, 278
- DepthCameraSensor.hh, 1135
- DepthCameraSensor_V
 - gazebo::sensors, 118
- DepthCameraSensorPtr
 - gazebo::sensors, 118
- depthTarget
 - gazebo::rendering::DepthCamera, 276
- depthTexture
 - gazebo::rendering::DepthCamera, 276
- depthViewport
 - gazebo::rendering::DepthCamera, 276
- depths
 - gazebo::physics::Contact, 255
- Detach
 - gazebo::physics::Joint, 416
 - gazebo::physics::SimbodyJoint, 807
 - Rendering, 77
- DetachAllStaticModels
 - gazebo::physics::Link, 462
- DetachEntity
 - gazebo::rendering::RTShaderSystem, 727
- DetachObjects
 - gazebo::rendering::Visual, 1042
- DetachStaticModel
 - gazebo::physics::Link, 462
 - gazebo::physics::Model, 542
- DetachViewport
 - gazebo::rendering::RTShaderSystem, 727
- DetachVisual
 - gazebo::rendering::Visual, 1042
- diagTimerStart
 - gazebo::event::Events, 321
- diagTimerStop
 - gazebo::event::Events, 321
- DiagnosticTimer
 - gazebo::util::DiagnosticTimer, 283
- DiagnosticTimerPtr
 - gazebo::common, 98
 - gazebo::util, 122
- Diagnostics.hh, 1136
- diffuse
 - gazebo::common::Material, 506
- dirtyPose
 - gazebo::physics::Entity, 304
- dirtyPoses
 - gazebo::physics::World, 1081
- DisableAllModels
 - gazebo::physics::World, 1073
- DisableTrackVisual
 - gazebo::rendering::Visual, 1042
- Disconnect
 - Events, 44, 45
 - gazebo::event::Event, 307
- DisconnectAddEntity
 - gazebo::event::Events, 317
- DisconnectCreateEntity
 - gazebo::event::Events, 317
- DisconnectCreateScene
 - gazebo::rendering::Events, 309
- DisconnectDeleteEntity
 - gazebo::event::Events, 318
- DisconnectDiagTimerStart
 - gazebo::event::Events, 318
- DisconnectDiagTimerStop
 - gazebo::event::Events, 318
- DisconnectEnabled
 - gazebo::physics::Link, 462
- DisconnectJointUpdate

- gazebo::physics::Joint, 417
- DisconnectNewDepthFrame
 - gazebo::rendering::DepthCamera, 275
- DisconnectNewImageFrame
 - gazebo::rendering::Camera, 188
- DisconnectNewLaserFrame
 - gazebo::rendering::GpuLaser, 347
 - gazebo::sensors::GpuRaySensor, 357
- DisconnectNewLaserScans
 - gazebo::physics::MultiRayShape, 582
- DisconnectNewRGBPointCloud
 - gazebo::rendering::DepthCamera, 275
- DisconnectPause
 - gazebo::event::Events, 318
- DisconnectPostRender
 - gazebo::event::Events, 318
- DisconnectPreRender
 - gazebo::event::Events, 319
- DisconnectPubFromSub
 - gazebo::transport::TopicManager, 971
- DisconnectRemoveScene
 - gazebo::rendering::Events, 309
- DisconnectRender
 - gazebo::event::Events, 319
- DisconnectSetSelectedEntity
 - gazebo::event::Events, 319
- DisconnectShutdown
 - gazebo::transport::Connection, 244
- DisconnectSigInt
 - gazebo::event::Events, 319
- DisconnectStep
 - gazebo::event::Events, 319
- DisconnectStop
 - gazebo::event::Events, 320
- DisconnectSubFromPub
 - gazebo::transport::TopicManager, 971
- DisconnectUpdate
 - gazebo::sensors::ForceTorqueSensor, 335
 - gazebo::sensors::SonarSensor, 892
- DisconnectUpdated
 - gazebo::sensors::Sensor, 755
- DisconnectWorldCreated
 - gazebo::event::Events, 320
- DisconnectWorldUpdateBegin
 - gazebo::event::Events, 320
- DisconnectWorldUpdateEnd
 - gazebo::event::Events, 320
- discreteForces
 - gazebo::physics::SimbodyPhysics, 837
- Distance
 - gazebo::math::Plane, 641
 - gazebo::math::Vector2d, 990
 - gazebo::math::Vector2i, 998
 - gazebo::math::Vector3, 1008, 1009
 - gazebo::math::Vector4, 1021
- Dot
 - gazebo::math::Quaternion, 679
 - gazebo::math::Vector3, 1009
- Double
 - gazebo::common::Time, 949
- DownloadDependencies
 - Common, 37
- dragging
 - gazebo::common::MouseEvent, 565
- DrawLine
 - gazebo::rendering::Scene, 735
- dummyContext
 - gazebo::rendering::RenderEngine, 709
- dummyDisplay
 - gazebo::rendering::RenderEngine, 709
- dummyWindowId
 - gazebo::rendering::RenderEngine, 710
- duration
 - gazebo::physics::TrajectoryInfo, 974
- DynamicLines
 - gazebo::rendering::DynamicLines, 287
- DynamicLines.hh, 1137
- DynamicLinesPtr
 - gazebo::rendering, 114
- DynamicRenderable
 - gazebo::rendering::DynamicRenderable, 291
- DynamicRenderable.hh, 1137
- EARTH_WGS84
 - gazebo::common::SphericalCoordinates, 902
- ENTITY
 - gazebo::physics::Base, 156
- ERP
 - gazebo::physics::Joint, 415
- effortLimit
 - gazebo::physics::Joint, 429
- emissive
 - gazebo::common::Material, 506
- Enable
 - gazebo::rendering::Grid, 366
- EnableAllModels
 - gazebo::physics::World, 1073
- EnablePhysicsEngine
 - gazebo::physics::World, 1073
- EnableSaveFrame
 - gazebo::rendering::Camera, 188
- EnableTrackVisual
 - gazebo::rendering::Visual, 1042
- EnableViewController
 - gazebo::rendering::UserCamera, 982
- enabled
 - gazebo::rendering::ViewController, 1034
- EncodeTopicName

- gazebo::transport::Node, 589
- endTime
 - gazebo::physics::TrajectoryInfo, 974
- EnqueueMsg
 - gazebo::transport::Connection, 244
- Entity
 - gazebo::physics::Entity, 296
- Entity.hh, 1138
- entityCreated
 - gazebo::event::Events, 321
- EntityPtr
 - gazebo::physics, 109
- EntityType
 - gazebo::physics::Base, 156
- EntityTypename
 - Classes for physics and dynamics, 69
- Equal
 - gazebo::math::Vector3, 1009
- equal
 - Math, 48
- erp
 - gazebo::physics::SurfaceParams, 935
- EulerToQuaternion
 - gazebo::math::Quaternion, 679
- Event.hh, 1139
- eventConnections
 - gazebo::transport::ConnectionManager, 252
- EventType
 - gazebo::common::KeyEvent, 445
 - gazebo::common::MouseEvent, 564
- Events, 43
 - ~EventT, 43
 - Connect, 43
 - ConnectionCount, 44
 - Disconnect, 44, 45
- Events.hh, 1140
- Exception
 - gazebo::common::Exception, 332
- Exception.hh, 1141
- FACE_MAX
 - STLLoader.hh, 1280
- FLAT
 - gazebo::common::Material, 501
- FMAX
 - gazebo::physics::Joint, 415
- FORWARD
 - gazebo::rendering::RenderEngine, 707
- FPSViewController
 - gazebo::rendering::FPSViewController, 339
- FPSViewController.hh, 1143
- FUDGE_FACTOR
 - gazebo::physics::Joint, 415
- fakeAnchor
 - gazebo::physics::ScrewJoint, 749
 - gazebo::physics::SliderJoint, 888
- far
- gazebo::rendering::GpuLaser, 352
- fdir1
 - gazebo::physics::SurfaceParams, 935
- filename
 - gazebo::PluginT, 648
- FillArrays
 - gazebo::common::Mesh, 522
 - gazebo::common::SubMesh, 921
- FillBufferFromFile
 - gazebo::util::OpenALSource, 614
- FillBufferFromPCM
 - gazebo::util::OpenALSource, 614
- FillHardwareBuffers
 - gazebo::rendering::DynamicRenderable, 291
- FillMsg
 - gazebo::physics::BoxShape, 171
 - gazebo::physics::Collision, 216
 - gazebo::physics::Contact, 254
 - gazebo::physics::CylinderShape, 270
 - gazebo::physics::HeightmapShape, 383
 - gazebo::physics::Joint, 417
 - gazebo::physics::Link, 463
 - gazebo::physics::MapShape, 494
 - gazebo::physics::MeshShape, 535
 - gazebo::physics::Model, 542
 - gazebo::physics::MultiRayShape, 582
 - gazebo::physics::PlaneShape, 644
 - gazebo::physics::RayShape, 702
 - gazebo::physics::Shape, 777
 - gazebo::physics::SphereShape, 900
 - gazebo::physics::SurfaceParams, 934
 - gazebo::rendering::Light, 450
 - gazebo::sensors::Sensor, 755
- FillSDF
 - gazebo::physics::CollisionState, 224
 - gazebo::physics::JointState, 438
 - gazebo::physics::LinkState, 478
 - gazebo::physics::ModelState, 557
 - gazebo::physics::WorldState, 1085
- find_file
 - Common, 37, 38
 - gazebo, 95
- find_file_path
 - Common, 38
- FindFile
 - gazebo::common::SystemPaths, 940
- FindFileURI
 - gazebo::common::SystemPaths, 941
- FindPublication
 - gazebo::transport::TopicManager, 971
- Fini

- Common, 38
- gazebo::Master, 497
- gazebo::physics::Actor, 128
- gazebo::physics::Base, 157
- gazebo::physics::Collision, 216
- gazebo::physics::Entity, 297
- gazebo::physics::Link, 463
- gazebo::physics::Model, 542
- gazebo::physics::PhysicsEngine, 626
- gazebo::physics::SimbodyLink, 818
- gazebo::physics::SimbodyPhysics, 832
- gazebo::physics::World, 1073
- gazebo::rendering::Camera, 188
- gazebo::rendering::DepthCamera, 275
- gazebo::rendering::GpuLaser, 348
- gazebo::rendering::RenderEngine, 708
- gazebo::rendering::RTShaderSystem, 727
- gazebo::rendering::UserCamera, 982
- gazebo::rendering::Visual, 1043
- gazebo::rendering::WindowManager, 1058
- gazebo::sensors::CameraSensor, 208
- gazebo::sensors::ContactSensor, 262
- gazebo::sensors::DepthCameraSensor, 278
- gazebo::sensors::ForceTorqueSensor, 336
- gazebo::sensors::GpsSensor, 342
- gazebo::sensors::GpuRaySensor, 357
- gazebo::sensors::ImuSensor, 396
- gazebo::sensors::MultiCameraSensor, 575
- gazebo::sensors::RaySensor, 695
- gazebo::sensors::RFIDSensor, 711
- gazebo::sensors::RFIDTag, 714
- gazebo::sensors::Sensor, 755
- gazebo::sensors::SensorManager, 765
- gazebo::sensors::SonarSensor, 893
- gazebo::sensors::WirelessReceiver, 1062
- gazebo::sensors::WirelessTransceiver, 1065
- gazebo::Server, 770
- gazebo::transport::ConnectionManager, 249
- gazebo::transport::Node, 589
- gazebo::transport::PublicationTransport, 670
- gazebo::transport::TopicManager, 971
- gazebo::util::LogRecord, 489
- gazebo::util::OpenAL, 611
- fini
 - Classes for physics and dynamics, 67
 - gazebo, 95
 - Rendering, 77
 - Sensors, 82
 - Transport, 87
- Flatten
 - gazebo::rendering::Heightmap, 377
- flipY
 - gazebo::physics::HeightmapShape, 385
- Float
 - gazebo::common::Time, 949
- FogFromSDF
 - Messages, 57
- ForceTorque1
 - Joint_TEST, 431
- ForceTorque2
 - Joint_TEST, 432
- ForceTorqueSensor
 - gazebo::sensors::ForceTorqueSensor, 335
- ForceTorqueSensor.hh, 1143
- ForceTorqueSensorPtr
 - gazebo::sensors, 118
- forces
 - gazebo::physics::SimbodyPhysics, 837
- freq
 - gazebo::sensors::WirelessTransmitter, 1069
- g
 - gazebo::common::Color, 237
- GAUSSIAN
 - gazebo::sensors::Noise, 604
- GAUSSIAN_QUANTIZED
 - gazebo::sensors::Noise, 604
- GAZEBO_DEPRECATED
 - CommonTypes.hh, 1122
- GAZEBO_FORCEINLINE
 - CommonTypes.hh, 1122
- GOURAUD
 - gazebo::common::Material, 501
- GPtrArray
 - MeshCSG.hh, 1185
- GUIFromSDF
 - Messages, 58
- GUIOverlay
 - gazebo::rendering::GUIOverlay, 371
- GUIOverlay.hh, 1151
- GUIPluginPtr
 - gazebo, 95
- GZ_ALL_COLLIDE
 - PhysicsTypes.hh, 1212
- GZ_ASSERT
 - Assert.hh, 1100
- GZ_DBL_MAX
 - Helpers.hh, 1155
- GZ_DBL_MIN
 - Helpers.hh, 1155
- GZ_DTOR
 - Angle.hh, 1096
- GZ_FIXED_COLLIDE
 - PhysicsTypes.hh, 1212
- GZ_FLT_MAX
 - Helpers.hh, 1155
- GZ_FLT_MIN
 - Helpers.hh, 1156

GZ_GHOST_COLLIDE
 PhysicsTypes.hh, 1212
 GZ_LOG_VERSION
 LogRecord.hh, 1176
 GZ_MODEL_DB_MANIFEST_FILENAME
 ModelDatabase.hh, 1192
 GZ_MODEL_MANIFEST_FILENAME
 ModelDatabase.hh, 1192
 GZ_NONE_COLLIDE
 PhysicsTypes.hh, 1212
 GZ_NORMALIZE
 Angle.hh, 1096
 GZ_REGISTER_MODEL_PLUGIN
 Plugin.hh, 1217
 GZ_REGISTER_PHYSICS_ENGINE
 Classes for physics and dynamics, 66
 GZ_REGISTER_SENSOR_PLUGIN
 Plugin.hh, 1218
 GZ_REGISTER_STATIC_MSG
 Messages, 54
 GZ_REGISTER_STATIC_SENSOR
 Sensors, 82
 GZ_REGISTER_SYSTEM_PLUGIN
 Plugin.hh, 1218
 GZ_REGISTER_VISUAL_PLUGIN
 Plugin.hh, 1218
 GZ_REGISTER_WORLD_PLUGIN
 Plugin.hh, 1219
 GZ_RTOD
 Angle.hh, 1096
 GZ_SENSOR_COLLIDE
 PhysicsTypes.hh, 1212
 GZ_SKYX_ALL
 gazebo::rendering::Scene, 732
 GZ_SKYX_CLOUDS
 gazebo::rendering::Scene, 732
 GZ_SKYX_MOON
 gazebo::rendering::Scene, 732
 GZ_SKYX_NONE
 gazebo::rendering::Scene, 732
 GZ_UINT32_MAX
 Helpers.hh, 1156
 GZ_UINT32_MIN
 Helpers.hh, 1156
 GZ_VISIBILITY_ALL
 RenderTypes.hh, 1236
 GZ_VISIBILITY_GUI
 RenderTypes.hh, 1236
 GZ_VISIBILITY_SELECTABLE
 RenderTypes.hh, 1236
 GZ_VISIBILITY_SELECTION
 RenderTypes.hh, 1236
 gain
 gazebo::sensors::WirelessTransceiver, 1066
 gazebo, 93
 add_plugin, 95
 find_file, 95
 fini, 95
 GUIPluginPtr, 95
 init, 95
 load, 95
 ModelPluginPtr, 95
 print_version, 95
 run, 95
 SensorPluginPtr, 95
 stop, 95
 SystemPluginPtr, 95
 VisualPluginPtr, 95
 WorldPluginPtr, 95
 gazebo.hh, 1144
 gazebo::Master, 496
 ~Master, 497
 Fini, 497
 Init, 497
 Master, 497
 Run, 497
 RunOnce, 497
 RunThread, 497
 Stop, 498
 gazebo::ModelPlugin, 553
 ~ModelPlugin, 554
 Init, 554
 Load, 554
 ModelPlugin, 554
 Reset, 554
 gazebo::PluginT
 ~PluginT, 647
 Create, 647
 filename, 648
 GetFilename, 648
 GetHandle, 648
 GetType, 648
 handle, 648
 PluginT, 647
 TPtr, 647
 type, 648
 gazebo::PluginT< T >, 646
 gazebo::SensorPlugin, 767
 ~SensorPlugin, 768
 Init, 769
 Load, 769
 Reset, 769
 SensorPlugin, 768
 gazebo::Server, 769
 ~Server, 770
 Fini, 770
 GetInitialized, 770
 Init, 770

- LoadFile, 770
- LoadString, 770
- ParseArgs, 770
- PreLoad, 770
- PrintUsage, 770
- Run, 770
- Server, 770
- SetParams, 770
- Stop, 770
- systemPluginsArgc, 770
- systemPluginsArgv, 771
- gazebo::SystemPlugin, 942
 - ~SystemPlugin, 943
 - Init, 944
 - Load, 944
 - Reset, 944
 - SystemPlugin, 943
- gazebo::VisualPlugin, 1055
 - Init, 1056
 - Load, 1056
 - Reset, 1056
 - VisualPlugin, 1056
- gazebo::WorldPlugin, 1082
 - ~WorldPlugin, 1082
 - Init, 1083
 - Load, 1083
 - Reset, 1083
 - WorldPlugin, 1082
- gazebo::common, 95
 - AnimationPtr, 98
 - DiagnosticTimerPtr, 98
 - NodeMap, 98
 - NodeMapIter, 98
 - NumericAnimationPtr, 98
 - Param_V, 98
 - PoseAnimationPtr, 98
 - RawNodeAnim, 98
 - RawNodeWeights, 98
 - RawSkeletonAnim, 98
 - SpeedOfLight, 99
 - SphericalCoordinatesPtr, 98
 - StrStr_M, 98
- gazebo::common::Animation, 139
 - ~Animation, 141
 - AddTime, 141
 - Animation, 141
 - build, 143
 - GetKeyFrame, 141
 - GetKeyFrameCount, 141
 - GetKeyFramesAtTime, 142
 - GetLength, 142
 - GetTime, 142
 - KeyFrame_V, 141
 - keyFrames, 143
 - length, 143
 - loop, 143
 - name, 143
 - SetLength, 142
 - SetTime, 142
 - timePos, 143
- gazebo::common::AssertionInternalError, 145
 - ~AssertionInternalError, 147
 - AssertionInternalError, 146
- gazebo::common::AudioDecoder, 147
 - ~AudioDecoder, 147
 - AudioDecoder, 147
 - Decode, 148
 - GetFile, 148
 - GetSampleRate, 148
 - SetFile, 148
- gazebo::common::BVHLoader, 172
 - ~BVHLoader, 173
 - BVHLoader, 173
 - Load, 173
- gazebo::common::ColladaLoader, 211
 - ~ColladaLoader, 212
 - ColladaLoader, 212
 - Load, 212
- gazebo::common::Color, 226
 - ~Color, 229
 - a, 237
 - ABGR, 229
 - ARGB, 229
 - b, 237
 - BGRA, 229
 - Black, 237
 - Blue, 237
 - Color, 229
 - g, 237
 - GetAsABGR, 230
 - GetAsARGB, 230
 - GetAsBGRA, 230
 - GetAsHSV, 230
 - GetAsRGBA, 230
 - GetAsYUV, 230
 - Green, 237
 - operator<<, 236
 - operator>>, 236
 - operator*, 231
 - operator*=:, 231
 - operator+, 232
 - operator+=, 232
 - operator-, 232, 233
 - operator-=, 233
 - operator/, 233
 - operator/=, 234
 - operator=, 234
 - operator==, 234

- operator[], 234
- Purple, 237
- r, 237
- RGBA, 229
- Red, 237
- Reset, 235
- Set, 235
- SetFromABGR, 235
- SetFromARGB, 235
- SetFromBGRA, 235
- SetFromHSV, 235
- SetFromRGBA, 236
- SetFromYUV, 236
- White, 237
- Yellow, 237
- gazebo::common::Console, 252
- gazebo::common::Exception, 331
 - ~Exception, 332
 - Exception, 332
 - GetErrorFile, 332
 - GetErrorStr, 333
 - operator<<, 333
 - Print, 333
- gazebo::common::Image, 389
 - ~Image, 391
 - BAYER_GBRG8, 391
 - BAYER_GRBG8, 391
 - BAYER_RGGB8, 391
 - BAYER_RGGR8, 391
 - BGR_INT16, 391
 - BGR_INT32, 391
 - BGR_INT8, 391
 - BGRA_INT8, 391
 - ConvertPixelFormat, 391
 - GetAvgColor, 392
 - GetBPP, 392
 - GetData, 392
 - GetFilename, 392
 - GetHeight, 392
 - GetMaxColor, 393
 - GetPitch, 393
 - GetPixel, 393
 - GetPixelFormat, 393
 - GetRGBData, 393
 - GetWidth, 393
 - Image, 391
 - L_INT16, 391
 - L_INT8, 391
 - Load, 394
 - PIXEL_FORMAT_COUNT, 391
 - PixelFormat, 391
 - R_FLOAT16, 391
 - R_FLOAT32, 391
 - RGB_FLOAT16, 391
 - RGB_FLOAT32, 391
 - RGB_INT16, 391
 - RGB_INT32, 391
 - RGB_INT8, 391
 - RGBA_INT8, 391
 - Rescale, 394
 - SavePNG, 394
 - SetFromData, 394
 - UNKNOWN_PIXEL_FORMAT, 391
 - Valid, 394
- gazebo::common::InternalError, 408
 - ~InternalError, 409
 - InternalError, 409
- gazebo::common::KeyEvent, 444
 - EventType, 445
 - key, 445
 - KeyEvent, 445
 - NO_EVENT, 445
 - PRESS, 445
 - RELEASE, 445
 - type, 445
- gazebo::common::KeyFrame, 445
 - ~KeyFrame, 446
 - GetTime, 447
 - KeyFrame, 446
 - time, 447
- gazebo::common::Material, 498
 - ~Material, 501
 - ADD, 500
 - ambient, 506
 - BLEND_COUNT, 500
 - BLINN, 501
 - BlendMode, 500
 - blendMode, 506
 - BlendModeStr, 506
 - diffuse, 506
 - emissive, 506
 - FLAT, 501
 - GOURAUD, 501
 - GetAmbient, 501
 - GetBlendFactors, 501
 - GetBlendMode, 501
 - GetDepthWrite, 502
 - GetDiffuse, 502
 - GetEmissive, 502
 - GetLighting, 502
 - GetName, 502
 - GetPointSize, 502
 - GetShadeMode, 503
 - GetShininess, 503
 - GetSpecular, 503
 - GetTextureImage, 503
 - GetTransparency, 503
 - MODULATE, 500

- Material, 501
- name, 506
- operator<<, 506
- PHONG, 501
- pointSize, 506
- REPLACE, 500
- SHADE_COUNT, 501
- SetAmbient, 503
- SetBlendFactors, 504
- SetBlendMode, 504
- SetDepthWrite, 504
- SetDiffuse, 504
- SetEmissive, 504
- SetLighting, 505
- SetPointSize, 505
- SetShadeMode, 505
- SetShininess, 505
- SetSpecular, 505
- SetTextureImage, 505
- SetTransparency, 506
- ShadeMode, 500
- shadeMode, 507
- ShadeModeStr, 507
- shininess, 507
- specular, 507
- texImage, 507
- transparency, 507
- gazebo::common::Mesh, 519
 - ~Mesh, 521
 - AddMaterial, 521
 - AddSubMesh, 521
 - Center, 521
 - FillArrays, 522
 - GenSphericalTexCoord, 522
 - GetAABB, 522
 - GetIndexCount, 522
 - GetMaterial, 522
 - GetMaterialCount, 523
 - GetMax, 523
 - GetMin, 523
 - GetName, 523
 - GetNormalCount, 523
 - GetPath, 523
 - GetSkeleton, 523
 - GetSubMesh, 524
 - GetSubMeshCount, 524
 - GetTexCoordCount, 524
 - GetVertexCount, 524
 - HasSkeleton, 525
 - Mesh, 521
 - RecalculateNormals, 525
 - Scale, 525
 - SetName, 525
 - SetPath, 525
 - SetScale, 525
 - SetSkeleton, 525
 - Translate, 526
- gazebo::common::MeshCSG, 526
 - ~MeshCSG, 527
 - BooleanOperation, 526
 - CreateBoolean, 527
 - DIFFERENCE, 527
 - INTERSECTION, 527
 - MeshCSG, 527
 - UNION, 527
- gazebo::common::MeshLoader, 527
 - ~MeshLoader, 528
 - Load, 528
 - MeshLoader, 528
- gazebo::common::MeshManager, 529
 - AddMesh, 530
 - CreateBox, 530
 - CreateCamera, 530
 - CreateCone, 531
 - CreateCylinder, 531
 - CreatePlane, 531
 - CreateSphere, 532
 - CreateTube, 532
 - GenSphericalTexCoord, 532
 - GetMesh, 532
 - GetMeshAABB, 533
 - HasMesh, 533
 - IsValidFilename, 533
 - Load, 533
- gazebo::common::ModelDatabase, 551
- gazebo::common::MouseEvent, 563
 - alt, 565
 - button, 565
 - Buttons, 564
 - buttons, 565
 - control, 565
 - dragging, 565
 - EventType, 564
 - LEFT, 564
 - MIDDLE, 564
 - MOVE, 564
 - MouseEvent, 565
 - moveScale, 565
 - NO_BUTTON, 564
 - NO_EVENT, 564
 - PRESS, 564
 - pos, 565
 - pressPos, 565
 - prevPos, 565
 - RELEASE, 564
 - RIGHT, 564
 - SCROLL, 564
 - scroll, 565

- shift, 566
- type, 566
- gazebo::common::NodeAnimation, 593
 - ~NodeAnimation, 595
 - AddKeyFrame, 595
 - GetFrameAt, 595
 - GetFrameCount, 595
 - GetKeyFrame, 596
 - GetLength, 596
 - GetName, 596
 - GetTimeAtX, 596
 - keyFrames, 597
 - length, 597
 - name, 597
 - NodeAnimation, 595
 - Scale, 597
 - SetName, 597
- gazebo::common::NodeAssignment, 597
 - nodeIndex, 598
 - vertexIndex, 598
 - weight, 598
- gazebo::common::NodeTransform, 598
 - ~NodeTransform, 600
 - Get, 600
 - GetSID, 600
 - GetType, 601
 - MATRIX, 600
 - NodeTransform, 600
 - operator*, 601
 - operator(), 601
 - PrintSource, 601
 - ROTATE, 600
 - RecalculateMatrix, 602
 - SCALE, 600
 - Set, 602
 - SetComponent, 602
 - SetSID, 602
 - SetSourceValues, 602
 - SetType, 603
 - sid, 603
 - source, 603
 - TRANSLATE, 600
 - transform, 603
 - TransformType, 600
 - type, 603
- gazebo::common::NumericAnimation, 606
 - ~NumericAnimation, 607
 - CreateKeyFrame, 607
 - GetInterpolatedKeyFrame, 607
 - NumericAnimation, 606
- gazebo::common::NumericKeyFrame, 607
 - ~NumericKeyFrame, 608
 - GetValue, 609
 - NumericKeyFrame, 608
 - SetValue, 609
 - value, 609
- gazebo::common::PID, 636
 - ~PID, 637
 - GetCmd, 637
 - GetErrors, 637
 - Init, 637
 - operator=, 638
 - PID, 637
 - Reset, 638
 - SetCmd, 638
 - SetCmdMax, 638
 - SetCmdMin, 638
 - SetDGain, 639
 - SetIGain, 639
 - SetIMax, 639
 - SetIMin, 639
 - SetPGain, 639
 - Update, 639
- gazebo::common::ParamT < T >, 620
- gazebo::common::PoseAnimation, 657
 - ~PoseAnimation, 659
 - BuildInterpolationSplines, 659
 - CreateKeyFrame, 659
 - GetInterpolatedKeyFrame, 659
 - PoseAnimation, 658
- gazebo::common::PoseKeyFrame, 660
 - ~PoseKeyFrame, 661
 - GetRotation, 661
 - GetTranslation, 661
 - PoseKeyFrame, 661
 - rotate, 662
 - SetRotation, 661
 - SetTranslation, 661
 - translate, 662
- gazebo::common::STLLoader, 915
 - ~STLLoader, 916
 - Load, 916
 - STLLoader, 916
- gazebo::common::Skeleton, 866
 - ~Skeleton, 868
 - AddAnimation, 868
 - AddVertNodeWeight, 868
 - anims, 872
 - bindShapeTransform, 872
 - BuildNodeMap, 869
 - GetAnimation, 869
 - GetBindShapeTransform, 869
 - GetNodeByHandle, 869
 - GetNodeById, 869
 - GetNodeByName, 870
 - GetNodes, 870
 - GetNumAnimations, 870
 - GetNumJoints, 870

- GetNumNodes, 870
- GetNumVertNodeWeights, 870
- GetRootNode, 871
- GetVertNodeWeight, 871
- nodes, 872
- PrintTransforms, 871
- rawNW, 872
- root, 872
- Scale, 871
- SetBindShapeTransform, 871
- SetNumVertAttached, 872
- SetRootNode, 872
- Skeleton, 868
- gazebo::common::SkeletonAnimation, 873
 - ~SkeletonAnimation, 874
 - AddKeyFrame, 874
 - animations, 876
 - GetLength, 874
 - GetName, 875
 - GetNodeCount, 875
 - GetNodePoseAt, 875
 - GetPoseAt, 875
 - GetPoseAtX, 876
 - HasNode, 876
 - length, 876
 - name, 877
 - Scale, 876
 - SetName, 876
 - SkeletonAnimation, 874
- gazebo::common::SkeletonNode, 877
 - ~SkeletonNode, 880
 - AddChild, 880
 - AddRawTransform, 880
 - children, 885
 - GetChild, 880
 - GetChildById, 880
 - GetChildByName, 881
 - GetChildCount, 881
 - GetHandle, 881
 - GetId, 881
 - GetInverseBindTransform, 881
 - GetModelTransform, 881
 - GetName, 882
 - GetNumRawTrans, 882
 - GetParent, 882
 - GetRawTransform, 882
 - GetRawTransforms, 882
 - GetTransform, 883
 - GetTransforms, 883
 - handle, 885
 - id, 885
 - initialTransform, 885
 - invBindTransform, 885
 - IsJoint, 883
 - IsRootNode, 883
 - JOINT, 879
 - modelTransform, 885
 - NODE, 879
 - name, 885
 - parent, 886
 - rawTransforms, 886
 - Reset, 883
 - SetHandle, 883
 - SetId, 883
 - SetInitialTransform, 884
 - SetInverseBindTransform, 884
 - SetModelTransform, 884
 - SetName, 884
 - SetParent, 884
 - SetTransform, 884
 - SetType, 885
 - SkeletonNode, 879, 880
 - SkeletonNodeType, 879
 - transform, 886
 - type, 886
 - UpdateChildrenTransforms, 885
- gazebo::common::SphericalCoordinates, 901
 - ~SphericalCoordinates, 903
 - Convert, 903
 - EARTH_WGS84, 902
 - GetElevationReference, 903
 - GetHeadingOffset, 904
 - GetLatitudeReference, 904
 - GetLongitudeReference, 904
 - GetSurfaceType, 904
 - GlobalFromLocal, 904
 - SetElevationReference, 905
 - SetHeadingOffset, 905
 - SetLatitudeReference, 905
 - SetLongitudeReference, 905
 - SetSurfaceType, 905
 - SphericalCoordinates, 903
 - SphericalFromLocal, 905
 - SurfaceType, 902
- gazebo::common::SubMesh, 916
 - ~SubMesh, 919
 - AddIndex, 919
 - AddNodeAssignment, 919
 - AddNormal, 920
 - AddTexCoord, 920
 - AddVertex, 920
 - Center, 921
 - CopyNormals, 921
 - CopyVertices, 921
 - FillArrays, 921
 - GenSphericalTexCoord, 921
 - GetIndex, 921
 - GetIndexCount, 922

- GetMaterialIndex, 922
- GetMax, 922
- GetMaxIndex, 922
- GetMin, 922
- GetName, 922
- GetNodeAssignment, 922
- GetNodeAssignmentsCount, 923
- GetNormal, 923
- GetNormalCount, 923
- GetPrimitiveType, 923
- GetTexCoord, 923
- GetTexCoordCount, 923
- GetVertex, 924
- GetVertexCount, 924
- GetVertexIndex, 924
- HasVertex, 924
- LINES, 919
- LINESTRIPS, 919
- POINTS, 919
- PrimitiveType, 919
- RecalculateNormals, 924
- Scale, 924
- SetIndexCount, 924
- SetMaterialIndex, 925
- SetName, 925
- SetNormal, 925
- SetNormalCount, 925
- SetPrimitiveType, 925
- SetScale, 926
- SetSubMeshCenter, 926
- SetTexCoord, 926
- SetTexCoordCount, 926
- SetVertex, 926
- SetVertexCount, 926
- SubMesh, 919
- TRIANGLES, 919
- TRIFANS, 919
- TRISTRIPS, 919
- Translate, 927
- gazebo::common::SystemPaths, 937
 - AddGazeboPaths, 939
 - AddModelPaths, 939
 - AddOgrePaths, 939
 - AddPluginPaths, 940
 - AddSearchPathSuffix, 940
 - ClearGazeboPaths, 940
 - ClearModelPaths, 940
 - ClearOgrePaths, 940
 - ClearPluginPaths, 940
 - FindFile, 940
 - FindFileURI, 941
 - gazeboPathsFromEnv, 942
 - GetGazeboPaths, 941
 - GetLogPath, 941
 - GetModelPaths, 941
 - GetOgrePaths, 941
 - GetPluginPaths, 941
 - GetWorldPathExtension, 942
 - modelPathsFromEnv, 942
 - ogrePathsFromEnv, 942
 - pluginPathsFromEnv, 942
- gazebo::common::Time, 944
 - ~Time, 949
 - Double, 949
 - Float, 949
 - GetWallTime, 949
 - GetWallTimeAsISOString, 949
 - MSleep, 950
 - MicToNano, 950
 - MilToNano, 950
 - NSleep, 950
 - nsec, 965
 - operator<, 958
 - operator<<, 964
 - operator<=, 959
 - operator>, 961, 962
 - operator>>, 965
 - operator>=, 962, 963
 - operator*, 952
 - operator*=, 952, 953
 - operator+, 953, 954
 - operator+=, 954
 - operator-, 955
 - operator-=, 955, 956
 - operator/, 956, 957
 - operator/=, 957
 - operator=, 960
 - operator==, 960, 961
 - sec, 965
 - SecToNano, 963
 - Set, 964
 - SetToWallTime, 964
 - Sleep, 964
 - Time, 948, 949
 - Zero, 965
- gazebo::common::Timer, 965
 - ~Timer, 966
 - GetElapsed, 967
 - GetRunning, 967
 - operator<<, 967
 - Start, 967
 - Stop, 967
 - Timer, 966
- gazebo::common::UpdateInfo, 978
 - realTime, 978
 - simTime, 978
 - worldName, 979
- gazebo::common::Video, 1028

- ~Video, 1028
- GetHeight, 1028
- GetNextFrame, 1028
- GetWidth, 1029
- Load, 1029
- Video, 1028
- gazebo::event, 99
 - Connection_V, 99
 - ConnectionPtr, 99
- gazebo::event::Connection, 239
 - ~Connection, 240
 - Connection, 240
 - GetId, 240
- gazebo::event::Event, 305
 - ~Event, 307
 - Disconnect, 307
- gazebo::event::EventT
 - operator(), 325–327
 - Signal, 328–330
- gazebo::event::EventT< T >, 322
- gazebo::event::Events, 310
 - addEntity, 320
 - ConnectAddEntity, 313
 - ConnectCreateEntity, 313
 - ConnectDeleteEntity, 313
 - ConnectDiagTimerStart, 313
 - ConnectDiagTimerStop, 314
 - ConnectPause, 314
 - ConnectPostRender, 314
 - ConnectPreRender, 315
 - ConnectRender, 315
 - ConnectSetSelectedEntity, 315
 - ConnectSigInt, 315
 - ConnectStep, 316
 - ConnectStop, 316
 - ConnectWorldCreated, 316
 - ConnectWorldUpdateBegin, 317
 - ConnectWorldUpdateEnd, 317
 - deleteEntity, 320
 - diagTimerStart, 321
 - diagTimerStop, 321
 - DisconnectAddEntity, 317
 - DisconnectCreateEntity, 317
 - DisconnectDeleteEntity, 318
 - DisconnectDiagTimerStart, 318
 - DisconnectDiagTimerStop, 318
 - DisconnectPause, 318
 - DisconnectPostRender, 318
 - DisconnectPreRender, 319
 - DisconnectRender, 319
 - DisconnectSetSelectedEntity, 319
 - DisconnectSigInt, 319
 - DisconnectStep, 319
 - DisconnectStop, 320
 - DisconnectWorldCreated, 320
 - DisconnectWorldUpdateBegin, 320
 - DisconnectWorldUpdateEnd, 320
 - entityCreated, 321
 - pause, 321
 - postRender, 321
 - preRender, 321
 - render, 321
 - setSelectedEntity, 321
 - sigInt, 321
 - step, 322
 - stop, 322
 - worldCreated, 322
 - worldUpdateBegin, 322
 - worldUpdateEnd, 322
- gazebo::math, 99
 - GeneratorType, 102
 - NRealGen, 102
 - NormalRealDist, 102
 - UIntGen, 102
 - URealGen, 102
 - UniformIntDist, 102
 - UniformRealDist, 102
- gazebo::math::Angle, 131
 - ~Angle, 133
 - Angle, 133
 - Degree, 134
 - HalfPi, 139
 - Normalize, 134
 - operator<, 136
 - operator<<, 138
 - operator<=, 136
 - operator>, 137
 - operator>>, 138
 - operator>=, 137
 - operator*, 134
 - operator*=:, 134
 - operator+, 135
 - operator+=, 135
 - operator-, 135
 - operator-=, 135
 - operator/, 136
 - operator/=, 136
 - operator==, 137
 - Pi, 139
 - Radian, 137
 - SetFromDegree, 138
 - SetFromRadian, 138
 - TwoPi, 139
 - Zero, 139
- gazebo::math::Box, 164
 - ~Box, 166
 - Box, 165, 166
 - GetCenter, 166

- GetSize, 166
- GetXLength, 166
- GetYLength, 166
- GetZLength, 167
- max, 169
- Merge, 167
- min, 169
- operator<<, 168
- operator+, 167
- operator+=", 167
- operator-, 168
- operator=, 168
- operator==, 168
- gazebo::math::Matrix3, 507
 - ~Matrix3, 509
 - m, 512
 - Matrix3, 508, 509
 - operator<<, 511
 - operator*, 509, 511
 - operator+, 509
 - operator-, 510
 - operator==, 510
 - operator[], 510
 - SetCol, 510
 - SetFromAxes, 511
 - SetFromAxis, 511
- gazebo::math::Matrix4, 512
 - ~Matrix4, 514
 - GetAsPose, 514
 - GetEulerRotation, 515
 - GetRotation, 515
 - GetTranslation, 515
 - IDENTITY, 519
 - Inverse, 515
 - IsAffine, 515
 - m, 519
 - Matrix4, 514
 - operator<<, 518
 - operator*, 515, 516
 - operator=, 516
 - operator==, 517
 - operator[], 517
 - Set, 517
 - SetScale, 518
 - SetTranslate, 518
 - TransformAffine, 518
 - ZERO, 519
- gazebo::math::Plane, 640
 - ~Plane, 641
 - d, 642
 - Distance, 641
 - normal, 642
 - operator=, 641
 - Plane, 641
 - Set, 642
 - size, 642
- gazebo::math::Pose, 648
 - ~Pose, 651
 - CoordPoseSolve, 651
 - CoordPositionAdd, 651, 652
 - CoordPositionSub, 652
 - CoordRotationAdd, 652
 - CoordRotationSub, 652
 - Correct, 653
 - GetInverse, 653
 - IsFinite, 653
 - operator<<, 656
 - operator>>, 657
 - operator*, 653
 - operator+, 654
 - operator+=", 654
 - operator-, 654
 - operator-=, 655
 - operator=, 655
 - operator==, 655
 - pos, 657
 - Pose, 650, 651
 - Reset, 655
 - rot, 657
 - RotatePositionAboutOrigin, 655
 - Round, 656
 - Set, 656
 - Zero, 657
- gazebo::math::Quaternion, 675
 - ~Quaternion, 679
 - Correct, 679
 - Dot, 679
 - EulerToQuaternion, 679
 - GetAsAxis, 679
 - GetAsEuler, 680
 - GetAsMatrix3, 680
 - GetAsMatrix4, 680
 - GetExp, 680
 - GetInverse, 680
 - GetLog, 680
 - GetPitch, 681
 - GetRoll, 681
 - GetXAxis, 681
 - GetYAxis, 681
 - GetYaw, 681
 - GetZAxis, 681
 - Invert, 682
 - IsFinite, 682
 - Normalize, 682
 - operator<<, 687
 - operator>>, 688
 - operator*, 682, 683
 - operator*=", 683

- operator+, 683
- operator+=", 683
- operator-, 684
- operator-=, 684
- operator=, 684
- operator==, 685
- Quaternion, 678
- RotateVector, 685
- RotateVectorReverse, 685
- Round, 685
- Scale, 685
- Set, 686
- SetFromAxis, 686
- SetFromEuler, 686
- SetToIdentity, 687
- Slerp, 687
- Squad, 687
- w, 688
- x, 688
- y, 688
- z, 688
- gazebo::math::Rand, 688
 - GetDbfNormal, 689
 - GetDbfUniform, 689
 - GetIntNormal, 689
 - GetIntUniform, 689
 - GetSeed, 690
 - SetSeed, 690
- gazebo::math::RotationSpline, 720
 - ~RotationSpline, 721
 - AddPoint, 722
 - autoCalc, 724
 - Clear, 722
 - GetNumPoints, 722
 - GetPoint, 722
 - Interpolate, 722, 723
 - points, 724
 - RecalcTangents, 723
 - RotationSpline, 721
 - SetAutoCalculate, 723
 - tangents, 724
 - UpdatePoint, 723
- gazebo::math::Spline, 906
 - ~Spline, 907
 - AddPoint, 907
 - autoCalc, 910
 - Clear, 907
 - coeffs, 910
 - GetPoint, 907
 - GetPointCount, 908
 - GetTangent, 908
 - GetTension, 908
 - Interpolate, 908
 - points, 910
 - RecalcTangents, 908
 - SetAutoCalculate, 909
 - SetTension, 909
 - Spline, 907
 - tangents, 910
 - tension, 910
 - UpdatePoint, 909
- gazebo::math::Vector2d, 987
 - ~Vector2d, 989
 - Cross, 990
 - Distance, 990
 - IsFinite, 990
 - Normalize, 990
 - operator<<, 995
 - operator>>, 995
 - operator*, 990, 991
 - operator*=", 991
 - operator+, 992
 - operator+=", 992
 - operator-, 992
 - operator-=, 992
 - operator/, 992, 993
 - operator/=, 993
 - operator=, 994
 - operator==, 994
 - operator[], 994
 - Set, 995
 - Vector2d, 989
 - x, 995
 - y, 995
- gazebo::math::Vector2i, 996
 - ~Vector2i, 998
 - Cross, 998
 - Distance, 998
 - IsFinite, 998
 - Normalize, 999
 - operator<<, 1004
 - operator>>, 1004
 - operator*, 999
 - operator*=", 1000
 - operator+, 1000
 - operator+=", 1000
 - operator-, 1001
 - operator-=, 1001
 - operator/, 1001
 - operator/=, 1002
 - operator=, 1002, 1003
 - operator==, 1003
 - operator[], 1003
 - Set, 1003
 - Vector2i, 997, 998
 - x, 1004
 - y, 1004
- gazebo::math::Vector3, 1004

~Vector3, 1008
 Correct, 1008
 Cross, 1008
 Distance, 1008, 1009
 Dot, 1009
 Equal, 1009
 GetAbs, 1009
 GetDistToLine, 1009
 GetLength, 1010
 GetMax, 1010
 GetMin, 1010
 GetNormal, 1010
 GetPerpendicular, 1010
 GetRounded, 1011
 GetSquaredLength, 1011
 GetSum, 1011
 IsFinite, 1011
 Normalize, 1011
 One, 1017
 operator<<, 1017
 operator>>, 1017
 operator*, 1012, 1017
 operator*=: 1012
 operator+, 1013
 operator+=, 1013
 operator-, 1013
 operator-=, 1014
 operator/, 1014
 operator/=, 1014
 operator=, 1015
 operator==, 1015
 operator[], 1015
 Round, 1016
 Set, 1016
 SetToMax, 1016
 SetToMin, 1016
 UnitX, 1017
 UnitY, 1017
 UnitZ, 1018
 Vector3, 1007, 1008
 x, 1018
 y, 1018
 z, 1018
 Zero, 1018
 gazebo::math::Vector4, 1018
 ~Vector4, 1021
 Distance, 1021
 GetLength, 1021
 GetSquaredLength, 1021
 IsFinite, 1021
 Normalize, 1021
 operator<<, 1026
 operator>>, 1027
 operator*, 1022
 operator*=: 1023
 operator+, 1023
 operator+=, 1023
 operator-, 1024
 operator-=, 1024
 operator/, 1024
 operator/=, 1025
 operator=, 1025, 1026
 operator==, 1026
 operator[], 1026
 Set, 1026
 Vector4, 1020
 w, 1027
 x, 1027
 y, 1027
 z, 1027
 gazebo::msgs, 102
 MsgFactoryFn, 104
 gazebo::msgs::MsgFactory, 572
 GetMsgTypes, 573
 NewMsg, 573
 RegisterMsg, 573
 gazebo::physics, 104
 Actor_V, 109
 ActorPtr, 109
 Base_V, 109
 BasePtr, 109
 BoxShapePtr, 109
 Collision_V, 109
 CollisionPtr, 109
 ContactPtr, 109
 CylinderShapePtr, 109
 EntityPtr, 109
 GripperPtr, 109
 HeightmapShapePtr, 109
 InertialPtr, 109
 Joint_V, 109
 JointController_V, 110
 JointControllerPtr, 110
 JointPtr, 110
 JointState_M, 110
 Link_V, 110
 LinkPtr, 110
 LinkState_M, 110
 MeshShapePtr, 110
 Model_V, 110
 ModelPtr, 110
 ModelState_M, 110
 MultiRayShapePtr, 110
 PhysicsEnginePtr, 110
 RayShapePtr, 110
 RoadPtr, 110
 ShapePtr, 110
 SimbodyCollisionPtr, 110

- SimbodyLinkPtr, 110
- SimbodyModelPtr, 110
- SimbodyPhysicsPtr, 110
- SimbodyRayShapePtr, 110
- SphereShapePtr, 110
- SurfaceParamsPtr, 110
- WorldPtr, 110
- gazebo::physics::Actor, 125
 - ~Actor, 128
 - active, 129
 - Actor, 127
 - autoStart, 129
 - bonePosePub, 129
 - Fini, 128
 - GetSDF, 128
 - Init, 128
 - interpolateX, 129
 - isActive, 128
 - lastPos, 129
 - lastScriptTime, 129
 - lastTraj, 129
 - Load, 128
 - loop, 130
 - mainLink, 130
 - mesh, 130
 - oldAction, 130
 - pathLength, 130
 - Play, 128
 - playStartTime, 130
 - prevFrameTime, 130
 - scriptLength, 130
 - skelAnimation, 130
 - skelNodesMap, 130
 - skeleton, 130
 - skinFile, 131
 - skinScale, 131
 - startDelay, 131
 - Stop, 128
 - trajInfo, 131
 - trajectories, 131
 - Update, 129
 - UpdateParameters, 129
 - visualId, 131
 - visualName, 131
- gazebo::physics::BallJoint
 - ~BallJoint, 152
 - BallJoint, 152
 - GetAngleCount, 152
 - GetHighStop, 152
 - GetLowStop, 152
 - Load, 152
 - SetAxis, 153
 - SetHighStop, 153
 - SetLowStop, 153
- gazebo::physics::BallJoint< T >, 151
- gazebo::physics::Base, 153
 - ~Base, 157
 - ACTOR, 156
 - AddChild, 157
 - AddType, 157
 - BALL_JOINT, 156
 - BASE, 156
 - BOX_SHAPE, 156
 - Base, 157
 - COLLISION, 156
 - CYLINDER_SHAPE, 156
 - children, 164
 - childrenEnd, 164
 - ComputeScopedName, 157
 - ENTITY, 156
 - EntityType, 156
 - Fini, 157
 - GetByName, 158
 - GetChild, 158
 - GetChildCount, 158
 - GetId, 158
 - GetName, 159
 - GetParent, 159
 - GetParentId, 159
 - GetSDF, 159
 - GetSaveable, 159
 - GetScopedName, 159
 - GetType, 160
 - GetWorld, 160
 - HEIGHTMAP_SHAPE, 156
 - HINGE2_JOINT, 156
 - HINGE_JOINT, 156
 - HasType, 160
 - Init, 160
 - isSelected, 161
 - JOINT, 156
 - LIGHT, 156
 - LINK, 156
 - Load, 161
 - MAP_SHAPE, 156
 - MESH_SHAPE, 157
 - MODEL, 156
 - MULTIRAY_SHAPE, 156
 - operator==, 161
 - PLANE_SHAPE, 157
 - parent, 164
 - Print, 161
 - RAY_SHAPE, 157
 - RemoveChild, 162
 - RemoveChildren, 162
 - Reset, 162
 - SCREW_JOINT, 156
 - SENSOR_COLLISION, 157

- SHAPE, 156
- SLIDER_JOINT, 156
- SPHERE_SHAPE, 157
- sdf, 164
- SetName, 162
- SetParent, 162
- SetSaveable, 163
- SetSelected, 163
- SetWorld, 163
- UNIVERSAL_JOINT, 156
- Update, 163
- UpdateParameters, 163
- VISUAL, 156
- world, 164
- gazebo::physics::BoxShape, 169
 - ~BoxShape, 171
 - BoxShape, 171
 - FillMsg, 171
 - GetSize, 171
 - Init, 171
 - ProcessMsg, 171
 - SetScale, 172
 - SetSize, 172
- gazebo::physics::Collision, 213
 - ~Collision, 216
 - AddContact, 216
 - Collision, 216
 - FillMsg, 216
 - Fini, 216
 - GetBoundingBox, 216
 - GetContactsEnabled, 216
 - GetLaserRetro, 217
 - GetLink, 217
 - GetMaxContacts, 217
 - GetModel, 217
 - GetRelativeAngularAccel, 217
 - GetRelativeAngularVel, 217
 - GetRelativeLinearAccel, 218
 - GetRelativeLinearVel, 218
 - GetShape, 218
 - GetShapeType, 218
 - GetState, 218
 - GetSurface, 219
 - GetWorldAngularAccel, 219
 - GetWorldAngularVel, 219
 - GetWorldLinearAccel, 219
 - GetWorldLinearVel, 219
 - Init, 219
 - IsPlaceable, 220
 - link, 222
 - Load, 220
 - placeable, 222
 - ProcessMsg, 220
 - SetCategoryBits, 220
 - SetCollideBits, 220
 - SetCollision, 221
 - SetContactsEnabled, 221
 - SetLaserRetro, 221
 - SetMaxContacts, 221
 - SetScale, 221
 - SetShape, 221
 - SetState, 222
 - shape, 222
 - UpdateParameters, 222
- gazebo::physics::CollisionState, 222
 - ~CollisionState, 224
 - CollisionState, 224
 - FillSDF, 224
 - GetPose, 224
 - IsZero, 225
 - Load, 225
 - operator<<, 226
 - operator+, 225
 - operator-, 225
 - operator=, 226
- gazebo::physics::Contact, 253
 - ~Contact, 254
 - collision1, 255
 - collision2, 255
 - Contact, 254
 - count, 255
 - DebugString, 254
 - depths, 255
 - FillMsg, 254
 - normals, 256
 - operator=, 255
 - positions, 256
 - Reset, 255
 - time, 256
 - world, 256
 - wrench, 256
- gazebo::physics::ContactManager, 256
 - ~ContactManager, 257
 - Clear, 257
 - ContactManager, 257
 - CreateFilter, 257, 258
 - GetContact, 258
 - GetContactCount, 258
 - GetContacts, 258
 - Init, 258
 - NewContact, 259
 - PublishContacts, 259
 - ResetCount, 259
- gazebo::physics::ContactPublisher, 259
 - collisionNames, 260
 - collisions, 260
 - contacts, 260
 - publisher, 260

- gazebo::physics::CylinderShape, 268
 - ~CylinderShape, 270
 - CylinderShape, 270
 - FillMsg, 270
 - GetLength, 270
 - GetRadius, 270
 - Init, 271
 - ProcessMsg, 271
 - SetLength, 271
 - SetRadius, 271
 - SetScale, 271
 - SetSize, 271
- gazebo::physics::Entity, 293
 - ~Entity, 296
 - animation, 304
 - animationConnection, 304
 - animationStartPose, 304
 - connections, 304
 - dirtyPose, 304
 - Entity, 296
 - Fini, 297
 - GetBoundingBox, 297
 - GetChildCollision, 297
 - GetChildLink, 297
 - GetCollisionBoundingBox, 297
 - GetDirtyPose, 298
 - GetInitialRelativePose, 298
 - GetNearestEntityBelow, 298
 - GetParentModel, 298
 - GetRelativeAngularAccel, 298
 - GetRelativeAngularVel, 298
 - GetRelativeLinearAccel, 299
 - GetRelativeLinearVel, 299
 - GetRelativePose, 299
 - GetWorldAngularAccel, 299
 - GetWorldAngularVel, 299
 - GetWorldLinearAccel, 300
 - GetWorldLinearVel, 300
 - GetWorldPose, 300
 - IsCanonicalLink, 300
 - IsStatic, 300
 - Load, 301
 - node, 304
 - OnPoseChange, 301
 - parentEntity, 304
 - PlaceOnEntity, 301
 - PlaceOnNearestEntityBelow, 301
 - prevAnimationTime, 304
 - requestPub, 304
 - Reset, 301
 - scale, 304
 - SetAnimation, 301, 302
 - SetCanonicalLink, 302
 - SetInitialRelativePose, 302
 - SetName, 302
 - SetRelativePose, 302
 - SetStatic, 303
 - SetWorldPose, 303
 - SetWorldTwist, 303
 - StopAnimation, 303
 - UpdateParameters, 303
 - visPub, 304
 - visualMsg, 305
- gazebo::physics::Gripper, 368
 - ~Gripper, 369
 - GetName, 369
 - Gripper, 369
 - Init, 369
 - IsAttached, 370
 - Load, 370
 - node, 370
- gazebo::physics::HeightmapShape, 380
 - ~HeightmapShape, 383
 - FillMsg, 383
 - flipY, 385
 - GetHeight, 383
 - GetImage, 383
 - GetMaxHeight, 383
 - GetMinHeight, 383
 - GetPos, 384
 - GetSize, 384
 - GetSubSampling, 384
 - GetURI, 384
 - GetVertexCount, 384
 - HeightmapShape, 382
 - heights, 385
 - img, 385
 - Init, 384
 - Load, 384
 - ProcessMsg, 385
 - SetScale, 385
 - subSampling, 385
 - vertSize, 385
- gazebo::physics::Hinge2Joint
 - ~Hinge2Joint, 387
 - GetAngleCount, 387
 - Hinge2Joint, 387
 - Load, 387
- gazebo::physics::Hinge2Joint< T >, 386
- gazebo::physics::HingeJoint
 - ~HingeJoint, 388
 - GetAngleCount, 389
 - HingeJoint, 388
 - Init, 389
 - Load, 389
- gazebo::physics::HingeJoint< T >, 387
- gazebo::physics::Inertial, 398
 - ~Inertial, 401

- GetCoG, 401
- GetlXX, 401
- GetlXY, 402
- GetlXZ, 402
- GetlYY, 402
- GetlYZ, 402
- GetlZZ, 402
- GetInertial, 401
- GetMOI, 402, 403
- GetMass, 402
- GetPose, 403
- GetPrincipalMoments, 403
- GetProductsofInertia, 403
- Inertial, 400, 401
- Load, 403
- operator<<, 407
- operator+, 404
- operator+=", 404
- operator=, 404
- ProcessMsg, 404
- Reset, 405
- Rotate, 405
- SetCoG, 405
- SetlXX, 406
- SetlXY, 406
- SetlXZ, 406
- SetlYY, 406
- SetlYZ, 407
- SetlZZ, 407
- SetInertiaMatrix, 406
- SetMOI, 407
- SetMass, 407
- UpdateParameters, 407
- gazebo::physics::Joint, 411
 - ~Joint, 415
 - anchorLink, 428
 - anchorPos, 428
 - anchorPose, 429
 - ApplyDamping, 415
 - applyDamping, 429
 - AreConnected, 415
 - Attach, 416
 - Attribute, 415
 - CFM, 415
 - CacheForceTorque, 416
 - CheckAndTruncateForce, 416
 - childLink, 429
 - ConnectJointUpdate, 416
 - dampingCoefficient, 429
 - Detach, 416
 - DisconnectJointUpdate, 417
 - ERP, 415
 - effortLimit, 429
 - FMAX, 415
 - FUDGE_FACTOR, 415
 - FillMsg, 417
 - GetAnchor, 417
 - GetAngle, 417
 - GetAngleCount, 417
 - GetAngleImpl, 418
 - GetAttribute, 418
 - GetChild, 418
 - GetDamping, 418
 - GetDampingCoefficient, 419
 - GetEffortLimit, 419
 - GetForce, 419
 - GetForceTorque, 419
 - GetGlobalAxis, 420
 - GetHighStop, 420
 - GetInertiaRatio, 421
 - GetJointLink, 421
 - GetLinkForce, 421
 - GetLinkTorque, 421
 - GetLocalAxis, 422
 - GetLowStop, 422
 - GetLowerLimit, 422
 - GetMaxForce, 423
 - GetParent, 423
 - GetUpperLimit, 423
 - GetVelocity, 423
 - GetVelocityLimit, 424
 - HI_STOP, 415
 - inertiaRatio, 429
 - Init, 424
 - Joint, 415
 - LO_STOP, 415
 - Load, 424
 - lowerLimit, 429
 - model, 429
 - parentLink, 429
 - provideFeedback, 429
 - Reset, 425
 - STOP_CFM, 415
 - STOP_ERP, 415
 - SUSPENSION_CFM, 415
 - SUSPENSION_ERP, 415
 - SetAnchor, 425
 - SetAngle, 425
 - SetAttribute, 425
 - SetAxis, 426
 - SetDamping, 426
 - SetForce, 426
 - SetHighStop, 426
 - SetLowStop, 427
 - SetMaxForce, 427
 - SetModel, 427
 - SetProvideFeedback, 427
 - SetState, 428

- SetVelocity, 428
- Update, 428
- UpdateParameters, 428
- upperLimit, 429
- useCFMDamping, 430
- VEL, 415
- velocityLimit, 430
- wrench, 430
- gazebo::physics::JointController, 434
 - AddJoint, 435
 - JointController, 435
 - Reset, 435
 - SetJointPosition, 435
 - SetJointPositions, 435
 - Update, 436
- gazebo::physics::JointState, 436
 - ~JointState, 438
 - FillSDF, 438
 - GetAngle, 438
 - GetAngleCount, 438
 - GetAngles, 439
 - IsZero, 439
 - JointState, 437, 438
 - Load, 439
 - operator<<, 440
 - operator+, 439
 - operator-, 440
 - operator=, 440
- gazebo::physics::JointWrench, 442
 - body1Force, 443
 - body1Torque, 444
 - body2Force, 444
 - body2Torque, 444
 - operator+, 443
 - operator-, 443
 - operator=, 443
- gazebo::physics::Link, 455
 - ~Link, 460
 - AddChildJoint, 460
 - AddForce, 460
 - AddForceAtRelativePosition, 460
 - AddForceAtWorldPosition, 461
 - AddParentJoint, 461
 - AddRelativeForce, 461
 - AddRelativeTorque, 461
 - AddTorque, 461
 - angularAccel, 475
 - AttachStaticModel, 462
 - attachedModelsOffset, 475
 - cgVisuals, 475
 - ConnectEnabled, 462
 - DetachAllStaticModels, 462
 - DetachStaticModel, 462
 - DisconnectEnabled, 462
 - FillMsg, 463
 - Fini, 463
 - GetAngularDamping, 463
 - GetBoundingBox, 463
 - GetChildJoints, 463
 - GetChildJointsLinks, 463
 - GetCollision, 463, 464
 - GetCollisions, 464
 - GetEnabled, 464
 - GetGravityMode, 464
 - GetInertial, 464
 - GetKinematic, 465
 - GetLinearDamping, 465
 - GetModel, 465
 - GetParentJoints, 465
 - GetParentJointsLinks, 465
 - GetRelativeAngularAccel, 465
 - GetRelativeAngularVel, 466
 - GetRelativeForce, 466
 - GetRelativeLinearAccel, 466
 - GetRelativeLinearVel, 466
 - GetRelativeTorque, 466
 - GetSelfCollide, 467
 - GetSensorCount, 467
 - GetSensorName, 467
 - GetWorldAngularAccel, 467
 - GetWorldCoGLinearVel, 468
 - GetWorldCoGPose, 468
 - GetWorldForce, 468
 - GetWorldLinearAccel, 468
 - GetWorldLinearVel, 468, 469
 - GetWorldTorque, 469
 - inertial, 475
 - Init, 469
 - linearAccel, 475
 - Link, 460
 - Load, 469
 - OnPoseChange, 470
 - ProcessMsg, 470
 - RemoveChild, 470
 - RemoveChildJoint, 470
 - RemoveCollision, 470
 - RemoveParentJoint, 470
 - Reset, 470
 - ResetPhysicsStates, 471
 - SetAngularAccel, 471
 - SetAngularDamping, 471
 - SetAngularVel, 471
 - SetAutoDisable, 471
 - SetCollideMode, 471
 - SetEnabled, 472
 - SetForce, 472
 - SetGravityMode, 472
 - SetInertial, 472

- SetKinematic, 472
- SetLaserRetro, 473
- SetLinearAccel, 473
- SetLinearDamping, 473
- SetLinearVel, 473
- SetLinkStatic, 473
- SetPublishData, 473
- SetScale, 474
- SetSelected, 474
- SetSelfCollide, 474
- SetState, 474
- SetTorque, 474
- Update, 474
- UpdateMass, 475
- UpdateParameters, 475
- UpdateSurface, 475
- visuals, 476
- Visuals_M, 460
- gazebo::physics::LinkState, 476
 - ~LinkState, 478
 - FillSDF, 478
 - GetAcceleration, 479
 - GetCollisionState, 479
 - GetCollisionStateCount, 479
 - GetCollisionStates, 480
 - GetPose, 480
 - GetVelocity, 480
 - GetWrench, 480
 - IsZero, 480
 - LinkState, 478
 - Load, 480, 481
 - operator<<, 482
 - operator+, 481
 - operator-, 481
 - operator=, 482
 - SetRealTime, 482
 - SetSimTime, 482
 - SetWallTime, 482
- gazebo::physics::MapShape, 492
 - ~MapShape, 494
 - FillMsg, 494
 - GetGranularity, 494
 - GetHeight, 494
 - GetScale, 495
 - GetThreshold, 495
 - GetURI, 495
 - Init, 495
 - Load, 495
 - MapShape, 494
 - ProcessMsg, 496
 - SetScale, 496
 - Update, 496
- gazebo::physics::MeshShape, 534
 - ~MeshShape, 535
 - FillMsg, 535
 - GetMeshURI, 536
 - GetSize, 536
 - Init, 536
 - mesh, 537
 - MeshShape, 535
 - ProcessMsg, 536
 - SetMesh, 536
 - SetScale, 536
 - submesh, 537
 - Update, 537
- gazebo::physics::Model, 537
 - ~Model, 541
 - AttachStaticModel, 541
 - attachedModels, 551
 - attachedModelsOffset, 551
 - DetachStaticModel, 542
 - FillMsg, 542
 - Fini, 542
 - GetAutoDisable, 542
 - GetBoundingBox, 542
 - GetGripper, 542
 - GetGripperCount, 543
 - GetJoint, 543
 - GetJointController, 543
 - GetJointCount, 543
 - GetJoints, 543
 - GetLink, 544
 - GetLinks, 544
 - GetPluginCount, 544
 - GetRelativeAngularAccel, 544
 - GetRelativeAngularVel, 544
 - GetRelativeLinearAccel, 545
 - GetRelativeLinearVel, 545
 - GetSDF, 545
 - GetSensorCount, 545
 - GetWorldAngularAccel, 545
 - GetWorldAngularVel, 546
 - GetWorldLinearAccel, 546
 - GetWorldLinearVel, 546
 - Init, 546
 - jointPub, 551
 - Load, 546
 - LoadJoints, 547
 - LoadPlugins, 547
 - Model, 541
 - OnPoseChange, 547
 - ProcessMsg, 547
 - RemoveChild, 547
 - Reset, 547
 - SetAngularAccel, 547
 - SetAngularVel, 548
 - SetAutoDisable, 548
 - SetCollideMode, 548

- SetEnabled, 548
- SetGravityMode, 548
- SetJointAnimation, 548
- SetJointPosition, 549
- SetJointPositions, 549
- SetLaserRetro, 549
- SetLinearAccel, 549
- SetLinearVel, 550
- SetLinkWorldPose, 550
- SetScale, 550
- SetState, 550
- StopAnimation, 550
- Update, 551
- UpdateParameters, 551
- gazebo::physics::ModelState, 554
 - ~ModelState, 557
 - FillSDF, 557
 - GetJointState, 557, 558
 - GetJointStateCount, 558
 - GetJointStates, 558, 559
 - GetLinkState, 559
 - GetLinkStateCount, 559
 - GetLinkStates, 559, 560
 - GetPose, 560
 - HasJointState, 560
 - HasLinkState, 560
 - IsZero, 560
 - Load, 561
 - ModelState, 556, 557
 - operator<<, 562
 - operator+, 561
 - operator-, 561
 - operator=, 562
 - SetRealTime, 562
 - SetSimTime, 562
 - SetWallTime, 562
- gazebo::physics::MultiRayShape, 578
 - ~MultiRayShape, 581
 - AddRay, 581
 - ConnectNewLaserScans, 582
 - DisconnectNewLaserScans, 582
 - FillMsg, 582
 - GetFiducial, 582
 - GetMaxAngle, 583
 - GetMaxRange, 583
 - GetMinAngle, 583
 - GetMinRange, 583
 - GetRange, 583
 - GetResRange, 583
 - GetRetro, 584
 - GetSampleCount, 584
 - GetScanResolution, 584
 - GetVerticalMaxAngle, 584
 - GetVerticalMinAngle, 584
 - GetVerticalSampleCount, 584
 - GetVerticalScanResolution, 585
 - horzElem, 586
 - Init, 585
 - MultiRayShape, 581
 - newLaserScans, 586
 - offset, 586
 - ProcessMsg, 585
 - rangeElem, 586
 - rayElem, 586
 - rays, 586
 - scanElem, 586
 - SetScale, 585
 - Update, 585
 - UpdateRays, 585
 - vertElem, 586
- gazebo::physics::PhysicsEngine, 620
 - ~PhysicsEngine, 624
 - contactManager, 633
 - CreateCollision, 624
 - CreateJoint, 625
 - CreateLink, 625
 - CreateModel, 625
 - CreateShape, 625
 - DebugPrint, 625
 - Fini, 626
 - GetAutoDisableFlag, 626
 - GetContactManager, 626
 - GetContactMaxCorrectingVel, 626
 - GetContactSurfaceLayer, 626
 - GetGravity, 626
 - GetMaxContacts, 627
 - GetMaxStepSize, 627
 - GetParam, 627
 - GetPhysicsUpdateMutex, 627
 - GetRealTimeUpdateRate, 627
 - GetSORPGSIters, 628
 - GetSORPGSPreconIters, 628
 - GetSORPGSW, 628
 - GetTargetRealTimeFactor, 628
 - GetType, 628
 - GetUpdatePeriod, 628
 - GetWorldCFM, 629
 - GetWorldERP, 629
 - Init, 629
 - InitForThread, 629
 - Load, 629
 - maxStepSize, 633
 - node, 633
 - OnPhysicsMsg, 629
 - OnRequest, 630
 - PhysicsEngine, 624
 - physicsSub, 633
 - physicsUpdateMutex, 634

- realTimeUpdateRate, 634
- requestSub, 634
- Reset, 630
- responsePub, 634
- sdf, 634
- SetAutoDisableFlag, 630
- SetContactMaxCorrectingVel, 630
- SetContactSurfaceLayer, 630
- SetGravity, 631
- SetMaxContacts, 631
- SetMaxStepSize, 631
- SetParam, 631
- SetRealTimeUpdateRate, 631
- SetSORPGSIlters, 632
- SetSORPGSPreconIlters, 632
- SetSORPGSW, 632
- SetSeed, 632
- SetTargetRealTimeFactor, 632
- SetWorldCFM, 633
- SetWorldERP, 633
- targetRealTimeFactor, 634
- UpdateCollision, 633
- UpdatePhysics, 633
- world, 634
- gazebo::physics::PhysicsFactory, 634
 - IsRegistered, 635
 - NewPhysicsEngine, 635
 - RegisterAll, 635
 - RegisterPhysicsEngine, 635
- gazebo::physics::PlaneShape, 642
 - ~PlaneShape, 644
 - CreatePlane, 644
 - FillMsg, 644
 - GetNormal, 645
 - GetSize, 645
 - Init, 645
 - PlaneShape, 644
 - ProcessMsg, 645
 - SetAltitude, 645
 - SetNormal, 645
 - SetScale, 646
 - SetSize, 646
- gazebo::physics::RayShape, 700
 - ~RayShape, 702
 - contactFiducial, 705
 - contactLen, 705
 - contactRetro, 705
 - FillMsg, 702
 - GetFiducial, 702
 - GetGlobalPoints, 702
 - GetIntersection, 703
 - GetLength, 703
 - GetRelativePoints, 703
 - GetRetro, 703
 - globalEndPos, 705
 - globalStartPos, 705
 - Init, 703
 - ProcessMsg, 704
 - RayShape, 702
 - relativeEndPos, 705
 - relativeStartPos, 705
 - SetFiducial, 704
 - SetLength, 704
 - SetPoints, 704
 - SetRetro, 704
 - SetScale, 705
 - Update, 705
- gazebo::physics::Road, 718
 - ~Road, 719
 - Init, 719
 - Load, 719
 - Road, 719
- gazebo::physics::ScrewJoint
 - ~ScrewJoint, 747
 - fakeAnchor, 749
 - GetAnchor, 747
 - GetAngleCount, 748
 - GetThreadPitch, 748
 - Load, 748
 - ScrewJoint, 747
 - SetAnchor, 748
 - SetThreadPitch, 748
 - threadPitch, 749
- gazebo::physics::ScrewJoint< T >, 746
- gazebo::physics::Shape, 775
 - ~Shape, 777
 - collisionParent, 778
 - FillMsg, 777
 - GetScale, 777
 - Init, 777
 - ProcessMsg, 777
 - scale, 778
 - SetScale, 778
 - Shape, 777
- gazebo::physics::SimbodyBallJoint, 778
 - ~SimbodyBallJoint, 780
 - GetAnchor, 781
 - GetAngleImpl, 781
 - GetAxis, 781
 - GetGlobalAxis, 781
 - GetMaxForce, 781
 - GetVelocity, 782
 - Init, 782
 - Load, 782
 - SetDamping, 782
 - SetForceImpl, 782
 - SetHighStop, 783
 - SetLowStop, 783

- SetMaxForce, 783
- SetVelocity, 783
- SimbodyBallJoint, 780
- gazebo::physics::SimbodyBoxShape, 784
 - ~SimbodyBoxShape, 785
 - SetSize, 785
 - SimbodyBoxShape, 785
- gazebo::physics::SimbodyCollision, 785
 - ~SimbodyCollision, 787
 - GetBoundingBox, 787
 - GetCollisionShape, 787
 - Load, 787
 - OnPoseChange, 788
 - SetCategoryBits, 788
 - SetCollideBits, 788
 - SetCollisionShape, 788
 - SimbodyCollision, 787
- gazebo::physics::SimbodyCylinderShape, 788
 - ~SimbodyCylinderShape, 790
 - SetSize, 790
 - SimbodyCylinderShape, 790
- gazebo::physics::SimbodyHeightmapShape, 790
 - ~SimbodyHeightmapShape, 792
 - Init, 792
 - SimbodyHeightmapShape, 792
- gazebo::physics::SimbodyHinge2Joint, 792
 - ~SimbodyHinge2Joint, 794
 - GetAnchor, 795
 - GetAngleImpl, 795
 - GetAxis, 795
 - GetGlobalAxis, 795
 - GetHighStop, 795
 - GetLowStop, 796
 - GetMaxForce, 796
 - GetVelocity, 796
 - Init, 797
 - Load, 797
 - SetAxis, 797
 - SetDamping, 797
 - SetForceImpl, 797
 - SetHighStop, 797
 - SetLowStop, 798
 - SetMaxForce, 798
 - SetVelocity, 798
 - SimbodyHinge2Joint, 794
- gazebo::physics::SimbodyHingeJoint, 799
 - ~SimbodyHingeJoint, 800
 - GetAngleImpl, 801
 - GetGlobalAxis, 801
 - GetHighStop, 801
 - GetLowStop, 801
 - GetMaxForce, 802
 - GetVelocity, 802
 - Load, 802
- RestoreSimbodyState, 803
- SaveSimbodyState, 803
- SetAxis, 803
- SetDamping, 803
- SetForceImpl, 803
- SetHighStop, 804
- SetLowStop, 804
- SetMaxForce, 804
- SetVelocity, 804
- SimbodyHingeJoint, 800
- gazebo::physics::SimbodyJoint, 805
 - ~SimbodyJoint, 807
 - AreConnected, 807
 - CacheForceTorque, 807
 - constraint, 812
 - damper, 812
 - defxAB, 812
 - Detach, 807
 - GetAnchor, 807
 - GetAttribute, 808
 - GetForce, 808
 - GetForceTorque, 808
 - GetJointLink, 809
 - GetLinkForce, 809
 - GetLinkTorque, 809
 - isReversed, 812
 - limitForce, 812
 - Load, 810
 - mobod, 813
 - mustBreakLoopHere, 813
 - physicsInitialized, 813
 - Reset, 810
 - RestoreSimbodyState, 810
 - SaveSimbodyState, 810
 - SetAnchor, 810
 - SetAttribute, 810, 811
 - SetAxis, 811
 - SetDamping, 811
 - SetForce, 811
 - SetForceImpl, 812
 - SimbodyJoint, 807
 - simbodyPhysics, 813
 - world, 813
 - xCB, 813
 - xPA, 813
- gazebo::physics::SimbodyLink, 813
 - ~SimbodyLink, 816
 - AddForce, 816
 - AddForceAtRelativePosition, 816
 - AddForceAtWorldPosition, 817
 - AddRelativeForce, 817
 - AddRelativeTorque, 817
 - AddTorque, 817
 - Fini, 818

- GetEffectiveMassProps, 818
- GetEnabled, 818
- GetGravityMode, 818
- GetMassProperties, 818
- GetWorldAngularVel, 818
- GetWorldCoGLinearVel, 818
- GetWorldForce, 818
- GetWorldLinearVel, 819
- GetWorldTorque, 819
- Init, 819
- Load, 820
- masterMobod, 822
- mustBeBaseLink, 822
- OnPoseChange, 820
- physicsInitialized, 822
- RestoreSimbodyState, 820
- SaveSimbodyState, 820
- SetAngularDamping, 820
- SetAngularVel, 820
- SetAutoDisable, 820
- SetDirtyPose, 821
- SetEnabled, 821
- SetForce, 821
- SetGravityMode, 821
- SetLinearDamping, 821
- SetLinearVel, 821
- SetLinkStatic, 822
- SetSelfCollide, 822
- SetTorque, 822
- SimbodyLink, 816
- slaveMobods, 822
- slaveWelds, 822
- gazebo::physics::SimbodyMeshShape, 823
 - ~SimbodyMeshShape, 824
 - Init, 824
 - Load, 824
 - SimbodyMeshShape, 824
- gazebo::physics::SimbodyModel, 825
 - ~SimbodyModel, 826
 - Init, 826
 - Load, 826
 - SimbodyModel, 826
- gazebo::physics::SimbodyMultiRayShape, 826
 - ~SimbodyMultiRayShape, 828
 - AddRay, 828
 - SimbodyMultiRayShape, 828
 - UpdateRays, 828
- gazebo::physics::SimbodyPhysics, 828
 - ~SimbodyPhysics, 831
 - contact, 837
 - CreateCollision, 831
 - CreateJoint, 831
 - CreateLink, 831
 - CreateModel, 832
 - CreateShape, 832
 - DebugPrint, 832
 - discreteForces, 837
 - Fini, 832
 - forces, 837
 - GetDynamicsWorld, 832
 - GetPose, 832
 - GetType, 833
 - GetTypeString, 833
 - gravity, 837
 - Init, 833
 - InitForThread, 833
 - InitModel, 833
 - integ, 837
 - Load, 834
 - matter, 837
 - OnPhysicsMsg, 834
 - OnRequest, 834
 - Pose2Transform, 834
 - QuadToQuad, 835
 - Reset, 835
 - SetGravity, 835
 - SetSeed, 835
 - SimbodyPhysics, 831
 - simbodyPhysicsInitialized, 837
 - simbodyPhysicsStepped, 837
 - system, 837
 - tracker, 837
 - Transform2Pose, 836
 - UpdateCollision, 836
 - UpdatePhysics, 836
 - Vec3ToVector3, 836
 - Vector3ToVec3, 836
- gazebo::physics::SimbodyPlaneShape, 837
 - ~SimbodyPlaneShape, 839
 - CreatePlane, 839
 - SetAltitude, 839
 - SimbodyPlaneShape, 839
- gazebo::physics::SimbodyRayShape, 839
 - ~SimbodyRayShape, 841
 - GetIntersection, 841
 - SetPoints, 841
 - SimbodyRayShape, 841
 - Update, 842
- gazebo::physics::SimbodyScrewJoint, 842
 - ~SimbodyScrewJoint, 845
 - GetAngleImpl, 845
 - GetGlobalAxis, 845
 - GetHighStop, 845
 - GetLowStop, 845
 - GetMaxForce, 846
 - GetThreadPitch, 846
 - GetVelocity, 846
 - Init, 847

- Load, 847
- SetAxis, 847
- SetDamping, 847
- SetForcelImpl, 847
- SetHighStop, 848
- SetLowStop, 848
- SetMaxForce, 848
- SetThreadPitch, 848
- SetVelocity, 849
- SimbodyScrewJoint, 844
- gazebo::physics::SimbodySliderJoint, 849
 - ~SimbodySliderJoint, 851
 - GetAngleImpl, 852
 - GetGlobalAxis, 852
 - GetHighStop, 852
 - GetLowStop, 852
 - GetMaxForce, 853
 - GetVelocity, 853
 - Load, 853
 - SetAxis, 853
 - SetDamping, 854
 - SetForcelImpl, 854
 - SetHighStop, 854
 - SetLowStop, 854
 - SetMaxForce, 855
 - SetVelocity, 855
 - SimbodySliderJoint, 851
- gazebo::physics::SimbodySphereShape, 855
 - ~SimbodySphereShape, 857
 - SetRadius, 857
 - SimbodySphereShape, 856
- gazebo::physics::SimbodyUniversalJoint, 857
 - ~SimbodyUniversalJoint, 860
 - GetAnchor, 860
 - GetAngleImpl, 860
 - GetAxis, 860
 - GetGlobalAxis, 860
 - GetHighStop, 860
 - GetLowStop, 861
 - GetMaxForce, 861
 - GetVelocity, 861
 - Init, 862
 - Load, 862
 - SetAxis, 862
 - SetDamping, 862
 - SetForcelImpl, 862
 - SetHighStop, 863
 - SetLowStop, 863
 - SetMaxForce, 863
 - SetVelocity, 863
 - SimbodyUniversalJoint, 859
- gazebo::physics::SliderJoint
 - ~SliderJoint, 887
 - fakeAnchor, 888
 - GetAnchor, 887
 - GetAngleCount, 888
 - Load, 888
 - SetAnchor, 888
 - SliderJoint, 887
- gazebo::physics::SliderJoint< T >, 886
- gazebo::physics::SphereShape, 898
 - ~SphereShape, 900
 - FillMsg, 900
 - GetRadius, 900
 - Init, 900
 - ProcessMsg, 900
 - SetRadius, 901
 - SetScale, 901
 - SphereShape, 900
- gazebo::physics::State, 910
 - ~State, 912
 - GetName, 912
 - GetRealTime, 913
 - GetSimTime, 913
 - GetWallTime, 913
 - Load, 913
 - name, 915
 - operator-, 913
 - operator=, 914
 - realTime, 915
 - SetName, 914
 - SetRealTime, 914
 - SetSimTime, 914
 - SetWallTime, 914
 - simTime, 915
 - State, 912
 - wallTime, 915
- gazebo::physics::SurfaceParams, 933
 - ~SurfaceParams, 934
 - bounce, 935
 - bounceThreshold, 935
 - cfm, 935
 - collideWithoutContact, 935
 - collideWithoutContactBitmask, 935
 - erp, 935
 - fdir1, 935
 - FillMsg, 934
 - kd, 936
 - kp, 936
 - Load, 934
 - maxVel, 936
 - minDepth, 936
 - mu1, 936
 - mu2, 937
 - ProcessMsg, 935
 - slip1, 937
 - slip2, 937
 - SurfaceParams, 934

- gazebo::physics::TrajectoryInfo, 974
 - duration, 974
 - endTime, 974
 - id, 974
 - startTime, 974
 - translated, 974
 - type, 974
- gazebo::physics::UniversalJoint
 - ~UniversalJoint, 977
 - GetAngleCount, 978
 - Load, 978
 - UniversalJoint, 977
- gazebo::physics::UniversalJoint< T >, 976
- gazebo::physics::World, 1070
 - ~World, 1073
 - Clear, 1073
 - dirtyPoses, 1081
 - DisableAllModels, 1073
 - EnableAllModels, 1073
 - EnablePhysicsEngine, 1073
 - Fini, 1073
 - GetByName, 1074
 - GetEnablePhysicsEngine, 1074
 - GetEntity, 1074
 - GetEntityBelowPoint, 1074
 - GetModel, 1075
 - GetModelBelowPoint, 1075
 - GetModelCount, 1075
 - GetModels, 1076
 - GetName, 1076
 - GetPauseTime, 1076
 - GetPhysicsEngine, 1076
 - GetRealTime, 1076
 - GetRunning, 1076
 - GetSelectedEntity, 1077
 - GetSetWorldPoseMutex, 1077
 - GetSimTime, 1077
 - GetSphericalCoordinates, 1077
 - GetStartTime, 1077
 - Init, 1077
 - InsertModelFile, 1078
 - InsertModelSDF, 1078
 - InsertModelString, 1078
 - IsLoaded, 1078
 - IsPaused, 1078
 - Load, 1078
 - LoadPlugin, 1079
 - PrintEntityTree, 1079
 - PublishModelPose, 1079
 - RemovePlugin, 1079
 - Reset, 1079
 - ResetEntities, 1079
 - ResetTime, 1080
 - Run, 1080
 - Save, 1080
 - SetPaused, 1080
 - SetSimTime, 1080
 - SetState, 1080
 - StepWorld, 1081
 - Stop, 1081
 - StripWorldName, 1081
 - UpdateStateSDF, 1081
 - World, 1073
- gazebo::physics::WorldState, 1083
 - ~WorldState, 1085
 - FillSDF, 1085
 - GetModelState, 1085
 - GetModelStateCount, 1086
 - GetModelStates, 1086
 - HasModelState, 1086
 - IsZero, 1087
 - Load, 1087
 - operator<<, 1089
 - operator+, 1087
 - operator-, 1087
 - operator=, 1088
 - SetRealTime, 1088
 - SetSimTime, 1088
 - SetWallTime, 1088
 - SetWorld, 1089
 - WorldState, 1085
- gazebo::rendering, 111
 - ArrowVisualPtr, 114
 - AxisVisualPtr, 114
 - COMVisualPtr, 114
 - CameraPtr, 114
 - CameraVisualPtr, 114
 - ContactVisualPtr, 114
 - DepthCameraPtr, 114
 - DynamicLinesPtr, 114
 - GpuLaserPtr, 114
 - JointVisualPtr, 114
 - LaserVisualPtr, 114
 - LightPtr, 114
 - RENDERING_LINE_LIST, 115
 - RENDERING_LINE_STRIP, 115
 - RENDERING_MESH_RESOURCE, 115
 - RENDERING_POINT_LIST, 115
 - RENDERING_TRIANGLE_FAN, 115
 - RENDERING_TRIANGLE_LIST, 115
 - RENDERING_TRIANGLE_STRIP, 115
 - RFIDTagVisualPtr, 114
 - RFIDVisualPtr, 114
 - RenderOpType, 115
 - ScenePtr, 114
 - SelectionObjPtr, 114
 - SonarVisualPtr, 114
 - UserCameraPtr, 114

- VisualPtr, 114
- WindowManagerPtr, 114
- WrenchVisualPtr, 115
- gazebo::rendering::ArrowVisual, 143
 - ~ArrowVisual, 145
 - ArrowVisual, 144
 - Load, 145
 - ShowRotation, 145
- gazebo::rendering::AxisVisual, 149
 - ~AxisVisual, 150
 - AxisVisual, 150
 - Load, 150
 - ScaleXAxis, 150
 - ScaleYAxis, 150
 - ScaleZAxis, 150
 - SetAxisMaterial, 151
 - ShowRotation, 151
- gazebo::rendering::COMVisual, 237
 - ~COMVisual, 239
 - COMVisual, 238
 - Load, 239
- gazebo::rendering::Camera, 179
 - ~Camera, 185
 - animState, 203
 - AnimationComplete, 186
 - AttachToVisual, 186
 - AttachToVisualImpl, 186, 187
 - bayerFrameBuffer, 203
 - Camera, 185
 - camera, 203
 - captureData, 203
 - captureDataOnce, 203
 - ConnectNewImageFrame, 187
 - connections, 204
 - CreateRenderTexture, 188
 - DisconnectNewImageFrame, 188
 - EnableSaveFrame, 188
 - Fini, 188
 - GetAspectRatio, 188
 - GetAvgFPS, 188
 - GetCameraToViewportRay, 189
 - GetCaptureData, 189
 - GetDirection, 189
 - GetFarClip, 189
 - GetFrameFilename, 189
 - GetHFOV, 189
 - GetImageByteSize, 190
 - GetImageData, 190
 - GetImageDepth, 190
 - GetImageFormat, 191
 - GetImageHeight, 191
 - GetImageWidth, 191
 - GetInitialized, 191
 - GetLastRenderWallTime, 191
 - GetName, 191
 - GetNearClip, 192
 - GetOgreCamera, 192
 - GetPitchNode, 192
 - GetRenderRate, 192
 - GetRenderTexture, 192
 - GetRight, 192
 - GetScene, 193
 - GetSceneNode, 193
 - GetScreenshotPath, 193
 - GetTextureHeight, 193
 - GetTextureWidth, 193
 - GetTriangleCount, 193
 - GetUp, 194
 - GetVFOV, 194
 - GetViewport, 194
 - GetViewportHeight, 194
 - GetViewportWidth, 194
 - GetWindowId, 194
 - GetWorldPointOnPlane, 195
 - GetWorldPose, 195
 - GetWorldPosition, 195
 - GetWorldRotation, 195
 - GetZValue, 195
 - imageFormat, 204
 - imageHeight, 204
 - imageWidth, 204
 - Init, 196
 - initialized, 204
 - IsAnimating, 196
 - IsVisible, 196
 - lastRenderWallTime, 204
 - Load, 196, 197
 - MoveToPosition, 197
 - MoveToPositions, 197
 - name, 204
 - newData, 204
 - newImageFrame, 204
 - onAnimationComplete, 204
 - pitchNode, 204
 - PostRender, 197
 - prevAnimTime, 205
 - ReadPixelBuffer, 198
 - Render, 198
 - RenderImpl, 198
 - renderTarget, 205
 - renderTexture, 205
 - requests, 205
 - RotatePitch, 198
 - RotateYaw, 198
 - saveCount, 205
 - SaveFrame, 198, 199
 - saveFrameBuffer, 205
 - scene, 205

- sceneNode, 205
- screenshotPath, 205
- sdf, 205
- SetAspectRatio, 199
- SetCaptureData, 199
- SetCaptureDataOnce, 199
- SetClipDist, 199
- SetHFOV, 200
- SetImageHeight, 200
- SetImageSize, 200
- SetImageWidth, 200
- SetName, 200
- SetRenderRate, 200
- SetRenderTarget, 201
- SetSaveFramePathname, 201
- SetScene, 201
- SetSceneNode, 201
- SetWindowId, 201
- SetWorldPose, 201
- SetWorldPosition, 202
- SetWorldRotation, 202
- ShowWireframe, 202
- textureHeight, 205
- textureWidth, 205
- ToggleShowWireframe, 202
- TrackVisual, 202
- TrackVisualImpl, 202, 203
- Translate, 203
- Update, 203
- viewport, 206
- windowId, 206
- gazebo::rendering::CameraVisual, 210
 - ~CameraVisual, 211
 - CameraVisual, 211
 - Load, 211
- gazebo::rendering::ContactVisual, 265
 - ~ContactVisual, 266
 - ContactVisual, 266
 - SetEnabled, 266
- gazebo::rendering::Conversions, 266
 - Convert, 267, 268
- gazebo::rendering::DepthCamera, 272
 - ~DepthCamera, 274
 - ConnectNewDepthFrame, 274
 - ConnectNewRGBPointCloud, 274
 - CreateDepthTexture, 274
 - DepthCamera, 274
 - depthTarget, 276
 - depthTexture, 276
 - depthViewport, 276
 - DisconnectNewDepthFrame, 275
 - DisconnectNewRGBPointCloud, 275
 - Fini, 275
 - GetDepthData, 275
 - Init, 275
 - Load, 275, 276
 - PostRender, 276
 - SetDepthTarget, 276
- gazebo::rendering::DummyPageProvider, 284
 - loadProceduralPage, 285
 - prepareProceduralPage, 285
 - unloadProceduralPage, 285
 - unprepareProceduralPage, 286
- gazebo::rendering::DynamicLines, 286
 - ~DynamicLines, 287
 - AddPoint, 287, 288
 - Clear, 288
 - DynamicLines, 287
 - GetMovableType, 288
 - getMovableType, 288
 - GetPoint, 288
 - GetPointCount, 289
 - SetColor, 289
 - SetPoint, 289
 - Update, 289
- gazebo::rendering::DynamicRenderable, 289
 - ~DynamicRenderable, 291
 - CreateVertexDeclaration, 291
 - DynamicRenderable, 291
 - FillHardwareBuffers, 291
 - getBoundingRadius, 291
 - GetMovableType, 292
 - GetOperationType, 292
 - getSquaredViewDepth, 292
 - indexBufferCapacity, 293
 - Init, 292
 - PrepareHardwareBuffers, 293
 - SetOperationType, 293
 - vertexBufferCapacity, 293
- gazebo::rendering::Events, 308
 - ConnectCreateScene, 308
 - ConnectRemoveScene, 309
 - createScene, 309
 - DisconnectCreateScene, 309
 - DisconnectRemoveScene, 309
 - removeScene, 310
- gazebo::rendering::FPSViewController, 337
 - ~FPSViewController, 339
 - FPSViewController, 339
 - GetTypeString, 339
 - HandleKeyPressEvent, 339
 - HandleKeyReleaseEvent, 339
 - HandleMouseEvent, 339
 - Init, 340
 - Update, 340
- gazebo::rendering::GUIOverlay, 370
 - ~GUIOverlay, 371
 - AttachCameraToImage, 371, 372

- ButtonCallback, 372
- CreateWindow, 372
- GUIOverlay, 371
- HandleKeyPressEvent, 372
- HandleKeyReleaseEvent, 373
- HandleMouseEvent, 373
- Hide, 373
- Init, 373
- IsInitialized, 373
- LoadLayout, 374
- Resize, 374
- Show, 374
- Update, 374
- gazebo::rendering::GpuLaser, 344
 - ~GpuLaser, 347
 - cameraCount, 352
 - chfov, 352
 - ConnectNewLaserFrame, 347
 - CreateLaserTexture, 347
 - cvfov, 352
 - DisconnectNewLaserFrame, 347
 - far, 352
 - Fini, 348
 - GetCameraCount, 348
 - GetCosHorzFOV, 348
 - GetCosVertFOV, 348
 - GetFarClip, 348
 - GetHorzFOV, 348
 - GetHorzHalfAngle, 348
 - GetLaserData, 349
 - GetNearClip, 349
 - GetRayCountRatio, 349
 - GetVertFOV, 349
 - GetVertHalfAngle, 349
 - GpuLaser, 347
 - hfov, 353
 - horzHalfAngle, 353
 - Init, 349
 - IsHorizontal, 350
 - isHorizontal, 353
 - Load, 350
 - near, 353
 - notifyRenderSingleObject, 350
 - PostRender, 350
 - rayCountRatio, 353
 - SetCameraCount, 350
 - SetCosHorzFOV, 350
 - SetCosVertFOV, 350
 - SetFarClip, 351
 - SetHorzFOV, 351
 - SetHorzHalfAngle, 351
 - SetIsHorizontal, 351
 - SetNearClip, 351
 - SetRangeCount, 351
 - SetRayCountRatio, 352
 - SetVertFOV, 352
 - SetVertHalfAngle, 352
 - vertHalfAngle, 353
 - vfov, 353
- gazebo::rendering::Grid, 365
 - ~Grid, 366
 - Enable, 366
 - GetCellCount, 366
 - GetCellLength, 366
 - GetColor, 367
 - GetHeight, 367
 - GetLineWidth, 367
 - GetSceneNode, 367
 - Grid, 366
 - Init, 367
 - SetCellCount, 367
 - SetCellLength, 367
 - SetColor, 368
 - SetHeight, 368
 - SetLineWidth, 368
 - SetUserData, 368
- gazebo::rendering::GzTerrainMatGen, 374
 - ~GzTerrainMatGen, 375
 - GzTerrainMatGen, 375
- gazebo::rendering::GzTerrainMatGen::SM2Profile, 888
 - ~SM2Profile, 890
 - addTechnique, 890
 - generate, 890
 - generateForCompositeMap, 890
 - SM2Profile, 890
 - UpdateParams, 890
 - UpdateParamsForCompositeMap, 890
- gazebo::rendering::GzTerrainMatGen::SM2Profile::-
 - ShaderHelperCg, 771
 - defaultVpParams, 772
 - generateFragmentProgram, 772
 - generateVertexProgram, 772
 - generateVertexProgramSource, 772
 - generateVpDynamicShadows, 772
 - generateVpDynamicShadowsParams, 772
 - generateVpFooter, 772
 - generateVpHeader, 772
- gazebo::rendering::GzTerrainMatGen::SM2Profile::-
 - ShaderHelperGLSL, 773
 - defaultVpParams, 774
 - generateFpDynamicShadows, 774
 - generateFpDynamicShadowsHelpers, 774
 - generateFpDynamicShadowsParams, 774
 - generateFpFooter, 774
 - generateFpHeader, 774
 - generateFpLayer, 774
 - generateFragmentProgram, 774
 - generateFragmentProgramSource, 775

- generateVertexProgram, 775
- generateVertexProgramSource, 775
- generateVpDynamicShadows, 775
- generateVpDynamicShadowsParams, 775
- generateVpFooter, 775
- generateVpHeader, 775
- updateParams, 775
- updateVpParams, 775
- gazebo::rendering::Heightmap, 375
 - ~Heightmap, 377
 - Flatten, 377
 - GetAvgHeight, 377
 - GetHeight, 377
 - GetImage, 378
 - GetMouseHit, 378
 - GetOgreTerrain, 378
 - Heightmap, 377
 - Load, 378
 - LoadFromMsg, 378
 - Lower, 378
 - NumTerrainSubdivisions, 380
 - Raise, 379
 - SetWireframe, 379
 - Smooth, 379
 - SplitHeights, 380
- gazebo::rendering::JointVisual, 441
 - ~JointVisual, 442
 - JointVisual, 441
 - Load, 442
- gazebo::rendering::LaserVisual, 447
 - ~LaserVisual, 448
 - LaserVisual, 448
 - SetEmissive, 448
- gazebo::rendering::Light, 448
 - ~Light, 450
 - FillMsg, 450
 - GetDiffuseColor, 450
 - GetDirection, 451
 - GetName, 451
 - GetPosition, 451
 - GetSpecularColor, 451
 - GetType, 451
 - Light, 450
 - Load, 451, 452
 - LoadFromMsg, 452
 - OnPoseChange, 452
 - SetAttenuation, 452
 - SetCastShadows, 452
 - SetDiffuseColor, 452
 - SetDirection, 453
 - SetLightType, 453
 - SetName, 453
 - SetPosition, 453
 - SetRange, 453
 - SetSelected, 453
 - SetSpecularColor, 454
 - SetSpotFalloff, 454
 - SetSpotInnerAngle, 454
 - SetSpotOuterAngle, 454
 - ShowVisual, 454
 - ToggleShowVisual, 454
 - UpdateFromMsg, 454
- gazebo::rendering::MovableText, 566
 - ~MovableText, 568
 - _setupGeometry, 569
 - _updateColors, 569
 - GetAABB, 569
 - GetBaseline, 569
 - getBoundingRadius, 569
 - GetCharHeight, 569
 - GetColor, 569
 - GetFont, 569
 - getLights, 569
 - getMaterial, 569
 - getRenderOperation, 569
 - GetShowOnTop, 570
 - GetSpaceWidth, 570
 - getSquaredViewDepth, 570
 - GetText, 570
 - getWorldTransforms, 570
 - H_CENTER, 568
 - H_LEFT, 568
 - HorizAlign, 568
 - Load, 570
 - MovableText, 568
 - SetBaseline, 570
 - SetCharHeight, 571
 - SetColor, 571
 - SetFontName, 571
 - SetShowOnTop, 571
 - SetSpaceWidth, 571
 - SetText, 571
 - SetTextAlignment, 572
 - Update, 572
 - V_ABOVE, 568
 - V_BELOW, 568
 - VertAlign, 568
 - visitRenderables, 572
- gazebo::rendering::OrbitViewController, 617
 - ~OrbitViewController, 618
 - GetFocalPoint, 618
 - GetTypeString, 618
 - HandleKeyPressEvent, 619
 - HandleKeyReleaseEvent, 619
 - HandleMouseEvent, 619
 - Init, 619
 - OrbitViewController, 618
 - SetDistance, 620

- SetFocalPoint, 620
- Update, 620
- gazebo::rendering::Projector, 662
 - ~Projector, 663
 - GetParent, 663
 - Load, 663
 - Projector, 663
 - SetEnabled, 664
 - SetTexture, 664
 - Toggle, 664
- gazebo::rendering::RFIDTagVisual, 715
 - ~RFIDTagVisual, 716
 - RFIDTagVisual, 716
- gazebo::rendering::RFIDVisual, 716
 - ~RFIDVisual, 717
 - RFIDVisual, 717
- gazebo::rendering::RTShaderSystem, 724
 - AddScene, 726
 - ApplyShadows, 726
 - AttachEntity, 726
 - AttachViewport, 726
 - Clear, 727
 - DetachEntity, 727
 - DetachViewport, 727
 - Fini, 727
 - GenerateShaders, 727
 - GetPSSMShadowCameraSetup, 727
 - Init, 727
 - LightingModel, 726
 - RemoveScene, 727
 - RemoveShadows, 728
 - SSLM_NormalMapLightingObjectSpace, 726
 - SSLM_NormalMapLightingTangentSpace, 726
 - SSLM_PerPixelLighting, 726
 - SSLM_PerVertexLighting, 726
 - SetPerPixelLighting, 728
 - UpdateShaders, 728
- gazebo::rendering::RenderEngine, 706
 - AddResourcePath, 708
 - CreateScene, 708
 - DEFERRED, 707
 - dummyContext, 709
 - dummyDisplay, 709
 - dummyWindowId, 710
 - FORWARD, 707
 - Fini, 708
 - GetRenderPathType, 708
 - GetScene, 708
 - GetSceneCount, 709
 - GetWindowManager, 709
 - Init, 709
 - Load, 709
 - NONE, 707
 - RENDER_PATH_COUNT, 707
 - RemoveScene, 709
 - RenderPathType, 707
 - root, 710
 - VERTEX, 707
- gazebo::rendering::Road2d, 720
 - ~Road2d, 720
 - Load, 720
 - Road2d, 720
- gazebo::rendering::Scene, 728
 - ~Scene, 733
 - AddVisual, 733
 - Clear, 733
 - CloneVisual, 733
 - CreateCamera, 733
 - CreateDepthCamera, 733
 - CreateGpuLaser, 734
 - CreateGrid, 734
 - CreateUserCamera, 734
 - DrawLine, 735
 - GZ_SKYX_ALL, 732
 - GZ_SKYX_CLOUDS, 732
 - GZ_SKYX_MOON, 732
 - GZ_SKYX_NONE, 732
 - GetAmbientColor, 735
 - GetBackgroundColor, 735
 - GetCamera, 735
 - GetCameraCount, 736
 - GetFirstContact, 736
 - GetGrid, 736
 - GetGridCount, 736
 - GetHeightBelowPoint, 736
 - GetHeightmap, 737
 - GetId, 737
 - GetIdString, 737
 - GetInitialized, 737
 - GetLight, 737, 738
 - GetLightCount, 738
 - GetManager, 738
 - GetModelVisualAt, 738
 - GetName, 738
 - GetSelectedVisual, 739
 - GetShadowsEnabled, 739
 - GetShowClouds, 739
 - GetSimTime, 739
 - GetUserCamera, 739
 - GetUserCameraCount, 740
 - GetVisual, 740
 - GetVisualAt, 740, 741
 - GetVisualBelow, 741
 - GetVisualCount, 741
 - GetVisualsBelowPoint, 741
 - GetWorldVisual, 741
 - Init, 742
 - Load, 742

- PreRender, 742
- PrintSceneGraph, 742
- RemoveCamera, 742
- RemoveVisual, 742
- Scene, 733
- SelectVisual, 742
- SetAmbientColor, 743
- SetBackgroundColor, 743
- SetFog, 743
- SetGrid, 743
- SetShadowsEnabled, 743
- SetSkyXMode, 744
- SetTransparent, 744
- SetVisible, 744
- SetWireframe, 744
- ShowCOMs, 745
- ShowClouds, 744
- ShowCollisions, 745
- ShowContacts, 745
- ShowJoints, 745
- SkyXMode, 732
- skyx, 746
- SnapVisualToNearestBelow, 745
- StripSceneName, 745
- gazebo::rendering::SelectionObj, 749
- gazebo::rendering::SonarVisual, 895
 - ~SonarVisual, 896
 - Load, 896
 - SonarVisual, 896
- gazebo::rendering::TransmitterVisual, 975
 - ~TransmitterVisual, 976
 - Load, 976
 - TransmitterVisual, 976
 - Update, 976
- gazebo::rendering::UserCamera, 979
 - ~UserCamera, 981
 - AnimationComplete, 981
 - AttachToVisualImpl, 981
 - EnableViewController, 982
 - Fini, 982
 - GetAvgFPS, 982
 - GetGUIOverlay, 982
 - GetImageHeight, 983
 - GetImageWidth, 983
 - GetTriangleCount, 983
 - GetViewControllerTypeString, 983
 - GetVisual, 983, 984
 - HandleKeyPressEvent, 984
 - HandleKeyReleaseEvent, 984
 - HandleMouseEvent, 984
 - Init, 984
 - Load, 984, 985
 - MoveToPosition, 985
 - MoveToVisual, 985
 - PostRender, 985
 - Resize, 985
 - SetFocalPoint, 986
 - SetRenderTarget, 986
 - SetViewController, 986
 - SetViewportDimensions, 986
 - SetWorldPose, 987
 - TrackVisualImpl, 987
 - Update, 987
 - UserCamera, 981
- gazebo::rendering::VideoVisual, 1029
 - ~VideoVisual, 1030
 - VideoVisual, 1030
- gazebo::rendering::ViewController, 1031
 - ~ViewController, 1032
 - camera, 1034
 - enabled, 1034
 - GetTypeString, 1032
 - HandleKeyPressEvent, 1032
 - HandleKeyReleaseEvent, 1033
 - HandleMouseEvent, 1033
 - Init, 1033
 - SetEnabled, 1033
 - typeString, 1034
 - Update, 1034
 - ViewController, 1032
- gazebo::rendering::Visual, 1034
 - ~Visual, 1040
 - AttachAxes, 1040
 - AttachLineVertex, 1040
 - AttachMesh, 1041
 - AttachObject, 1041
 - AttachVisual, 1041
 - ClearParent, 1041
 - Clone, 1041
 - CreateDynamicLine, 1041
 - DeleteDynamicLine, 1042
 - DetachObjects, 1042
 - DetachVisual, 1042
 - DisableTrackVisual, 1042
 - EnableTrackVisual, 1042
 - Fini, 1043
 - GetAttachedObjectCount, 1043
 - GetBoundingBox, 1043
 - GetChild, 1043
 - GetChildCount, 1043
 - GetId, 1043
 - GetMaterialName, 1043
 - GetMeshName, 1044
 - GetName, 1044
 - GetNormalMap, 1044
 - GetParent, 1044
 - GetPose, 1044
 - GetPosition, 1044

GetRootVisual, 1045
 GetRotation, 1045
 GetScale, 1045
 GetScene, 1045
 GetSceneNode, 1045
 GetShaderType, 1045
 GetSubMeshName, 1046
 GetTransparency, 1046
 GetVisibilityFlags, 1046
 GetVisible, 1046
 GetWorldPose, 1046
 HasAttachedObject, 1047
 Init, 1047
 InsertMesh, 1047
 IsPlane, 1047
 IsStatic, 1048
 Load, 1048
 LoadFromMsg, 1048
 LoadPlugin, 1048
 MakeStatic, 1048
 MoveToPosition, 1049
 MoveToPositions, 1049
 parent, 1055
 RemovePlugin, 1049
 scene, 1055
 sceneNode, 1055
 SetAmbient, 1049
 SetCastShadows, 1049
 SetDiffuse, 1049
 SetEmissive, 1050
 SetHighlighted, 1050
 SetId, 1050
 SetMaterial, 1050
 SetName, 1050
 SetNormalMap, 1050
 SetPose, 1051
 SetPosition, 1051
 SetRibbonTrail, 1051
 SetRotation, 1051
 SetScale, 1051
 SetScene, 1052
 SetShaderType, 1052
 SetSkeletonPose, 1052
 SetSpecular, 1052
 SetTransparency, 1052
 SetVisibilityFlags, 1052
 SetVisible, 1053
 SetWireframe, 1053
 SetWorldPose, 1053
 SetWorldPosition, 1053
 SetWorldRotation, 1053
 ShowBoundingBox, 1054
 ShowCOM, 1054
 ShowCollision, 1054
 ShowJoints, 1054
 ShowSkeleton, 1054
 ToggleVisible, 1054
 Update, 1054
 UpdateFromMsg, 1054
 Visual, 1040
 gazebo::rendering::WindowManager, 1057
 ~WindowManager, 1057
 CreateWindow, 1058
 Fini, 1058
 GetAvgFPS, 1058
 GetTriangleCount, 1058
 GetWindow, 1058
 Moved, 1059
 Resize, 1059
 SetCamera, 1059
 WindowManager, 1057
 gazebo::rendering::WireBox, 1059
 ~WireBox, 1060
 Init, 1060
 SetVisible, 1060
 WireBox, 1060
 gazebo::rendering::WrenchVisual, 1089
 ~WrenchVisual, 1091
 Load, 1091
 SetEnabled, 1091
 WrenchVisual, 1090
 gazebo::sensors, 115
 CATEGORY_COUNT, 119
 CameraSensor_V, 118
 CameraSensorPtr, 118
 ContactSensor_V, 118
 ContactSensorPtr, 118
 DepthCameraSensor_V, 118
 DepthCameraSensorPtr, 118
 ForceTorqueSensorPtr, 118
 GpsSensorPtr, 118
 GpuRaySensor_V, 118
 GpuRaySensorPtr, 118
 IMAGE, 119
 ImuSensor_V, 118
 ImuSensorPtr, 118
 NoisePtr, 118
 OTHER, 119
 RAY, 119
 RFIDSensor_V, 118
 RFIDSensorPtr, 118
 RFIDTag_V, 118
 RFIDTagPtr, 118
 RaySensor_V, 118
 RaySensorPtr, 118
 Sensor_V, 118
 SensorCategory, 119
 SensorFactoryFn, 118

- SensorPtr, 118
- SonarSensorPtr, 118
- WirelessReceiver_V, 118
- WirelessReceiverPtr, 119
- WirelessTransceiver_V, 119
- WirelessTransceiverPtr, 119
- WirelessTransmitter_V, 119
- WirelessTransmitterPtr, 119
- gazebo::sensors::CameraSensor, 206
 - ~CameraSensor, 207
 - CameraSensor, 207
 - Fini, 208
 - GetCamera, 208
 - GetImageData, 208
 - GetImageHeight, 208
 - GetImageWidth, 208
 - GetTopic, 208
 - Init, 208
 - IsActive, 209
 - Load, 209
 - SaveFrame, 209
 - UpdateImpl, 209
- gazebo::sensors::ContactSensor, 260
 - ~ContactSensor, 262
 - ContactSensor, 262
 - Fini, 262
 - GetCollisionContactCount, 262
 - GetCollisionCount, 262
 - GetCollisionName, 263
 - GetContacts, 263
 - Init, 264
 - IsActive, 264
 - Load, 264
 - UpdateImpl, 264
- gazebo::sensors::DepthCameraSensor, 276
 - ~DepthCameraSensor, 278
 - DepthCameraSensor, 278
 - Fini, 278
 - GetDepthCamera, 278
 - Init, 278
 - Load, 278
 - SaveFrame, 279
 - SetActive, 279
 - UpdateImpl, 279
- gazebo::sensors::ForceTorqueSensor, 333
 - ~ForceTorqueSensor, 335
 - ConnectUpdate, 335
 - DisconnectUpdate, 335
 - Fini, 336
 - ForceTorqueSensor, 335
 - GetForce, 336
 - GetTopic, 336
 - GetTorque, 336
 - Init, 336
 - IsActive, 336
 - Load, 336
 - update, 337
 - UpdateImpl, 337
- gazebo::sensors::GpsSensor, 341
 - ~GpsSensor, 342
 - Fini, 342
 - GetAltitude, 342
 - GetLatitude, 343
 - GetLongitude, 343
 - GpsSensor, 342
 - Init, 343
 - Load, 343
 - UpdateImpl, 343
- gazebo::sensors::GpuRaySensor, 353
 - ~GpuRaySensor, 356
 - cameraElem, 364
 - ConnectNewLaserFrame, 357
 - DisconnectNewLaserFrame, 357
 - Fini, 357
 - GetAngleMax, 357
 - GetAngleMin, 357
 - GetAngleResolution, 357
 - GetCameraCount, 357
 - GetCosHorzFOV, 358
 - GetCosVertFOV, 358
 - GetFiducial, 358
 - GetHorzFOV, 358
 - GetHorzHalfAngle, 358
 - GetLaserCamera, 359
 - GetRange, 359
 - GetRangeCount, 359
 - GetRangeCountRatio, 359
 - GetRangeMax, 359
 - GetRangeMin, 360
 - GetRangeResolution, 360
 - GetRanges, 360
 - GetRayCount, 360
 - GetRayCountRatio, 360
 - GetRetro, 360
 - GetTopic, 361
 - GetVertFOV, 361
 - GetVertHalfAngle, 361
 - GetVerticalAngleMax, 361
 - GetVerticalAngleMin, 361
 - GetVerticalRangeCount, 362
 - GetVerticalRayCount, 362
 - GpuRaySensor, 356
 - horzElem, 364
 - horzRangeCount, 364
 - horzRayCount, 364
 - Init, 362
 - IsActive, 362
 - IsHorizontal, 362

- Load, 362, 363
- rangeCountRatio, 364
- rangeElem, 364
- scanElem, 364
- SetAngleMax, 363
- SetAngleMin, 363
- SetVerticalAngleMax, 363
- SetVerticalAngleMin, 363
- UpdateImpl, 364
- vertElem, 364
- vertRangeCount, 364
- vertRayCount, 365
- gazebo::sensors::ImuSensor, 395
 - ~ImuSensor, 396
 - Fini, 396
 - GetAngularVelocity, 396
 - GetImuMessage, 397
 - GetLinearAcceleration, 397
 - GetOrientation, 397
 - ImuSensor, 396
 - Init, 397
 - IsActive, 397
 - Load, 397, 398
 - SetReferencePose, 398
 - UpdateImpl, 398
- gazebo::sensors::MultiCameraSensor, 573
 - ~MultiCameraSensor, 575
 - Fini, 575
 - GetCamera, 575
 - GetCameraCount, 575
 - GetImageData, 576
 - GetImageHeight, 576
 - GetImageWidth, 576
 - GetTopic, 577
 - Init, 577
 - IsActive, 577
 - Load, 577
 - MultiCameraSensor, 575
 - SaveFrame, 577
 - UpdateImpl, 578
- gazebo::sensors::Noise, 603
 - ~Noise, 604
 - Apply, 604
 - GAUSSIAN, 604
 - GAUSSIAN_QUANTIZED, 604
 - GetBias, 605
 - GetMean, 605
 - GetNoiseType, 605
 - GetStdDev, 605
 - Load, 605
 - NONE, 604
 - Noise, 604
 - NoiseType, 604
- gazebo::sensors::RFIDSensor, 710
 - ~RFIDSensor, 711
 - AddTag, 711
 - Fini, 711
 - Init, 711
 - Load, 711, 712
 - RFIDSensor, 711
 - UpdateImpl, 712
- gazebo::sensors::RFIDTag, 712
 - ~RFIDTag, 714
 - Fini, 714
 - GetTagPose, 714
 - Init, 714
 - Load, 714
 - RFIDTag, 714
 - UpdateImpl, 714
- gazebo::sensors::RaySensor, 693
 - ~RaySensor, 695
 - Fini, 695
 - GetAngleMax, 695
 - GetAngleMin, 695
 - GetAngleResolution, 695
 - GetFiducial, 695
 - GetLaserShape, 696
 - GetRange, 696
 - GetRangeCount, 696
 - GetRangeMax, 697
 - GetRangeMin, 697
 - GetRangeResolution, 697
 - GetRanges, 697
 - GetRayCount, 697
 - GetRetro, 697
 - GetTopic, 698
 - GetVerticalAngleMax, 698
 - GetVerticalAngleMin, 698
 - GetVerticalRangeCount, 698
 - GetVerticalRayCount, 698
 - Init, 699
 - IsActive, 699
 - Load, 699
 - RaySensor, 695
 - UpdateImpl, 699
- gazebo::sensors::Sensor, 751
 - ~Sensor, 754
 - active, 761
 - ConnectUpdated, 755
 - connections, 761
 - DisconnectUpdated, 755
 - FillMsg, 755
 - Fini, 755
 - GetCategory, 756
 - GetId, 756
 - GetLastMeasurementTime, 756
 - GetLastUpdateTime, 756
 - GetName, 756

- GetParentId, 756
- GetParentName, 757
- GetPose, 757
- GetScopedName, 757
- GetTopic, 757
- GetType, 757
- GetUpdateRate, 758
- GetVisualize, 758
- GetWorldName, 758
- Init, 758
- IsActive, 758
- lastMeasurementTime, 761
- lastUpdateTime, 761
- Load, 759
- mutexLastUpdateTime, 761
- node, 761
- parentId, 761
- parentName, 761
- plugins, 761
- pose, 761
- poseSub, 761
- ResetLastUpdateTime, 759
- scene, 762
- sdf, 762
- Sensor, 754
- SetActive, 759
- SetParent, 759, 760
- SetUpdateRate, 760
- Update, 760
- UpdateImpl, 760
- updatePeriod, 762
- world, 762
- gazebo::sensors::SensorFactory, 762
 - GetSensorTypes, 763
 - NewSensor, 763
 - RegisterAll, 763
 - RegisterSensor, 763
- gazebo::sensors::SensorManager, 764
 - CreateSensor, 765
 - Fini, 765
 - GetSensor, 766
 - GetSensorTypes, 766
 - GetSensors, 766
 - Init, 766
 - RemoveSensor, 766
 - RemoveSensors, 766
 - ResetLastUpdateTimes, 766
 - RunThreads, 767
 - SensorsInitialized, 767
 - Stop, 767
 - Update, 767
- gazebo::sensors::SonarSensor, 890
 - ~SonarSensor, 892
 - ConnectUpdate, 892
 - DisconnectUpdate, 892
 - Fini, 893
 - GetRadius, 893
 - GetRange, 893
 - GetRangeMax, 893
 - GetRangeMin, 893
 - GetTopic, 893
 - Init, 894
 - IsActive, 894
 - Load, 894
 - SonarSensor, 892
 - update, 895
 - UpdateImpl, 894
- gazebo::sensors::WirelessReceiver, 1060
 - ~WirelessReceiver, 1062
 - Fini, 1062
 - GetMaxFreqFiltered, 1062
 - GetMinFreqFiltered, 1062
 - GetSensitivity, 1062
 - Init, 1062
 - Load, 1063
 - WirelessReceiver, 1062
- gazebo::sensors::WirelessTransceiver, 1063
 - ~WirelessTransceiver, 1064
 - Fini, 1065
 - gain, 1066
 - GetGain, 1065
 - GetPower, 1065
 - GetTopic, 1065
 - Init, 1065
 - Load, 1065
 - parentEntity, 1066
 - power, 1066
 - pub, 1066
 - referencePose, 1066
 - WirelessTransceiver, 1064
- gazebo::sensors::WirelessTransmitter, 1066
 - ~WirelessTransmitter, 1068
 - freq, 1069
 - GetESSID, 1068
 - GetFreq, 1068
 - GetSignalStrength, 1068
 - Init, 1069
 - Load, 1069
 - ModelStdDesv, 1069
 - NEmpty, 1069
 - NObstacle, 1070
 - UpdateImpl, 1069
 - WirelessTransmitter, 1068
- gazebo::transport, 119
 - ConnectionPtr, 121
 - MessagePtr, 121
 - NodePtr, 121
 - PublicationPtr, 121

- PublicationTransportPtr, 121
- PublisherPtr, 121
- SubscriberPtr, 121
- SubscriptionTransportPtr, 121
- gazebo::transport::CallbackHelper, 173
 - ~CallbackHelper, 175
 - CallbackHelper, 175
 - GetId, 175
 - GetLatching, 175
 - GetMsgType, 175
 - HandleData, 175
 - HandleMessage, 176
 - IsLocal, 176
 - latching, 176
- gazebo::transport::CallbackHelperT
 - CallbackHelperT, 177
 - GetMsgType, 178
 - HandleData, 178
 - HandleMessage, 178
 - IsLocal, 179
- gazebo::transport::CallbackHelperT< M >, 177
- gazebo::transport::Connection, 240
 - ~Connection, 243
 - AcceptCallback, 243
 - AsyncRead, 243
 - Cancel, 243
 - Connect, 243
 - ConnectToShutdown, 243
 - Connection, 243
 - DisconnectShutdown, 244
 - EnqueueMsg, 244
 - GetIPWhiteList, 244
 - GetId, 244
 - GetLocalAddress, 245
 - GetLocalHostname, 245
 - GetLocalPort, 245
 - GetLocalURI, 245
 - GetRemoteAddress, 245
 - GetRemoteHostname, 245
 - GetRemotePort, 246
 - GetRemoteURI, 246
 - IsOpen, 246
 - Listen, 246
 - ProcessWriteQueue, 246
 - Read, 246
 - ReadCallback, 243
 - Shutdown, 247
 - StartRead, 247
 - StopRead, 247
 - ValidateIP, 247
- gazebo::transport::ConnectionManager, 247
 - Advertise, 249
 - ConnectToRemoteHost, 249
 - eventConnections, 252
 - Fini, 249
 - GetAllPublishers, 249
 - GetTopicNamespaces, 250
 - Init, 250
 - IsRunning, 250
 - RegisterTopicNamespace, 250
 - RemoveConnection, 250
 - Run, 250
 - Stop, 251
 - Subscribe, 251
 - TriggerUpdate, 251
 - Unadvertise, 251
 - Unsubscribe, 251
- gazebo::transport::IOManager, 409
 - ~IOManager, 410
 - DecCount, 410
 - GetCount, 410
 - GetIO, 410
 - IOManager, 410
 - IncCount, 410
 - Stop, 410
- gazebo::transport::Node, 587
 - ~Node, 588
 - Advertise, 589
 - DecodeTopicName, 589
 - EncodeTopicName, 589
 - Fini, 589
 - GetId, 589
 - GetMsgType, 590
 - GetTopicNamespace, 590
 - HandleData, 590
 - HandleMessage, 590
 - HasLatchedSubscriber, 591
 - Init, 591
 - InsertLatchedMsg, 591
 - Node, 588
 - ProcessIncoming, 591
 - ProcessPublishers, 591
 - Publish, 592
 - RemoveCallback, 592
 - Subscribe, 592, 593
- gazebo::transport::Publication, 664
 - ~Publication, 665
 - AddPublisher, 666
 - AddSubscription, 666
 - AddTransport, 666
 - GetCallbackCount, 666
 - GetLocallyAdvertised, 666
 - GetMsgType, 667
 - GetNodeCount, 667
 - GetRemoteSubscriptionCount, 667
 - GetTransportCount, 667
 - HasTransport, 667
 - LocalPublish, 668

- Publication, 665
- Publish, 668
- RemoveSubscription, 668
- RemoveTransport, 668
- SetLocallyAdvertised, 668
- gazebo::transport::PublicationTransport, 669
 - ~PublicationTransport, 670
 - AddCallback, 670
 - Finis, 670
 - GetConnection, 670
 - GetMsgType, 670
 - GetTopic, 670
 - Init, 670
 - PublicationTransport, 669
- gazebo::transport::Publisher, 671
 - ~Publisher, 672
 - GetMsgType, 672
 - GetOutgoingCount, 672
 - GetPrevMsg, 672
 - GetPrevMsgPtr, 673
 - GetTopic, 673
 - HasConnections, 673
 - Publish, 673
 - Publisher, 672
 - SendMessage, 673
 - SetNode, 674
 - SetPublication, 674
 - WaitForConnection, 674
- gazebo::transport::RawCallbackHelper, 690
 - GetMsgType, 692
 - HandleData, 692
 - HandleMessage, 692
 - IsLocal, 692
 - RawCallbackHelper, 691
- gazebo::transport::SubscribeOptions, 927
 - GetLatching, 928
 - GetMsgType, 928
 - GetNode, 928
 - GetTopic, 928
 - Init, 928, 929
 - SubscribeOptions, 928
- gazebo::transport::Subscriber, 929
 - ~Subscriber, 930
 - GetCallbackId, 930
 - GetTopic, 930
 - SetCallbackId, 930
 - Subscriber, 929
 - Unsubscribe, 930
- gazebo::transport::SubscriptionTransport, 930
 - ~SubscriptionTransport, 931
 - GetConnection, 932
 - HandleData, 932
 - HandleMessage, 932
 - Init, 932
 - IsLocal, 933
 - SubscriptionTransport, 931
- gazebo::transport::TopicManager, 967
 - AddNode, 969
 - AddNodeToProcess, 970
 - Advertise, 970
 - ClearBuffers, 970
 - ConnectPubToSub, 970
 - ConnectSubToPub, 970
 - ConnectSubscribers, 970
 - DisconnectPubFromSub, 971
 - DisconnectSubFromPub, 971
 - FindPublication, 971
 - Finis, 971
 - GetTopicNamespaces, 971
 - Init, 972
 - IsAdvertised, 972
 - PauseIncoming, 972
 - ProcessNodes, 972
 - Publish, 972
 - RegisterTopicNamespace, 973
 - RemoveNode, 973
 - SubNodeMap, 969
 - Subscribe, 973
 - Unadvertise, 973
 - Unsubscribe, 973
 - UpdatePublications, 974
- gazebo::util, 121
 - DiagnosticTimerPtr, 122
 - OpenALSinkPtr, 122
 - OpenALSourcePtr, 122
- gazebo::util::DiagnosticManager, 279
 - GetLabel, 281
 - GetLogPath, 281
 - GetTime, 281
 - GetTimerCount, 281
 - Init, 282
 - Lap, 282
 - StartTimer, 282
 - StopTimer, 282
- gazebo::util::DiagnosticTimer, 283
 - ~DiagnosticTimer, 284
 - DiagnosticTimer, 283
 - GetName, 284
 - Lap, 284
 - Start, 284
 - Stop, 284
- gazebo::util::LogPlay, 483
 - GetChunk, 484
 - GetChunkCount, 484
 - GetEncoding, 484
 - GetGazeboVersion, 484
 - GetHeader, 485
 - GetLogVersion, 485

- GetRandSeed, 485
- IsOpen, 485
- Open, 485
- Step, 486
- gazebo::util::LogRecord, 486
 - Add, 488
 - Fini, 489
 - GetBasePath, 489
 - GetBufferSize, 489
 - GetEncoding, 489
 - GetFileSize, 489
 - GetFilename, 489
 - GetFirstUpdate, 490
 - GetPaused, 490
 - GetRunTime, 490
 - GetRunning, 490
 - Init, 490
 - IsReadyToStart, 491
 - Notify, 491
 - Remove, 491
 - SetBasePath, 491
 - SetPaused, 491
 - Start, 492
 - Stop, 492
 - Write, 492
- gazebo::util::OpenAL, 609
 - CreateSink, 610
 - CreateSource, 610
 - Fini, 611
 - Load, 611
- gazebo::util::OpenALSink, 611
 - ~OpenALSink, 612
 - OpenALSink, 612
 - SetPose, 612
 - SetVelocity, 612
- gazebo::util::OpenALSource, 612
 - ~OpenALSource, 613
 - FillBufferFromFile, 614
 - FillBufferFromPCM, 614
 - GetCollisionNames, 614
 - GetOnContact, 614
 - HasCollisionName, 614
 - IsPlaying, 615
 - Load, 615
 - OpenALSource, 613
 - Pause, 615
 - Play, 615
 - Rewind, 615
 - SetGain, 615
 - SetLoop, 616
 - SetPitch, 616
 - SetPose, 616
 - SetVelocity, 616
 - Stop, 617
- gazebo_core.hh, 1145
- GazeboGenerator
 - google::protobuf::compiler::cpp::GazeboGenerator, 341
- GazeboGenerator.hh, 1146
- gazeboPathsFromEnv
 - gazebo::common::SystemPaths, 942
- GenSphericalTexCoord
 - gazebo::common::Mesh, 522
 - gazebo::common::MeshManager, 532
 - gazebo::common::SubMesh, 921
- Generate
 - google::protobuf::compiler::cpp::GazeboGenerator, 341
- generate
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 890
- generateForCompositeMap
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 890
- generateFpDynamicShadows
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 774
- generateFpDynamicShadowsHelpers
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 774
- generateFpDynamicShadowsParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 774
- generateFpFooter
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 774
- generateFpHeader
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 774
- generateFpLayer
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 774
- generateFragmentProgram
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperCg, 772
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 774
- generateFragmentProgramSource
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 775
- GenerateShaders
 - gazebo::rendering::RTShaderSystem, 727
- generateVertexProgram
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperCg, 772
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 775
- generateVertexProgramSource

- gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 772
- gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 775
- generateVpDynamicShadows
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 772
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 775
- generateVpDynamicShadowsParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 772
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 775
- generateVpFooter
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 772
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 775
- generateVpHeader
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 772
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 775
- GeneratorType
 - gazebo::math, 102
- GeometryFromSDF
 - Messages, 58
- Get
 - gazebo::common::NodeTransform, 600
- get_master_uri
 - Transport, 87
- get_scene
 - Rendering, 77
- get_sensor
 - Sensors, 82
- get_topic_namespaces
 - Transport, 87
- get_world
 - Classes for physics and dynamics, 67
- GetAABB
 - gazebo::common::Mesh, 522
 - gazebo::rendering::MovableText, 569
- GetAbs
 - gazebo::math::Vector3, 1009
- GetAcceleration
 - gazebo::physics::LinkState, 479
- getAdvertisedTopics
 - Transport, 88
- GetAllPublishers
 - gazebo::transport::ConnectionManager, 249
- GetAltitude
 - gazebo::sensors::GpsSensor, 342
- GetAmbient
 - gazebo::common::Material, 501
- GetAmbientColor
 - gazebo::rendering::Scene, 735
- GetAnchor
 - gazebo::physics::Joint, 417
 - gazebo::physics::ScrewJoint, 747
 - gazebo::physics::SimbodyBallJoint, 781
 - gazebo::physics::SimbodyHinge2Joint, 795
 - gazebo::physics::SimbodyJoint, 807
 - gazebo::physics::SimbodyUniversalJoint, 860
 - gazebo::physics::SliderJoint, 887
- GetAngle
 - gazebo::physics::Joint, 417
 - gazebo::physics::JointState, 438
- GetAngleCount
 - gazebo::physics::BallJoint, 152
 - gazebo::physics::Hinge2Joint, 387
 - gazebo::physics::HingeJoint, 389
 - gazebo::physics::Joint, 417
 - gazebo::physics::JointState, 438
 - gazebo::physics::ScrewJoint, 748
 - gazebo::physics::SliderJoint, 888
 - gazebo::physics::UniversalJoint, 978
- GetAngleImpl
 - gazebo::physics::Joint, 418
 - gazebo::physics::SimbodyBallJoint, 781
 - gazebo::physics::SimbodyHinge2Joint, 795
 - gazebo::physics::SimbodyHingeJoint, 801
 - gazebo::physics::SimbodyScrewJoint, 845
 - gazebo::physics::SimbodySliderJoint, 852
 - gazebo::physics::SimbodyUniversalJoint, 860
- GetAngleMax
 - gazebo::sensors::GpuRaySensor, 357
 - gazebo::sensors::RaySensor, 695
- GetAngleMin
 - gazebo::sensors::GpuRaySensor, 357
 - gazebo::sensors::RaySensor, 695
- GetAngleResolution
 - gazebo::sensors::GpuRaySensor, 357
 - gazebo::sensors::RaySensor, 695
- GetAngles
 - gazebo::physics::JointState, 439
- GetAngularDamping
 - gazebo::physics::Link, 463
- GetAngularVelocity
 - gazebo::sensors::ImuSensor, 396
- GetAnimation
 - gazebo::common::Skeleton, 869
- GetAsABGR
 - gazebo::common::Color, 230
- GetAsARGB
 - gazebo::common::Color, 230
- GetAsAxis
 - gazebo::math::Quaternion, 679

- GetAsBGRA
 - gazebo::common::Color, 230
- GetAsEuler
 - gazebo::math::Quaternion, 680
- GetAsHSV
 - gazebo::common::Color, 230
- GetAsMatrix3
 - gazebo::math::Quaternion, 680
- GetAsMatrix4
 - gazebo::math::Quaternion, 680
- GetAsPose
 - gazebo::math::Matrix4, 514
- GetAsRGBA
 - gazebo::common::Color, 230
- GetAsYUV
 - gazebo::common::Color, 230
- GetAspectRatio
 - gazebo::rendering::Camera, 188
- GetAttachedObjectCount
 - gazebo::rendering::Visual, 1043
- GetAttribute
 - gazebo::physics::Joint, 418
 - gazebo::physics::SimbodyJoint, 808
- GetAutoDisable
 - gazebo::physics::Model, 542
- GetAutoDisableFlag
 - gazebo::physics::PhysicsEngine, 626
- GetAvgColor
 - gazebo::common::Image, 392
- GetAvgFPS
 - gazebo::rendering::Camera, 188
 - gazebo::rendering::UserCamera, 982
 - gazebo::rendering::WindowManager, 1058
- GetAvgHeight
 - gazebo::rendering::Heightmap, 377
- GetAxis
 - gazebo::physics::SimbodyBallJoint, 781
 - gazebo::physics::SimbodyHinge2Joint, 795
 - gazebo::physics::SimbodyUniversalJoint, 860
- GetBPP
 - gazebo::common::Image, 392
- GetBackgroundColor
 - gazebo::rendering::Scene, 735
- GetBasePath
 - gazebo::util::LogRecord, 489
- GetBaseline
 - gazebo::rendering::MovableText, 569
- GetBias
 - gazebo::sensors::Noise, 605
- GetBindShapeTransform
 - gazebo::common::Skeleton, 869
- GetBlendFactors
 - gazebo::common::Material, 501
- GetBlendMode
 - gazebo::common::Material, 501
- GetBoundingBox
 - gazebo::physics::Collision, 216
 - gazebo::physics::Entity, 297
 - gazebo::physics::Link, 463
 - gazebo::physics::Model, 542
 - gazebo::physics::SimbodyCollision, 787
 - gazebo::rendering::Visual, 1043
- getBoundingBoxRadius
 - gazebo::rendering::DynamicRenderable, 291
 - gazebo::rendering::MovableText, 569
- GetBufferSize
 - gazebo::util::LogRecord, 489
- GetByName
 - gazebo::physics::Base, 158
 - gazebo::physics::World, 1074
- GetCallbackCount
 - gazebo::transport::Publication, 666
- GetCallbackId
 - gazebo::transport::Subscriber, 930
- GetCamera
 - gazebo::rendering::Scene, 735
 - gazebo::sensors::CameraSensor, 208
 - gazebo::sensors::MultiCameraSensor, 575
- GetCameraCount
 - gazebo::rendering::GpuLaser, 348
 - gazebo::rendering::Scene, 736
 - gazebo::sensors::GpuRaySensor, 357
 - gazebo::sensors::MultiCameraSensor, 575
- GetCameraToViewportRay
 - gazebo::rendering::Camera, 189
- GetCaptureData
 - gazebo::rendering::Camera, 189
- GetCategory
 - gazebo::sensors::Sensor, 756
- GetCellCount
 - gazebo::rendering::Grid, 366
- GetCellLength
 - gazebo::rendering::Grid, 366
- GetCenter
 - gazebo::math::Box, 166
- GetCharHeight
 - gazebo::rendering::MovableText, 569
- GetChild
 - gazebo::common::SkeletonNode, 880
 - gazebo::physics::Base, 158
 - gazebo::physics::Joint, 418
 - gazebo::rendering::Visual, 1043
- GetChildById
 - gazebo::common::SkeletonNode, 880
- GetChildByName
 - gazebo::common::SkeletonNode, 881
- GetChildCollision
 - gazebo::physics::Entity, 297

- GetChildCount
 - gazebo::common::SkeletonNode, 881
 - gazebo::physics::Base, 158
 - gazebo::rendering::Visual, 1043
- GetChildJoints
 - gazebo::physics::Link, 463
- GetChildJointsLinks
 - gazebo::physics::Link, 463
- GetChildLink
 - gazebo::physics::Entity, 297
- GetChunk
 - gazebo::util::LogPlay, 484
- GetChunkCount
 - gazebo::util::LogPlay, 484
- GetCmd
 - gazebo::common::PID, 637
- GetCoG
 - gazebo::physics::Inertial, 401
- GetCollision
 - gazebo::physics::Link, 463, 464
- GetCollisionBoundingBox
 - gazebo::physics::Entity, 297
- GetCollisionContactCount
 - gazebo::sensors::ContactSensor, 262
- GetCollisionCount
 - gazebo::sensors::ContactSensor, 262
- GetCollisionName
 - gazebo::sensors::ContactSensor, 263
- GetCollisionNames
 - gazebo::util::OpenALSource, 614
- GetCollisionShape
 - gazebo::physics::SimbodyCollision, 787
- GetCollisionState
 - gazebo::physics::LinkState, 479
- GetCollisionStateCount
 - gazebo::physics::LinkState, 479
- GetCollisionStates
 - gazebo::physics::LinkState, 480
- GetCollisions
 - gazebo::physics::Link, 464
- GetColor
 - gazebo::rendering::Grid, 367
 - gazebo::rendering::MovableText, 569
- GetConnection
 - gazebo::transport::PublicationTransport, 670
 - gazebo::transport::SubscriptionTransport, 932
- GetContact
 - gazebo::physics::ContactManager, 258
- GetContactCount
 - gazebo::physics::ContactManager, 258
- GetContactManager
 - gazebo::physics::PhysicsEngine, 626
- GetContactMaxCorrectingVel
 - gazebo::physics::PhysicsEngine, 626
- GetContactSurfaceLayer
 - gazebo::physics::PhysicsEngine, 626
- GetContacts
 - gazebo::physics::ContactManager, 258
 - gazebo::sensors::ContactSensor, 263
- GetContactsEnabled
 - gazebo::physics::Collision, 216
- GetCosHorzFOV
 - gazebo::rendering::GpuLaser, 348
 - gazebo::sensors::GpuRaySensor, 358
- GetCosVertFOV
 - gazebo::rendering::GpuLaser, 348
 - gazebo::sensors::GpuRaySensor, 358
- GetCount
 - gazebo::transport::IOManager, 410
- GetCurrentDir
 - SystemPaths.hh, 1288
- GetDBConfig
 - Common, 38
- GetDamping
 - gazebo::physics::Joint, 418
- GetDampingCoefficient
 - gazebo::physics::Joint, 419
- GetData
 - gazebo::common::Image, 392
- GetDblNormal
 - gazebo::math::Rand, 689
- GetDblUniform
 - gazebo::math::Rand, 689
- GetDepthCamera
 - gazebo::sensors::DepthCameraSensor, 278
- GetDepthData
 - gazebo::rendering::DepthCamera, 275
- GetDepthWrite
 - gazebo::common::Material, 502
- GetDiffuse
 - gazebo::common::Material, 502
- GetDiffuseColor
 - gazebo::rendering::Light, 450
- GetDirection
 - gazebo::rendering::Camera, 189
 - gazebo::rendering::Light, 451
- GetDirtyPose
 - gazebo::physics::Entity, 298
- GetDistToLine
 - gazebo::math::Vector3, 1009
- GetDynamicsWorld
 - gazebo::physics::SimbodyPhysics, 832
- GetESSID
 - gazebo::sensors::WirelessTransmitter, 1068
- GetEffectiveMassProps
 - gazebo::physics::SimbodyLink, 818
- GetEffortLimit
 - gazebo::physics::Joint, 419

- GetElapsed
 - gazebo::common::Timer, 967
- GetElevationReference
 - gazebo::common::SphericalCoordinates, 903
- GetEmissive
 - gazebo::common::Material, 502
- GetEnablePhysicsEngine
 - gazebo::physics::World, 1074
- GetEnabled
 - gazebo::physics::Link, 464
 - gazebo::physics::SimbodyLink, 818
- GetEncoding
 - gazebo::util::LogPlay, 484
 - gazebo::util::LogRecord, 489
- GetEntity
 - gazebo::physics::World, 1074
- GetEntityBelowPoint
 - gazebo::physics::World, 1074
- GetErrorFile
 - gazebo::common::Exception, 332
- GetErrorStr
 - gazebo::common::Exception, 333
- GetErrors
 - gazebo::common::PID, 637
- GetEulerRotation
 - gazebo::math::Matrix4, 515
- GetExp
 - gazebo::math::Quaternion, 680
- GetFarClip
 - gazebo::rendering::Camera, 189
 - gazebo::rendering::GpuLaser, 348
- GetFiducial
 - gazebo::physics::MultiRayShape, 582
 - gazebo::physics::RayShape, 702
 - gazebo::sensors::GpuRaySensor, 358
 - gazebo::sensors::RaySensor, 695
- GetFile
 - gazebo::common::AudioDecoder, 148
- GetFileSize
 - gazebo::util::LogRecord, 489
- GetFilename
 - gazebo::common::Image, 392
 - gazebo::PluginT, 648
 - gazebo::util::LogRecord, 489
- GetFirstContact
 - gazebo::rendering::Scene, 736
- GetFirstUpdate
 - gazebo::util::LogRecord, 490
- GetFocalPoint
 - gazebo::rendering::OrbitViewController, 618
- GetFont
 - gazebo::rendering::MovableText, 569
- GetForce
 - gazebo::physics::Joint, 419
 - gazebo::physics::SimbodyJoint, 808
- GetForceTorque
 - gazebo::physics::Joint, 419
 - gazebo::physics::SimbodyJoint, 808
- GetForceTorqueWithAppliedForce
 - Joint_TEST, 432
- GetFrameAt
 - gazebo::common::NodeAnimation, 595
- GetFrameCount
 - gazebo::common::NodeAnimation, 595
- GetFrameFilename
 - gazebo::rendering::Camera, 189
- GetFreq
 - gazebo::sensors::WirelessTransmitter, 1068
- GetGUIOverlay
 - gazebo::rendering::UserCamera, 982
- GetGain
 - gazebo::sensors::WirelessTransceiver, 1065
- GetGazeboPaths
 - gazebo::common::SystemPaths, 941
- GetGazeboVersion
 - gazebo::util::LogPlay, 484
- GetGlobalAxis
 - gazebo::physics::Joint, 420
 - gazebo::physics::SimbodyBallJoint, 781
 - gazebo::physics::SimbodyHinge2Joint, 795
 - gazebo::physics::SimbodyHingeJoint, 801
 - gazebo::physics::SimbodyScrewJoint, 845
 - gazebo::physics::SimbodySliderJoint, 852
 - gazebo::physics::SimbodyUniversalJoint, 860
- GetGlobalPoints
 - gazebo::physics::RayShape, 702
- GetGranularity
 - gazebo::physics::MapShape, 494
- GetGravity
 - gazebo::physics::PhysicsEngine, 626
- GetGravityMode
 - gazebo::physics::Link, 464
 - gazebo::physics::SimbodyLink, 818
- GetGrid
 - gazebo::rendering::Scene, 736
- GetGridCount
 - gazebo::rendering::Scene, 736
- GetGripper
 - gazebo::physics::Model, 542
- GetGripperCount
 - gazebo::physics::Model, 543
- GetHFOV
 - gazebo::rendering::Camera, 189
- GetHandle
 - gazebo::common::SkeletonNode, 881
 - gazebo::PluginT, 648
- GetHeader

- gazebo::util::LogPlay, 485
- Messages, 58
- GetHeadingOffset
 - gazebo::common::SphericalCoordinates, 904
- GetHeight
 - gazebo::common::Image, 392
 - gazebo::common::Video, 1028
 - gazebo::physics::HeightmapShape, 383
 - gazebo::physics::MapShape, 494
 - gazebo::rendering::Grid, 367
 - gazebo::rendering::Heightmap, 377
- GetHeightBelowPoint
 - gazebo::rendering::Scene, 736
- GetHeightmap
 - gazebo::rendering::Scene, 737
- GetHighStop
 - gazebo::physics::BallJoint, 152
 - gazebo::physics::Joint, 420
 - gazebo::physics::SimbodyHinge2Joint, 795
 - gazebo::physics::SimbodyHingeJoint, 801
 - gazebo::physics::SimbodyScrewJoint, 845
 - gazebo::physics::SimbodySliderJoint, 852
 - gazebo::physics::SimbodyUniversalJoint, 860
- GetHorzFOV
 - gazebo::rendering::GpuLaser, 348
 - gazebo::sensors::GpuRaySensor, 358
- GetHorzHalfAngle
 - gazebo::rendering::GpuLaser, 348
 - gazebo::sensors::GpuRaySensor, 358
- GetIO
 - gazebo::transport::IOManager, 410
- GetIPWhiteList
 - gazebo::transport::Connection, 244
- GetIXX
 - gazebo::physics::Inertial, 401
- GetIXY
 - gazebo::physics::Inertial, 402
- GetIXZ
 - gazebo::physics::Inertial, 402
- GetIYY
 - gazebo::physics::Inertial, 402
- GetIYZ
 - gazebo::physics::Inertial, 402
- GetIZZ
 - gazebo::physics::Inertial, 402
- GetId
 - gazebo::common::SkeletonNode, 881
 - gazebo::event::Connection, 240
 - gazebo::physics::Base, 158
 - gazebo::rendering::Scene, 737
 - gazebo::rendering::Visual, 1043
 - gazebo::sensors::Sensor, 756
 - gazebo::transport::CallbackHelper, 175
 - gazebo::transport::Connection, 244
 - gazebo::transport::Node, 589
- GetIdString
 - gazebo::rendering::Scene, 737
- GetImage
 - gazebo::physics::HeightmapShape, 383
 - gazebo::rendering::Heightmap, 378
- GetImageByteSize
 - gazebo::rendering::Camera, 190
- GetImageData
 - gazebo::rendering::Camera, 190
 - gazebo::sensors::CameraSensor, 208
 - gazebo::sensors::MultiCameraSensor, 576
- GetImageDepth
 - gazebo::rendering::Camera, 190
- GetImageFormat
 - gazebo::rendering::Camera, 191
- GetImageHeight
 - gazebo::rendering::Camera, 191
 - gazebo::rendering::UserCamera, 983
 - gazebo::sensors::CameraSensor, 208
 - gazebo::sensors::MultiCameraSensor, 576
- GetImageWidth
 - gazebo::rendering::Camera, 191
 - gazebo::rendering::UserCamera, 983
 - gazebo::sensors::CameraSensor, 208
 - gazebo::sensors::MultiCameraSensor, 576
- GetImuMessage
 - gazebo::sensors::ImuSensor, 397
- GetIndex
 - gazebo::common::SubMesh, 921
- GetIndexCount
 - gazebo::common::Mesh, 522
 - gazebo::common::SubMesh, 922
- GetInertiaRatio
 - gazebo::physics::Joint, 421
- GetInertial
 - gazebo::physics::Inertial, 401
 - gazebo::physics::Link, 464
- GetInitialRelativePose
 - gazebo::physics::Entity, 298
- GetInitialized
 - gazebo::rendering::Camera, 191
 - gazebo::rendering::Scene, 737
 - gazebo::Server, 770
- GetIntNormal
 - gazebo::math::Rand, 689
- GetIntUniform
 - gazebo::math::Rand, 689
- GetInterpolatedKeyFrame
 - gazebo::common::NumericAnimation, 607
 - gazebo::common::PoseAnimation, 659
- GetIntersection
 - gazebo::physics::RayShape, 703
 - gazebo::physics::SimbodyRayShape, 841

- GetInverse
 - gazebo::math::Pose, 653
 - gazebo::math::Quaternion, 680
- GetInverseBindTransform
 - gazebo::common::SkeletonNode, 881
- GetJoint
 - gazebo::physics::Model, 543
- GetJointController
 - gazebo::physics::Model, 543
- GetJointCount
 - gazebo::physics::Model, 543
- GetJointLink
 - gazebo::physics::Joint, 421
 - gazebo::physics::SimbodyJoint, 809
- GetJointState
 - gazebo::physics::ModelState, 557, 558
- GetJointStateCount
 - gazebo::physics::ModelState, 558
- GetJointStates
 - gazebo::physics::ModelState, 558, 559
- GetJoints
 - gazebo::physics::Model, 543
- GetKeyFrame
 - gazebo::common::Animation, 141
 - gazebo::common::NodeAnimation, 596
- GetKeyFrameCount
 - gazebo::common::Animation, 141
- GetKeyFramesAtTime
 - gazebo::common::Animation, 142
- GetKinematic
 - gazebo::physics::Link, 465
- GetLabel
 - gazebo::util::DiagnosticManager, 281
- GetLaserCamera
 - gazebo::sensors::GpuRaySensor, 359
- GetLaserData
 - gazebo::rendering::GpuLaser, 349
- GetLaserRetro
 - gazebo::physics::Collision, 217
- GetLaserShape
 - gazebo::sensors::RaySensor, 696
- GetLastMeasurementTime
 - gazebo::sensors::Sensor, 756
- GetLastRenderWallTime
 - gazebo::rendering::Camera, 191
- GetLastUpdateTime
 - gazebo::sensors::Sensor, 756
- GetLatching
 - gazebo::transport::CallbackHelper, 175
 - gazebo::transport::SubscribeOptions, 928
- GetLatitude
 - gazebo::sensors::GpsSensor, 343
- GetLatitudeReference
 - gazebo::common::SphericalCoordinates, 904
- GetLength
 - gazebo::common::Animation, 142
 - gazebo::common::NodeAnimation, 596
 - gazebo::common::SkeletonAnimation, 874
 - gazebo::math::Vector3, 1010
 - gazebo::math::Vector4, 1021
 - gazebo::physics::CylinderShape, 270
 - gazebo::physics::RayShape, 703
- GetLight
 - gazebo::rendering::Scene, 737, 738
- GetLightCount
 - gazebo::rendering::Scene, 738
- GetLighting
 - gazebo::common::Material, 502
- getLights
 - gazebo::rendering::MovableText, 569
- GetLineWidth
 - gazebo::rendering::Grid, 367
- GetLinearAcceleration
 - gazebo::sensors::ImuSensor, 397
- GetLinearDamping
 - gazebo::physics::Link, 465
- GetLink
 - gazebo::physics::Collision, 217
 - gazebo::physics::Model, 544
- GetLinkForce
 - gazebo::physics::Joint, 421
 - gazebo::physics::SimbodyJoint, 809
- GetLinkState
 - gazebo::physics::ModelState, 559
- GetLinkStateCount
 - gazebo::physics::ModelState, 559
- GetLinkStates
 - gazebo::physics::ModelState, 559, 560
- GetLinkTorque
 - gazebo::physics::Joint, 421
 - gazebo::physics::SimbodyJoint, 809
- GetLinks
 - gazebo::physics::Model, 544
- GetLocalAddress
 - gazebo::transport::Connection, 245
- GetLocalAxis
 - gazebo::physics::Joint, 422
- GetLocalHostname
 - gazebo::transport::Connection, 245
- GetLocalPort
 - gazebo::transport::Connection, 245
- GetLocalURI
 - gazebo::transport::Connection, 245
- GetLocallyAdvertised
 - gazebo::transport::Publication, 666
- GetLog
 - gazebo::math::Quaternion, 680
- GetLogPath

- gazebo::common::SystemPaths, 941
- gazebo::util::DiagnosticManager, 281
- GetLogVersion
 - gazebo::util::LogPlay, 485
- GetLongitude
 - gazebo::sensors::GpsSensor, 343
- GetLongitudeReference
 - gazebo::common::SphericalCoordinates, 904
- GetLowStop
 - gazebo::physics::BallJoint, 152
 - gazebo::physics::Joint, 422
 - gazebo::physics::SimbodyHinge2Joint, 796
 - gazebo::physics::SimbodyHingeJoint, 801
 - gazebo::physics::SimbodyScrewJoint, 845
 - gazebo::physics::SimbodySliderJoint, 852
 - gazebo::physics::SimbodyUniversalJoint, 861
- GetLowerLimit
 - gazebo::physics::Joint, 422
- GetMOI
 - gazebo::physics::Inertial, 402, 403
- GetManager
 - gazebo::rendering::Scene, 738
- GetMass
 - gazebo::physics::Inertial, 402
- GetMassProperties
 - gazebo::physics::SimbodyLink, 818
- GetMaterial
 - gazebo::common::Mesh, 522
- getMaterial
 - gazebo::rendering::MovableText, 569
- GetMaterialCount
 - gazebo::common::Mesh, 523
- GetMaterialIndex
 - gazebo::common::SubMesh, 922
- GetMaterialName
 - gazebo::rendering::Visual, 1043
- GetMax
 - gazebo::common::Mesh, 523
 - gazebo::common::SubMesh, 922
 - gazebo::math::Vector3, 1010
- GetMaxAngle
 - gazebo::physics::MultiRayShape, 583
- GetMaxColor
 - gazebo::common::Image, 393
- GetMaxContacts
 - gazebo::physics::Collision, 217
 - gazebo::physics::PhysicsEngine, 627
- GetMaxForce
 - gazebo::physics::Joint, 423
 - gazebo::physics::SimbodyBallJoint, 781
 - gazebo::physics::SimbodyHinge2Joint, 796
 - gazebo::physics::SimbodyHingeJoint, 802
 - gazebo::physics::SimbodyScrewJoint, 846
 - gazebo::physics::SimbodySliderJoint, 853
 - gazebo::physics::SimbodyUniversalJoint, 861
- GetMaxFreqFiltered
 - gazebo::sensors::WirelessReceiver, 1062
- GetMaxHeight
 - gazebo::physics::HeightmapShape, 383
- GetMaxIndex
 - gazebo::common::SubMesh, 922
- GetMaxRange
 - gazebo::physics::MultiRayShape, 583
- GetMaxStepSize
 - gazebo::physics::PhysicsEngine, 627
- GetMean
 - gazebo::sensors::Noise, 605
- GetMesh
 - gazebo::common::MeshManager, 532
- GetMeshAABB
 - gazebo::common::MeshManager, 533
- GetMeshName
 - gazebo::rendering::Visual, 1044
- GetMeshURI
 - gazebo::physics::MeshShape, 536
- GetMin
 - gazebo::common::Mesh, 523
 - gazebo::common::SubMesh, 922
 - gazebo::math::Vector3, 1010
- GetMinAngle
 - gazebo::physics::MultiRayShape, 583
- GetMinFreqFiltered
 - gazebo::sensors::WirelessReceiver, 1062
- GetMinHeight
 - gazebo::physics::HeightmapShape, 383
- GetMinRange
 - gazebo::physics::MultiRayShape, 583
- getMinimalComms
 - Transport, 88
- GetMode
 - Rendering, 77
- GetModel
 - gazebo::physics::Collision, 217
 - gazebo::physics::Link, 465
 - gazebo::physics::World, 1075
- GetModelBelowPoint
 - gazebo::physics::World, 1075
- GetModelConfig
 - Common, 38
- GetModelCount
 - gazebo::physics::World, 1075
- GetModelFile
 - Common, 38
- GetModelName
 - Common, 39
- GetModelPath
 - Common, 39
- GetModelPaths

- gazebo::common::SystemPaths, 941
- GetModelState
 - gazebo::physics::WorldState, 1085
- GetModelStateCount
 - gazebo::physics::WorldState, 1086
- GetModelStates
 - gazebo::physics::WorldState, 1086
- GetModelTransform
 - gazebo::common::SkeletonNode, 881
- GetModelVisualAt
 - gazebo::rendering::Scene, 738
- GetModels
 - Common, 39
 - gazebo::physics::World, 1076
- GetMouseHit
 - gazebo::rendering::Heightmap, 378
- GetMovableType
 - gazebo::rendering::DynamicLines, 288
 - gazebo::rendering::DynamicRenderable, 292
- getMovableType
 - gazebo::rendering::DynamicLines, 288
- GetMsgType
 - gazebo::transport::CallbackHelper, 175
 - gazebo::transport::CallbackHelperT, 178
 - gazebo::transport::Node, 590
 - gazebo::transport::Publication, 667
 - gazebo::transport::PublicationTransport, 670
 - gazebo::transport::Publisher, 672
 - gazebo::transport::RawCallbackHelper, 692
 - gazebo::transport::SubscribeOptions, 928
- GetMsgTypes
 - gazebo::msgs::MsgFactory, 573
- GetName
 - gazebo::common::Material, 502
 - gazebo::common::Mesh, 523
 - gazebo::common::NodeAnimation, 596
 - gazebo::common::SkeletonAnimation, 875
 - gazebo::common::SkeletonNode, 882
 - gazebo::common::SubMesh, 922
 - gazebo::physics::Base, 159
 - gazebo::physics::Gripper, 369
 - gazebo::physics::State, 912
 - gazebo::physics::World, 1076
 - gazebo::rendering::Camera, 191
 - gazebo::rendering::Light, 451
 - gazebo::rendering::Scene, 738
 - gazebo::rendering::Visual, 1044
 - gazebo::sensors::Sensor, 756
 - gazebo::util::DiagnosticTimer, 284
- GetNearClip
 - gazebo::rendering::Camera, 192
 - gazebo::rendering::GpuLaser, 349
- GetNearestEntityBelow
 - gazebo::physics::Entity, 298
- GetNextFrame
 - gazebo::common::Video, 1028
- GetNode
 - gazebo::transport::SubscribeOptions, 928
- GetNodeAssignment
 - gazebo::common::SubMesh, 922
- GetNodeAssignmentsCount
 - gazebo::common::SubMesh, 923
- GetNodeByHandle
 - gazebo::common::Skeleton, 869
- GetNodeById
 - gazebo::common::Skeleton, 869
- GetNodeByName
 - gazebo::common::Skeleton, 870
- GetNodeCount
 - gazebo::common::SkeletonAnimation, 875
 - gazebo::transport::Publication, 667
- GetNodePoseAt
 - gazebo::common::SkeletonAnimation, 875
- GetNodes
 - gazebo::common::Skeleton, 870
- GetNoiseType
 - gazebo::sensors::Noise, 605
- GetNormal
 - gazebo::common::SubMesh, 923
 - gazebo::math::Vector3, 1010
 - gazebo::physics::PlaneShape, 645
- GetNormalCount
 - gazebo::common::Mesh, 523
 - gazebo::common::SubMesh, 923
- GetNormalMap
 - gazebo::rendering::Visual, 1044
- GetNumAnimations
 - gazebo::common::Skeleton, 870
- GetNumJoints
 - gazebo::common::Skeleton, 870
- GetNumNodes
 - gazebo::common::Skeleton, 870
- GetNumPoints
 - gazebo::math::RotationSpline, 722
- GetNumRawTrans
 - gazebo::common::SkeletonNode, 882
- GetNumVertNodeWeights
 - gazebo::common::Skeleton, 870
- GetOgreCamera
 - gazebo::rendering::Camera, 192
- GetOgrePaths
 - gazebo::common::SystemPaths, 941
- GetOgreTerrain
 - gazebo::rendering::Heightmap, 378
- GetOnContact
 - gazebo::util::OpenALSource, 614
- GetOperationType
 - gazebo::rendering::DynamicRenderable, 292

- GetOrientation
 - gazebo::sensors::ImuSensor, 397
- GetOutgoingCount
 - gazebo::transport::Publisher, 672
- GetPSSMShadowCameraSetup
 - gazebo::rendering::RTShaderSystem, 727
- GetParam
 - gazebo::physics::PhysicsEngine, 627
- GetParent
 - gazebo::common::SkeletonNode, 882
 - gazebo::physics::Base, 159
 - gazebo::physics::Joint, 423
 - gazebo::rendering::Projector, 663
 - gazebo::rendering::Visual, 1044
- GetParentId
 - gazebo::physics::Base, 159
 - gazebo::sensors::Sensor, 756
- GetParentJoints
 - gazebo::physics::Link, 465
- GetParentJointsLinks
 - gazebo::physics::Link, 465
- GetParentModel
 - gazebo::physics::Entity, 298
- GetParentName
 - gazebo::sensors::Sensor, 757
- GetPath
 - gazebo::common::Mesh, 523
- GetPauseTime
 - gazebo::physics::World, 1076
- GetPaused
 - gazebo::util::LogRecord, 490
- GetPerpendicular
 - gazebo::math::Vector3, 1010
- GetPhysicsEngine
 - gazebo::physics::World, 1076
- GetPhysicsUpdateMutex
 - gazebo::physics::PhysicsEngine, 627
- GetPitch
 - gazebo::common::Image, 393
 - gazebo::math::Quaternion, 681
- GetPitchNode
 - gazebo::rendering::Camera, 192
- GetPixel
 - gazebo::common::Image, 393
- GetPixelFormat
 - gazebo::common::Image, 393
- GetPluginCount
 - gazebo::physics::Model, 544
- GetPluginPaths
 - gazebo::common::SystemPaths, 941
- GetPoint
 - gazebo::math::RotationSpline, 722
 - gazebo::math::Spline, 907
 - gazebo::rendering::DynamicLines, 288
- GetPointCount
 - gazebo::math::Spline, 908
 - gazebo::rendering::DynamicLines, 289
- GetPointSize
 - gazebo::common::Material, 502
- GetPos
 - gazebo::physics::HeightmapShape, 384
- GetPose
 - gazebo::physics::CollisionState, 224
 - gazebo::physics::Inertial, 403
 - gazebo::physics::LinkState, 480
 - gazebo::physics::ModelState, 560
 - gazebo::physics::SimbodyPhysics, 832
 - gazebo::rendering::Visual, 1044
 - gazebo::sensors::Sensor, 757
- GetPoseAt
 - gazebo::common::SkeletonAnimation, 875
- GetPoseAtX
 - gazebo::common::SkeletonAnimation, 876
- GetPosition
 - gazebo::rendering::Light, 451
 - gazebo::rendering::Visual, 1044
- GetPower
 - gazebo::sensors::WirelessTransceiver, 1065
- GetPrevMsg
 - gazebo::transport::Publisher, 672
- GetPrevMsgPtr
 - gazebo::transport::Publisher, 673
- GetPrimitiveType
 - gazebo::common::SubMesh, 923
- GetPrincipalMoments
 - gazebo::physics::Inertial, 403
- GetProductsofInertia
 - gazebo::physics::Inertial, 403
- GetQuiet
 - Common, 40
- GetRGBData
 - gazebo::common::Image, 393
- GetRadius
 - gazebo::physics::CylinderShape, 270
 - gazebo::physics::SphereShape, 900
 - gazebo::sensors::SonarSensor, 893
- GetRandSeed
 - gazebo::util::LogPlay, 485
- GetRange
 - gazebo::physics::MultiRayShape, 583
 - gazebo::sensors::GpuRaySensor, 359
 - gazebo::sensors::RaySensor, 696
 - gazebo::sensors::SonarSensor, 893
- GetRangeCount
 - gazebo::sensors::GpuRaySensor, 359
 - gazebo::sensors::RaySensor, 696
- GetRangeCountRatio
 - gazebo::sensors::GpuRaySensor, 359

- GetRangeMax
 - gazebo::sensors::GpuRaySensor, 359
 - gazebo::sensors::RaySensor, 697
 - gazebo::sensors::SonarSensor, 893
- GetRangeMin
 - gazebo::sensors::GpuRaySensor, 360
 - gazebo::sensors::RaySensor, 697
 - gazebo::sensors::SonarSensor, 893
- GetRangeResolution
 - gazebo::sensors::GpuRaySensor, 360
 - gazebo::sensors::RaySensor, 697
- GetRanges
 - gazebo::sensors::GpuRaySensor, 360
 - gazebo::sensors::RaySensor, 697
- GetRawTransform
 - gazebo::common::SkeletonNode, 882
- GetRawTransforms
 - gazebo::common::SkeletonNode, 882
- GetRayCount
 - gazebo::sensors::GpuRaySensor, 360
 - gazebo::sensors::RaySensor, 697
- GetRayCountRatio
 - gazebo::rendering::GpuLaser, 349
 - gazebo::sensors::GpuRaySensor, 360
- GetRealTime
 - gazebo::physics::State, 913
 - gazebo::physics::World, 1076
- GetRealTimeUpdateRate
 - gazebo::physics::PhysicsEngine, 627
- GetRelativeAngularAccel
 - gazebo::physics::Collision, 217
 - gazebo::physics::Entity, 298
 - gazebo::physics::Link, 465
 - gazebo::physics::Model, 544
- GetRelativeAngularVel
 - gazebo::physics::Collision, 217
 - gazebo::physics::Entity, 298
 - gazebo::physics::Link, 466
 - gazebo::physics::Model, 544
- GetRelativeForce
 - gazebo::physics::Link, 466
- GetRelativeLinearAccel
 - gazebo::physics::Collision, 218
 - gazebo::physics::Entity, 299
 - gazebo::physics::Link, 466
 - gazebo::physics::Model, 545
- GetRelativeLinearVel
 - gazebo::physics::Collision, 218
 - gazebo::physics::Entity, 299
 - gazebo::physics::Link, 466
 - gazebo::physics::Model, 545
- GetRelativePoints
 - gazebo::physics::RayShape, 703
- GetRelativePose
 - gazebo::physics::Entity, 299
- GetRelativeTorque
 - gazebo::physics::Link, 466
- GetRemoteAddress
 - gazebo::transport::Connection, 245
- GetRemoteHostname
 - gazebo::transport::Connection, 245
- GetRemotePort
 - gazebo::transport::Connection, 246
- GetRemoteSubscriptionCount
 - gazebo::transport::Publication, 667
- GetRemoteURI
 - gazebo::transport::Connection, 246
- getRenderOperation
 - gazebo::rendering::MovableText, 569
- GetRenderPathType
 - gazebo::rendering::RenderEngine, 708
- GetRenderRate
 - gazebo::rendering::Camera, 192
- GetRenderTexture
 - gazebo::rendering::Camera, 192
- GetResRange
 - gazebo::physics::MultiRayShape, 583
- GetRetro
 - gazebo::physics::MultiRayShape, 584
 - gazebo::physics::RayShape, 703
 - gazebo::sensors::GpuRaySensor, 360
 - gazebo::sensors::RaySensor, 697
- GetRight
 - gazebo::rendering::Camera, 192
- GetRoll
 - gazebo::math::Quaternion, 681
- GetRootNode
 - gazebo::common::Skeleton, 871
- GetRootVisual
 - gazebo::rendering::Visual, 1045
- GetRotation
 - gazebo::common::PoseKeyFrame, 661
 - gazebo::math::Matrix4, 515
 - gazebo::rendering::Visual, 1045
- GetRounded
 - gazebo::math::Vector3, 1011
- GetRunTime
 - gazebo::util::LogRecord, 490
- GetRunning
 - gazebo::common::Timer, 967
 - gazebo::physics::World, 1076
 - gazebo::util::LogRecord, 490
- GetSDF
 - gazebo::physics::Actor, 128
 - gazebo::physics::Base, 159
 - gazebo::physics::Model, 545
- GetSID
 - gazebo::common::NodeTransform, 600

- GetSORPGSIters
 - gazebo::physics::PhysicsEngine, 628
- GetSORPGSPreconIters
 - gazebo::physics::PhysicsEngine, 628
- GetSORPGSW
 - gazebo::physics::PhysicsEngine, 628
- GetSampleCount
 - gazebo::physics::MultiRayShape, 584
- GetSampleRate
 - gazebo::common::AudioDecoder, 148
- GetSaveable
 - gazebo::physics::Base, 159
- GetScale
 - gazebo::physics::MapShape, 495
 - gazebo::physics::Shape, 777
 - gazebo::rendering::Visual, 1045
- GetScanResolution
 - gazebo::physics::MultiRayShape, 584
- GetScene
 - gazebo::rendering::Camera, 193
 - gazebo::rendering::RenderEngine, 708
 - gazebo::rendering::Visual, 1045
- GetSceneCount
 - gazebo::rendering::RenderEngine, 709
- GetSceneNode
 - gazebo::rendering::Camera, 193
 - gazebo::rendering::Grid, 367
 - gazebo::rendering::Visual, 1045
- GetScopedName
 - gazebo::physics::Base, 159
 - gazebo::sensors::Sensor, 757
- GetScreenshotPath
 - gazebo::rendering::Camera, 193
- GetSeed
 - gazebo::math::Rand, 690
- GetSelectedEntity
 - gazebo::physics::World, 1077
- GetSelectedVisual
 - gazebo::rendering::Scene, 739
- GetSelfCollide
 - gazebo::physics::Link, 467
- GetSensitivity
 - gazebo::sensors::WirelessReceiver, 1062
- GetSensor
 - gazebo::sensors::SensorManager, 766
- GetSensorCount
 - gazebo::physics::Link, 467
 - gazebo::physics::Model, 545
- GetSensorName
 - gazebo::physics::Link, 467
- GetSensorTypes
 - gazebo::sensors::SensorFactory, 763
 - gazebo::sensors::SensorManager, 766
- GetSensors
 - gazebo::sensors::SensorManager, 766
- GetSetWorldPoseMutex
 - gazebo::physics::World, 1077
- GetShadeMode
 - gazebo::common::Material, 503
- GetShaderType
 - gazebo::rendering::Visual, 1045
- GetShadowsEnabled
 - gazebo::rendering::Scene, 739
- GetShape
 - gazebo::physics::Collision, 218
- GetShapeType
 - gazebo::physics::Collision, 218
- GetShininess
 - gazebo::common::Material, 503
- GetShowClouds
 - gazebo::rendering::Scene, 739
- GetShowOnTop
 - gazebo::rendering::MovableText, 570
- GetSignalStrength
 - gazebo::sensors::WirelessTransmitter, 1068
- GetSimTime
 - gazebo::physics::State, 913
 - gazebo::physics::World, 1077
 - gazebo::rendering::Scene, 739
- GetSize
 - gazebo::math::Box, 166
 - gazebo::physics::BoxShape, 171
 - gazebo::physics::HeightmapShape, 384
 - gazebo::physics::MeshShape, 536
 - gazebo::physics::PlaneShape, 645
- GetSkeleton
 - gazebo::common::Mesh, 523
- GetSpaceWidth
 - gazebo::rendering::MovableText, 570
- GetSpecular
 - gazebo::common::Material, 503
- GetSpecularColor
 - gazebo::rendering::Light, 451
- GetSphericalCoordinates
 - gazebo::physics::World, 1077
- GetSquaredLength
 - gazebo::math::Vector3, 1011
 - gazebo::math::Vector4, 1021
- getSquaredViewDepth
 - gazebo::rendering::DynamicRenderable, 292
 - gazebo::rendering::MovableText, 570
- GetStartTime
 - gazebo::physics::World, 1077
- GetState
 - gazebo::physics::Collision, 218
 - Rendering, 77
- GetStdDev
 - gazebo::sensors::Noise, 605

- GetSubMesh
 - gazebo::common::Mesh, 524
- GetSubMeshCount
 - gazebo::common::Mesh, 524
- GetSubMeshName
 - gazebo::rendering::Visual, 1046
- GetSubSampling
 - gazebo::physics::HeightmapShape, 384
- GetSum
 - gazebo::math::Vector3, 1011
- GetSurface
 - gazebo::physics::Collision, 219
- GetSurfaceType
 - gazebo::common::SphericalCoordinates, 904
- GetTagPose
 - gazebo::sensors::RFIDTag, 714
- GetTangent
 - gazebo::math::Spline, 908
- GetTargetRealTimeFactor
 - gazebo::physics::PhysicsEngine, 628
- GetTension
 - gazebo::math::Spline, 908
- GetTexCoord
 - gazebo::common::SubMesh, 923
- GetTexCoordCount
 - gazebo::common::Mesh, 524
 - gazebo::common::SubMesh, 923
- GetText
 - gazebo::rendering::MovableText, 570
- GetTextureHeight
 - gazebo::rendering::Camera, 193
- GetTextureImage
 - gazebo::common::Material, 503
- GetTextureWidth
 - gazebo::rendering::Camera, 193
- GetThreadPitch
 - gazebo::physics::ScrewJoint, 748
 - gazebo::physics::SimbodyScrewJoint, 846
- GetThreshold
 - gazebo::physics::MapShape, 495
- GetTime
 - gazebo::common::Animation, 142
 - gazebo::common::KeyFrame, 447
 - gazebo::util::DiagnosticManager, 281
- GetTimeAtX
 - gazebo::common::NodeAnimation, 596
- GetTimerCount
 - gazebo::util::DiagnosticManager, 281
- GetTopic
 - gazebo::sensors::CameraSensor, 208
 - gazebo::sensors::ForceTorqueSensor, 336
 - gazebo::sensors::GpuRaySensor, 361
 - gazebo::sensors::MultiCameraSensor, 577
 - gazebo::sensors::RaySensor, 698
 - gazebo::sensors::Sensor, 757
 - gazebo::sensors::SonarSensor, 893
 - gazebo::sensors::WirelessTransceiver, 1065
 - gazebo::transport::PublicationTransport, 670
 - gazebo::transport::Publisher, 673
 - gazebo::transport::SubscribeOptions, 928
 - gazebo::transport::Subscriber, 930
- getTopicMsgType
 - Transport, 88
- GetTopicNamespace
 - gazebo::transport::Node, 590
- GetTopicNamespaces
 - gazebo::transport::ConnectionManager, 250
 - gazebo::transport::TopicManager, 971
- GetTorque
 - gazebo::sensors::ForceTorqueSensor, 336
- GetTransform
 - gazebo::common::SkeletonNode, 883
- GetTransforms
 - gazebo::common::SkeletonNode, 883
- GetTranslation
 - gazebo::common::PoseKeyFrame, 661
 - gazebo::math::Matrix4, 515
- GetTransparency
 - gazebo::common::Material, 503
 - gazebo::rendering::Visual, 1046
- GetTransportCount
 - gazebo::transport::Publication, 667
- GetTriangleCount
 - gazebo::rendering::Camera, 193
 - gazebo::rendering::UserCamera, 983
 - gazebo::rendering::WindowManager, 1058
- GetType
 - gazebo::common::NodeTransform, 601
 - gazebo::physics::Base, 160
 - gazebo::physics::PhysicsEngine, 628
 - gazebo::physics::SimbodyPhysics, 833
 - gazebo::PluginT, 648
 - gazebo::rendering::Light, 451
 - gazebo::sensors::Sensor, 757
- GetTypeString
 - gazebo::physics::SimbodyPhysics, 833
 - gazebo::rendering::FPSViewController, 339
 - gazebo::rendering::OrbitViewController, 618
 - gazebo::rendering::ViewController, 1032
- GetURI
 - Common, 40
 - gazebo::physics::HeightmapShape, 384
 - gazebo::physics::MapShape, 495
- getUniqueld
 - Classes for physics and dynamics, 67
- GetUp
 - gazebo::rendering::Camera, 194
- GetUpdatePeriod

- gazebo::physics::PhysicsEngine, 628
- GetUpdateRate
 - gazebo::sensors::Sensor, 758
- GetUpperLimit
 - gazebo::physics::Joint, 423
- GetUserCamera
 - gazebo::rendering::Scene, 739
- GetUserCameraCount
 - gazebo::rendering::Scene, 740
- GetVFOV
 - gazebo::rendering::Camera, 194
- GetValue
 - gazebo::common::NumericKeyFrame, 609
- GetVelocity
 - gazebo::physics::Joint, 423
 - gazebo::physics::LinkState, 480
 - gazebo::physics::SimbodyBallJoint, 782
 - gazebo::physics::SimbodyHinge2Joint, 796
 - gazebo::physics::SimbodyHingeJoint, 802
 - gazebo::physics::SimbodyScrewJoint, 846
 - gazebo::physics::SimbodySliderJoint, 853
 - gazebo::physics::SimbodyUniversalJoint, 861
- GetVelocityLimit
 - gazebo::physics::Joint, 424
- GetVertFOV
 - gazebo::rendering::GpuLaser, 349
 - gazebo::sensors::GpuRaySensor, 361
- GetVertHalfAngle
 - gazebo::rendering::GpuLaser, 349
 - gazebo::sensors::GpuRaySensor, 361
- GetVertNodeWeight
 - gazebo::common::Skeleton, 871
- GetVertex
 - gazebo::common::SubMesh, 924
- GetVertexCount
 - gazebo::common::Mesh, 524
 - gazebo::common::SubMesh, 924
 - gazebo::physics::HeightmapShape, 384
- GetVertexIndex
 - gazebo::common::SubMesh, 924
- GetVerticalAngleMax
 - gazebo::sensors::GpuRaySensor, 361
 - gazebo::sensors::RaySensor, 698
- GetVerticalAngleMin
 - gazebo::sensors::GpuRaySensor, 361
 - gazebo::sensors::RaySensor, 698
- GetVerticalMaxAngle
 - gazebo::physics::MultiRayShape, 584
- GetVerticalMinAngle
 - gazebo::physics::MultiRayShape, 584
- GetVerticalRangeCount
 - gazebo::sensors::GpuRaySensor, 362
 - gazebo::sensors::RaySensor, 698
- GetVerticalRayCount
 - gazebo::sensors::GpuRaySensor, 362
 - gazebo::sensors::RaySensor, 698
- GetVerticalSampleCount
 - gazebo::physics::MultiRayShape, 584
- GetVerticalScanResolution
 - gazebo::physics::MultiRayShape, 585
- GetViewControllerTypeString
 - gazebo::rendering::UserCamera, 983
- GetViewport
 - gazebo::rendering::Camera, 194
- GetViewportHeight
 - gazebo::rendering::Camera, 194
- GetViewportWidth
 - gazebo::rendering::Camera, 194
- GetVisibilityFlags
 - gazebo::rendering::Visual, 1046
- GetVisible
 - gazebo::rendering::Visual, 1046
- GetVisual
 - gazebo::rendering::Scene, 740
 - gazebo::rendering::UserCamera, 983, 984
- GetVisualAt
 - gazebo::rendering::Scene, 740, 741
- GetVisualBelow
 - gazebo::rendering::Scene, 741
- GetVisualCount
 - gazebo::rendering::Scene, 741
- GetVisualize
 - gazebo::sensors::Sensor, 758
- GetVisualsBelowPoint
 - gazebo::rendering::Scene, 741
- GetWallTime
 - gazebo::common::Time, 949
 - gazebo::physics::State, 913
- GetWallTimeAsISOString
 - gazebo::common::Time, 949
- GetWidth
 - gazebo::common::Image, 393
 - gazebo::common::Video, 1029
- GetWindow
 - gazebo::rendering::WindowManager, 1058
- GetWindowId
 - gazebo::rendering::Camera, 194
- GetWindowManager
 - gazebo::rendering::RenderEngine, 709
- GetWorld
 - gazebo::physics::Base, 160
- GetWorldAngularAccel
 - gazebo::physics::Collision, 219
 - gazebo::physics::Entity, 299
 - gazebo::physics::Link, 467
 - gazebo::physics::Model, 545
- GetWorldAngularVel
 - gazebo::physics::Collision, 219

- gazebo::physics::Entity, 299
- gazebo::physics::Model, 546
- gazebo::physics::SimbodyLink, 818
- GetWorldCFM
 - gazebo::physics::PhysicsEngine, 629
- GetWorldCoGLinearVel
 - gazebo::physics::Link, 468
 - gazebo::physics::SimbodyLink, 818
- GetWorldCoGPose
 - gazebo::physics::Link, 468
- GetWorldERP
 - gazebo::physics::PhysicsEngine, 629
- GetWorldForce
 - gazebo::physics::Link, 468
 - gazebo::physics::SimbodyLink, 818
- GetWorldLinearAccel
 - gazebo::physics::Collision, 219
 - gazebo::physics::Entity, 300
 - gazebo::physics::Link, 468
 - gazebo::physics::Model, 546
- GetWorldLinearVel
 - gazebo::physics::Collision, 219
 - gazebo::physics::Entity, 300
 - gazebo::physics::Link, 468, 469
 - gazebo::physics::Model, 546
 - gazebo::physics::SimbodyLink, 819
- GetWorldName
 - gazebo::sensors::Sensor, 758
- GetWorldPathExtension
 - gazebo::common::SystemPaths, 942
- GetWorldPointOnPlane
 - gazebo::rendering::Camera, 195
- GetWorldPose
 - gazebo::physics::Entity, 300
 - gazebo::rendering::Camera, 195
 - gazebo::rendering::Visual, 1046
- GetWorldPosition
 - gazebo::rendering::Camera, 195
- GetWorldRotation
 - gazebo::rendering::Camera, 195
- GetWorldTorque
 - gazebo::physics::Link, 469
 - gazebo::physics::SimbodyLink, 819
- getWorldTransforms
 - gazebo::rendering::MovableText, 570
- GetWorldVisual
 - gazebo::rendering::Scene, 741
- GetWrench
 - gazebo::physics::LinkState, 480
- GetXAxis
 - gazebo::math::Quaternion, 681
- GetXLength
 - gazebo::math::Box, 166
- GetYAxis
 - gazebo::math::Quaternion, 681
- GetYLength
 - gazebo::math::Box, 166
- GetYaw
 - gazebo::math::Quaternion, 681
- GetZAxis
 - gazebo::math::Quaternion, 681
- GetZLength
 - gazebo::math::Box, 167
- GetZValue
 - gazebo::rendering::Camera, 195
- globalEndPos
 - gazebo::physics::RayShape, 705
- GlobalFromLocal
 - gazebo::common::SphericalCoordinates, 904
- globalStartPos
 - gazebo::physics::RayShape, 705
- google, 122
- google::protobuf, 122
- google::protobuf::compiler, 122
- google::protobuf::compiler::cpp, 123
- google::protobuf::compiler::cpp::GazeboGenerator, 340
 - ~GazeboGenerator, 341
 - GazeboGenerator, 341
 - Generate, 341
- GpsSensor
 - gazebo::sensors::GpsSensor, 342
- GpsSensor.hh, 1147
- GpsSensorPtr
 - gazebo::sensors, 118
- GpuLaser
 - gazebo::rendering::GpuLaser, 347
- GpuLaser.hh, 1148
- GpuLaserPtr
 - gazebo::rendering, 114
- GpuRaySensor
 - gazebo::sensors::GpuRaySensor, 356
- GpuRaySensor.hh, 1148
- GpuRaySensor_V
 - gazebo::sensors, 118
- GpuRaySensorPtr
 - gazebo::sensors, 118
- gravity
 - gazebo::physics::SimbodyPhysics, 837
- Green
 - gazebo::common::Color, 237
- Grid
 - gazebo::rendering::Grid, 366
- Grid.hh, 1149
- Gripper
 - gazebo::physics::Gripper, 369
- Gripper.hh, 1150
- GripperPtr
 - gazebo::physics, 109

- GtsSurface
 - MeshCSG.hh, 1185
- GzTerrainMatGen
 - gazebo::rendering::GzTerrainMatGen, 375
- gzclr_end
 - Common, 35
- gzclr_start
 - Common, 35
- gzdbg
 - Common, 35
- gzerr
 - Common, 35
- gzlog
 - Common, 35
- gzmsg
 - Common, 35
- gzthrow
 - Common, 36
- gzwarn
 - Common, 36

- H_CENTER
 - gazebo::rendering::MovableText, 568
- H_LEFT
 - gazebo::rendering::MovableText, 568
- HEADER_LENGTH
 - Connection.hh, 1125
- HEIGHTMAP_SHAPE
 - gazebo::physics::Base, 156
- HI_STOP
 - gazebo::physics::Joint, 415
- HINGE2_JOINT
 - gazebo::physics::Base, 156
- HINGE_JOINT
 - gazebo::physics::Base, 156
- HalfPi
 - gazebo::math::Angle, 139
- handle
 - gazebo::common::SkeletonNode, 885
 - gazebo::PluginT, 648
- HandleData
 - gazebo::transport::CallbackHelper, 175
 - gazebo::transport::CallbackHelperT, 178
 - gazebo::transport::Node, 590
 - gazebo::transport::RawCallbackHelper, 692
 - gazebo::transport::SubscriptionTransport, 932
- HandleKeyPressEvent
 - gazebo::rendering::FPSViewController, 339
 - gazebo::rendering::GUIOverlay, 372
 - gazebo::rendering::OrbitViewController, 619
 - gazebo::rendering::UserCamera, 984
 - gazebo::rendering::ViewController, 1032
- HandleKeyReleaseEvent
 - gazebo::rendering::FPSViewController, 339
 - gazebo::rendering::GUIOverlay, 373
 - gazebo::rendering::OrbitViewController, 619
 - gazebo::rendering::UserCamera, 984
 - gazebo::rendering::ViewController, 1033
- HandleMessage
 - gazebo::transport::CallbackHelper, 176
 - gazebo::transport::CallbackHelperT, 178
 - gazebo::transport::Node, 590
 - gazebo::transport::RawCallbackHelper, 692
 - gazebo::transport::SubscriptionTransport, 932
- HandleMouseEvent
 - gazebo::rendering::FPSViewController, 339
 - gazebo::rendering::GUIOverlay, 373
 - gazebo::rendering::OrbitViewController, 619
 - gazebo::rendering::UserCamera, 984
 - gazebo::rendering::ViewController, 1033
- HasAttachedObject
 - gazebo::rendering::Visual, 1047
- HasCollisionName
 - gazebo::util::OpenALSource, 614
- HasConnections
 - gazebo::transport::Publisher, 673
- HasJointState
 - gazebo::physics::ModelState, 560
- HasLatchedSubscriber
 - gazebo::transport::Node, 591
- HasLinkState
 - gazebo::physics::ModelState, 560
- HasMesh
 - gazebo::common::MeshManager, 533
- HasModel
 - Common, 40
- HasModelState
 - gazebo::physics::WorldState, 1086
- HasNode
 - gazebo::common::SkeletonAnimation, 876
- HasSkeleton
 - gazebo::common::Mesh, 525
- HasTransport
 - gazebo::transport::Publication, 667
- HasType
 - gazebo::physics::Base, 160
- HasVertex
 - gazebo::common::SubMesh, 924
- Heightmap
 - gazebo::rendering::Heightmap, 377
- Heightmap.hh, 1151
- HeightmapShape
 - gazebo::physics::HeightmapShape, 382
- HeightmapShape.hh, 1152
- HeightmapShapePtr
 - gazebo::physics, 109
- heights
 - gazebo::physics::HeightmapShape, 385

- Helpers.hh, 1153
 - GZ_DBL_MAX, 1155
 - GZ_DBL_MIN, 1155
 - GZ_FLT_MAX, 1155
 - GZ_FLT_MIN, 1156
 - GZ_UINT32_MAX, 1156
 - GZ_UINT32_MIN, 1156
- hfov
 - gazebo::rendering::GpuLaser, 353
- Hide
 - gazebo::rendering::GUIOverlay, 373
- Hinge2Joint
 - gazebo::physics::Hinge2Joint, 387
- Hinge2Joint.hh, 1156
- HingeJoint
 - gazebo::physics::HingeJoint, 388
- HingeJoint.hh, 1157
- HorizAlign
 - gazebo::rendering::MovableText, 568
- horzElem
 - gazebo::physics::MultiRayShape, 586
 - gazebo::sensors::GpuRaySensor, 364
- horzHalfAngle
 - gazebo::rendering::GpuLaser, 353
- horzRangeCount
 - gazebo::sensors::GpuRaySensor, 364
- horzRayCount
 - gazebo::sensors::GpuRaySensor, 364
- IDENTITY
 - gazebo::math::Matrix4, 519
- IMAGE
 - gazebo::sensors, 119
- INTERSECTION
 - gazebo::common::MeshCSG, 527
- IOManager
 - gazebo::transport::IOManager, 410
- IOManager.hh, 1161
- id
 - gazebo::common::SkeletonNode, 885
 - gazebo::physics::TrajectoryInfo, 974
- Image
 - gazebo::common::Image, 391
- Image.hh, 1159
- imageFormat
 - gazebo::rendering::Camera, 204
- imageHeight
 - gazebo::rendering::Camera, 204
- imageWidth
 - gazebo::rendering::Camera, 204
- img
 - gazebo::physics::HeightmapShape, 385
- ImuSensor
 - gazebo::sensors::ImuSensor, 396
- ImuSensor.hh, 1160
- ImuSensor_V
 - gazebo::sensors, 118
- ImuSensorPtr
 - gazebo::sensors, 118
- IncCount
 - gazebo::transport::IOManager, 410
- indexBufferCapacity
 - gazebo::rendering::DynamicRenderable, 293
- inertiaRatio
 - gazebo::physics::Joint, 429
- Inertial
 - gazebo::physics::Inertial, 400, 401
- inertial
 - gazebo::physics::Link, 475
- Inertial.hh, 1160
- InertialPtr
 - gazebo::physics, 109
- Init
 - Common, 40
 - gazebo::common::PID, 637
 - gazebo::Master, 497
 - gazebo::ModelPlugin, 554
 - gazebo::physics::Actor, 128
 - gazebo::physics::Base, 160
 - gazebo::physics::BoxShape, 171
 - gazebo::physics::Collision, 219
 - gazebo::physics::ContactManager, 258
 - gazebo::physics::CylinderShape, 271
 - gazebo::physics::Gripper, 369
 - gazebo::physics::HeightmapShape, 384
 - gazebo::physics::HingeJoint, 389
 - gazebo::physics::Joint, 424
 - gazebo::physics::Link, 469
 - gazebo::physics::MapShape, 495
 - gazebo::physics::MeshShape, 536
 - gazebo::physics::Model, 546
 - gazebo::physics::MultiRayShape, 585
 - gazebo::physics::PhysicsEngine, 629
 - gazebo::physics::PlaneShape, 645
 - gazebo::physics::RayShape, 703
 - gazebo::physics::Road, 719
 - gazebo::physics::Shape, 777
 - gazebo::physics::SimbodyBallJoint, 782
 - gazebo::physics::SimbodyHeightmapShape, 792
 - gazebo::physics::SimbodyHinge2Joint, 797
 - gazebo::physics::SimbodyLink, 819
 - gazebo::physics::SimbodyMeshShape, 824
 - gazebo::physics::SimbodyModel, 826
 - gazebo::physics::SimbodyPhysics, 833
 - gazebo::physics::SimbodyScrewJoint, 847
 - gazebo::physics::SimbodyUniversalJoint, 862
 - gazebo::physics::SphereShape, 900
 - gazebo::physics::World, 1077

- gazebo::rendering::Camera, 196
- gazebo::rendering::DepthCamera, 275
- gazebo::rendering::DynamicRenderable, 292
- gazebo::rendering::FPSViewController, 340
- gazebo::rendering::GpuLaser, 349
- gazebo::rendering::Grid, 367
- gazebo::rendering::GUIOverlay, 373
- gazebo::rendering::OrbitViewController, 619
- gazebo::rendering::RenderEngine, 709
- gazebo::rendering::RTShaderSystem, 727
- gazebo::rendering::Scene, 742
- gazebo::rendering::UserCamera, 984
- gazebo::rendering::ViewController, 1033
- gazebo::rendering::Visual, 1047
- gazebo::rendering::WireBox, 1060
- gazebo::SensorPlugin, 769
- gazebo::sensors::CameraSensor, 208
- gazebo::sensors::ContactSensor, 264
- gazebo::sensors::DepthCameraSensor, 278
- gazebo::sensors::ForceTorqueSensor, 336
- gazebo::sensors::GpsSensor, 343
- gazebo::sensors::GpuRaySensor, 362
- gazebo::sensors::ImuSensor, 397
- gazebo::sensors::MultiCameraSensor, 577
- gazebo::sensors::RaySensor, 699
- gazebo::sensors::RFIDSensor, 711
- gazebo::sensors::RFIDTag, 714
- gazebo::sensors::Sensor, 758
- gazebo::sensors::SensorManager, 766
- gazebo::sensors::SonarSensor, 894
- gazebo::sensors::WirelessReceiver, 1062
- gazebo::sensors::WirelessTransceiver, 1065
- gazebo::sensors::WirelessTransmitter, 1069
- gazebo::Server, 770
- gazebo::SystemPlugin, 944
- gazebo::transport::ConnectionManager, 250
- gazebo::transport::Node, 591
- gazebo::transport::PublicationTransport, 670
- gazebo::transport::SubscribeOptions, 928, 929
- gazebo::transport::SubscriptionTransport, 932
- gazebo::transport::TopicManager, 972
- gazebo::util::DiagnosticManager, 282
- gazebo::util::LogRecord, 490
- gazebo::VisualPlugin, 1056
- gazebo::WorldPlugin, 1083
- Messages, 58
- init
 - gazebo, 95
 - Rendering, 77
 - Sensors, 83
 - Transport, 88
- init_world
 - Classes for physics and dynamics, 67
- init_worlds
 - Classes for physics and dynamics, 68
- InitForThread
 - gazebo::physics::PhysicsEngine, 629
 - gazebo::physics::SimbodyPhysics, 833
- InitModel
 - gazebo::physics::SimbodyPhysics, 833
- initialTransform
 - gazebo::common::SkeletonNode, 885
- initialized
 - gazebo::rendering::Camera, 204
- InsertLatchedMsg
 - gazebo::transport::Node, 591
- InsertMesh
 - gazebo::rendering::Visual, 1047
- InsertModelFile
 - gazebo::physics::World, 1078
- InsertModelSDF
 - gazebo::physics::World, 1078
- InsertModelString
 - gazebo::physics::World, 1078
- Instance
 - SingletonT, 866
- integ
 - gazebo::physics::SimbodyPhysics, 837
- InternalError
 - gazebo::common::InternalError, 409
- Interpolate
 - gazebo::math::RotationSpline, 722, 723
 - gazebo::math::Spline, 908
- interpolateX
 - gazebo::physics::Actor, 129
- invBindTransform
 - gazebo::common::SkeletonNode, 885
- Inverse
 - gazebo::math::Matrix4, 515
- Invert
 - gazebo::math::Quaternion, 682
- is_stopped
 - Transport, 89
- IsActive
 - gazebo::physics::Actor, 128
 - gazebo::sensors::CameraSensor, 209
 - gazebo::sensors::ContactSensor, 264
 - gazebo::sensors::ForceTorqueSensor, 336
 - gazebo::sensors::GpuRaySensor, 362
 - gazebo::sensors::ImuSensor, 397
 - gazebo::sensors::MultiCameraSensor, 577
 - gazebo::sensors::RaySensor, 699
 - gazebo::sensors::Sensor, 758
 - gazebo::sensors::SonarSensor, 894
- IsAdvertised
 - gazebo::transport::TopicManager, 972
- IsAffine
 - gazebo::math::Matrix4, 515

- IsAnimating
 - gazebo::rendering::Camera, 196
- IsAttached
 - gazebo::physics::Gripper, 370
- IsCanonicalLink
 - gazebo::physics::Entity, 300
- IsFinite
 - gazebo::math::Pose, 653
 - gazebo::math::Quaternion, 682
 - gazebo::math::Vector2d, 990
 - gazebo::math::Vector2i, 998
 - gazebo::math::Vector3, 1011
 - gazebo::math::Vector4, 1021
- IsHorizontal
 - gazebo::rendering::GpuLaser, 350
 - gazebo::sensors::GpuRaySensor, 362
- isHorizontal
 - gazebo::rendering::GpuLaser, 353
- IsInitialized
 - Common, 40
 - gazebo::rendering::GUIOverlay, 373
- IsJoint
 - gazebo::common::SkeletonNode, 883
- IsLoaded
 - gazebo::physics::World, 1078
- IsLocal
 - gazebo::transport::CallbackHelper, 176
 - gazebo::transport::CallbackHelperT, 179
 - gazebo::transport::RawCallbackHelper, 692
 - gazebo::transport::SubscriptionTransport, 933
- IsOpen
 - gazebo::transport::Connection, 246
 - gazebo::util::LogPlay, 485
- IsPaused
 - gazebo::physics::World, 1078
- IsPlaceable
 - gazebo::physics::Collision, 220
- IsPlane
 - gazebo::rendering::Visual, 1047
- IsPlaying
 - gazebo::util::OpenALSource, 615
- isPowerOfTwo
 - Math, 49
- IsReadyToStart
 - gazebo::util::LogRecord, 491
- IsRegistered
 - gazebo::physics::PhysicsFactory, 635
- isReversed
 - gazebo::physics::SimbodyJoint, 812
- IsRootNode
 - gazebo::common::SkeletonNode, 883
- IsRunning
 - gazebo::transport::ConnectionManager, 250
- IsSelected
 - gazebo::physics::Base, 161
- IsStatic
 - gazebo::physics::Entity, 300
 - gazebo::rendering::Visual, 1048
- IsValidFilename
 - gazebo::common::MeshManager, 533
- IsVisible
 - gazebo::rendering::Camera, 196
- IsZero
 - gazebo::physics::CollisionState, 225
 - gazebo::physics::JointState, 439
 - gazebo::physics::LinkState, 480
 - gazebo::physics::ModelState, 560
 - gazebo::physics::WorldState, 1087
- isnan
 - Math, 48
- JOINT
 - gazebo::common::SkeletonNode, 879
 - gazebo::physics::Base, 156
- Joint
 - gazebo::physics::Joint, 415
- Joint.hh, 1163
 - MAX_JOINT_AXIS, 1164
- Joint_TEST, 430
 - ForceTorque1, 431
 - ForceTorque2, 432
 - GetForceTorqueWithAppliedForce, 432
 - Joint_TEST, 431
 - JointCreationDestructionTest, 432
 - JointTorqueTest, 432
 - jointType, 434
 - Joint_TEST, 431
 - physicsEngine, 434
 - SetUp, 432
 - SpawnJoint, 432, 433
 - SpawnJointRotational, 433
 - SpawnJointRotationalWorld, 433
 - SpawnJointTypes, 433
- Joint_TEST.hh, 1164
 - std_string2, 1165
- Joint_TEST::SpawnJointOptions, 896
 - ~SpawnJointOptions, 897
 - axis, 897
 - childLinkPose, 897
 - jointPose, 897
 - modelPose, 898
 - parentLinkPose, 898
 - SpawnJointOptions, 897
 - type, 898
 - wait, 898
 - worldChild, 898
 - worldParent, 898
- Joint_V

- gazebo::physics, 109
- JointController
 - gazebo::physics::JointController, 435
- JointController.hh, 1165
- JointController_V
 - gazebo::physics, 110
- JointControllerPtr
 - gazebo::physics, 110
- JointCreationDestructionTest
 - Joint_TEST, 432
- jointPose
 - Joint_TEST::SpawnJointOptions, 897
- JointPtr
 - gazebo::physics, 110
- jointPub
 - gazebo::physics::Model, 551
- JointState
 - gazebo::physics::JointState, 437, 438
- JointState.hh, 1166
- JointState_M
 - gazebo::physics, 110
- JointTorqueTest
 - Joint_TEST, 432
- jointType
 - Joint_TEST, 434
- JointVisual
 - gazebo::rendering::JointVisual, 441
- JointVisual.hh, 1167
- JointVisualPtr
 - gazebo::rendering, 114
- JointWrench.hh, 1167
- kd
 - gazebo::physics::SurfaceParams, 936
- key
 - gazebo::common::KeyEvent, 445
- KeyEvent
 - gazebo::common::KeyEvent, 445
- KeyEvent.hh, 1169
- KeyFrame
 - gazebo::common::KeyFrame, 446
- KeyFrame.hh, 1169
- KeyFrame_V
 - gazebo::common::Animation, 141
- keyFrames
 - gazebo::common::Animation, 143
 - gazebo::common::NodeAnimation, 597
- kp
 - gazebo::physics::SurfaceParams, 936
- L_INT16
 - gazebo::common::Image, 391
- L_INT8
 - gazebo::common::Image, 391
- LEFT
 - gazebo::common::MouseEvent, 564
- LIGHT
 - gazebo::physics::Base, 156
- LINE_MAX_LEN
 - STLLoader.hh, 1280
- LINES
 - gazebo::common::SubMesh, 919
- LINESTRIPS
 - gazebo::common::SubMesh, 919
- LINK
 - gazebo::physics::Base, 156
- LINUX
 - SystemPaths.hh, 1288
- LO_STOP
 - gazebo::physics::Joint, 415
- Lap
 - gazebo::util::DiagnosticManager, 282
 - gazebo::util::DiagnosticTimer, 284
- LaserVisual
 - gazebo::rendering::LaserVisual, 448
- LaserVisual.hh, 1171
- LaserVisualPtr
 - gazebo::rendering, 114
- lastMeasurementTime
 - gazebo::sensors::Sensor, 761
- lastPos
 - gazebo::physics::Actor, 129
- lastRenderWallTime
 - gazebo::rendering::Camera, 204
- lastScriptTime
 - gazebo::physics::Actor, 129
- lastTraj
 - gazebo::physics::Actor, 129
- lastUpdateTime
 - gazebo::sensors::Sensor, 761
- latching
 - gazebo::transport::CallbackHelper, 176
- length
 - gazebo::common::Animation, 143
 - gazebo::common::NodeAnimation, 597
 - gazebo::common::SkeletonAnimation, 876
- Light
 - gazebo::rendering::Light, 450
- Light.hh, 1171
- LightFromSDF
 - Messages, 59
- LightPtr
 - gazebo::rendering, 114
- LightingModel
 - gazebo::rendering::RTShaderSystem, 726
- limitForce
 - gazebo::physics::SimbodyJoint, 812
- linearAccel
 - gazebo::physics::Link, 475

Link
 gazebo::physics::Link, 460
 link
 gazebo::physics::Collision, 222
 Link.hh, 1172
 Link_V
 gazebo::physics, 110
 LinkPtr
 gazebo::physics, 110
 LinkState
 gazebo::physics::LinkState, 478
 LinkState.hh, 1173
 LinkState_M
 gazebo::physics, 110
 Listen
 gazebo::transport::Connection, 246
 Load
 gazebo::common::BVHLoader, 173
 gazebo::common::ColladaLoader, 212
 gazebo::common::Image, 394
 gazebo::common::MeshLoader, 528
 gazebo::common::MeshManager, 533
 gazebo::common::STLLoader, 916
 gazebo::common::Video, 1029
 gazebo::ModelPlugin, 554
 gazebo::physics::Actor, 128
 gazebo::physics::BallJoint, 152
 gazebo::physics::Base, 161
 gazebo::physics::Collision, 220
 gazebo::physics::CollisionState, 225
 gazebo::physics::Entity, 301
 gazebo::physics::Gripper, 370
 gazebo::physics::HeightmapShape, 384
 gazebo::physics::Hinge2Joint, 387
 gazebo::physics::HingeJoint, 389
 gazebo::physics::Inertial, 403
 gazebo::physics::Joint, 424
 gazebo::physics::JointState, 439
 gazebo::physics::Link, 469
 gazebo::physics::LinkState, 480, 481
 gazebo::physics::MapShape, 495
 gazebo::physics::Model, 546
 gazebo::physics::ModelState, 561
 gazebo::physics::PhysicsEngine, 629
 gazebo::physics::Road, 719
 gazebo::physics::ScrewJoint, 748
 gazebo::physics::SimbodyBallJoint, 782
 gazebo::physics::SimbodyCollision, 787
 gazebo::physics::SimbodyHinge2Joint, 797
 gazebo::physics::SimbodyHingeJoint, 802
 gazebo::physics::SimbodyJoint, 810
 gazebo::physics::SimbodyLink, 820
 gazebo::physics::SimbodyMeshShape, 824
 gazebo::physics::SimbodyModel, 826
 gazebo::physics::SimbodyPhysics, 834
 gazebo::physics::SimbodyScrewJoint, 847
 gazebo::physics::SimbodySliderJoint, 853
 gazebo::physics::SimbodyUniversalJoint, 862
 gazebo::physics::SliderJoint, 888
 gazebo::physics::State, 913
 gazebo::physics::SurfaceParams, 934
 gazebo::physics::UniversalJoint, 978
 gazebo::physics::World, 1078
 gazebo::physics::WorldState, 1087
 gazebo::rendering::ArrowVisual, 145
 gazebo::rendering::AxisVisual, 150
 gazebo::rendering::Camera, 196, 197
 gazebo::rendering::CameraVisual, 211
 gazebo::rendering::COMVisual, 239
 gazebo::rendering::DepthCamera, 275, 276
 gazebo::rendering::GpuLaser, 350
 gazebo::rendering::Heightmap, 378
 gazebo::rendering::JointVisual, 442
 gazebo::rendering::Light, 451, 452
 gazebo::rendering::MovableText, 570
 gazebo::rendering::Projector, 663
 gazebo::rendering::RenderEngine, 709
 gazebo::rendering::Road2d, 720
 gazebo::rendering::Scene, 742
 gazebo::rendering::SonarVisual, 896
 gazebo::rendering::TransmitterVisual, 976
 gazebo::rendering::UserCamera, 984, 985
 gazebo::rendering::Visual, 1048
 gazebo::rendering::WrenchVisual, 1091
 gazebo::SensorPlugin, 769
 gazebo::sensors::CameraSensor, 209
 gazebo::sensors::ContactSensor, 264
 gazebo::sensors::DepthCameraSensor, 278
 gazebo::sensors::ForceTorqueSensor, 336
 gazebo::sensors::GpsSensor, 343
 gazebo::sensors::GpuRaySensor, 362, 363
 gazebo::sensors::ImuSensor, 397, 398
 gazebo::sensors::MultiCameraSensor, 577
 gazebo::sensors::Noise, 605
 gazebo::sensors::RaySensor, 699
 gazebo::sensors::RFIDSensor, 711, 712
 gazebo::sensors::RFIDTag, 714
 gazebo::sensors::Sensor, 759
 gazebo::sensors::SonarSensor, 894
 gazebo::sensors::WirelessReceiver, 1063
 gazebo::sensors::WirelessTransceiver, 1065
 gazebo::sensors::WirelessTransmitter, 1069
 gazebo::SystemPlugin, 944
 gazebo::util::OpenAL, 611
 gazebo::util::OpenALSource, 615
 gazebo::VisualPlugin, 1056
 gazebo::WorldPlugin, 1083
 Rendering, 78

- load
 - Classes for physics and dynamics, 68
 - Common, 40
 - gazebo, 95
 - Rendering, 77
 - Sensors, 83
- load_world
 - Classes for physics and dynamics, 68
- load_worlds
 - Classes for physics and dynamics, 68
- LoadFile
 - gazebo::Server, 770
- LoadFromMsg
 - gazebo::rendering::Heightmap, 378
 - gazebo::rendering::Light, 452
 - gazebo::rendering::Visual, 1048
- LoadJoints
 - gazebo::physics::Model, 547
- LoadLayout
 - gazebo::rendering::GUIOverlay, 374
- LoadPlugin
 - gazebo::physics::World, 1079
 - gazebo::rendering::Visual, 1048
- LoadPlugins
 - gazebo::physics::Model, 547
- loadProceduralPage
 - gazebo::rendering::DummyPageProvider, 285
- LoadString
 - gazebo::Server, 770
- LocalPublish
 - gazebo::transport::Publication, 668
- Log
 - Common, 41
- LogPlay.hh, 1175
- LogRecord.hh, 1175
 - GZ_LOG_VERSION, 1176
- Logplay, 486
- loop
 - gazebo::common::Animation, 143
 - gazebo::physics::Actor, 130
- Lower
 - gazebo::rendering::Heightmap, 378
- lowerLimit
 - gazebo::physics::Joint, 429
- m
 - gazebo::math::Matrix3, 512
 - gazebo::math::Matrix4, 519
- MAP_SHAPE
 - gazebo::physics::Base, 156
- MATRIX
 - gazebo::common::NodeTransform, 600
- MAX_COLLIDE_RETURNS
 - Contact.hh, 1129
- MAX_CONTACT_JOINTS
 - Contact.hh, 1129
- MAX_JOINT_AXIS
 - Joint.hh, 1164
- MESH_SHAPE
 - gazebo::physics::Base, 157
- MIDDLE
 - gazebo::common::MouseEvent, 564
- MODEL
 - gazebo::physics::Base, 156
- MODEL_PLUGIN
 - Common, 36
- MODULATE
 - gazebo::common::Material, 500
- MOVE
 - gazebo::common::MouseEvent, 564
- MSleep
 - gazebo::common::Time, 950
- MULTIRAY_SHAPE
 - gazebo::physics::Base, 156
- mainLink
 - gazebo::physics::Actor, 130
- mainpage.html, 1176
- MakeStatic
 - gazebo::rendering::Visual, 1048
- MapShape
 - gazebo::physics::MapShape, 494
- MapShape.hh, 1176
- Master
 - gazebo::Master, 497
- Master.hh, 1178
- masterMobod
 - gazebo::physics::SimbodyLink, 822
- Material
 - gazebo::common::Material, 501
- Material.hh, 1179, 1180
- Math, 46
 - clamp, 48
 - equal, 48
 - isPowerOfTwo, 49
 - isnan, 48
 - max, 49
 - mean, 49
 - min, 49
 - NAN_D, 51
 - NAN_I, 51
 - parseFloat, 50
 - parseInt, 50
 - precision, 50
 - variance, 50
- MathTypes.hh, 1180
- Matrix3
 - gazebo::math::Matrix3, 508, 509
- Matrix3.hh, 1181

- Matrix4
 - gazebo::math::Matrix4, 514
- Matrix4.hh, 1181
- matter
 - gazebo::physics::SimbodyPhysics, 837
- max
 - gazebo::math::Box, 169
 - Math, 49
- maxStepSize
 - gazebo::physics::PhysicsEngine, 633
- maxVel
 - gazebo::physics::SurfaceParams, 936
- mean
 - Math, 49
- Merge
 - gazebo::math::Box, 167
- Mesh
 - gazebo::common::Mesh, 521
- mesh
 - gazebo::physics::Actor, 130
 - gazebo::physics::MeshShape, 537
- Mesh.hh, 1182
- MeshCSG
 - gazebo::common::MeshCSG, 527
- MeshCSG.hh, 1184
 - GPtrArray, 1185
 - GtsSurface, 1185
- MeshFromSDF
 - Messages, 59
- MeshLoader
 - gazebo::common::MeshLoader, 528
- MeshLoader.hh, 1186
- MeshManager.hh, 1187
- MeshShape
 - gazebo::physics::MeshShape, 535
- MeshShape.hh, 1188
- MeshShapePtr
 - gazebo::physics, 110
- MessagePtr
 - gazebo::transport, 121
- Messages, 52
 - Convert, 54–57
 - CreateRequest, 57
 - FogFromSDF, 57
 - GUIFromSDF, 58
 - GZ_REGISTER_STATIC_MSG, 54
 - GeometryFromSDF, 58
 - GetHeader, 58
 - Init, 58
 - LightFromSDF, 59
 - MeshFromSDF, 59
 - SceneFromSDF, 59
 - Set, 59–61
 - Stamp, 61
 - TrackVisualFromSDF, 61
 - VisualFromSDF, 62
- MicToNano
 - gazebo::common::Time, 950
- MilToNano
 - gazebo::common::Time, 950
- min
 - gazebo::math::Box, 169
 - Math, 49
- minDepth
 - gazebo::physics::SurfaceParams, 936
- mobod
 - gazebo::physics::SimbodyJoint, 813
- Model
 - gazebo::physics::Model, 541
- model
 - gazebo::physics::Joint, 429
- Model.hh, 1190
- Model_V
 - gazebo::physics, 110
- ModelDatabase.hh, 1191
 - GZ_MODEL_DB_MANIFEST_FILENAME, 1192
 - GZ_MODEL_MANIFEST_FILENAME, 1192
- modelPathsFromEnv
 - gazebo::common::SystemPaths, 942
- ModelPlugin
 - gazebo::ModelPlugin, 554
- ModelPluginPtr
 - gazebo, 95
- modelPose
 - Joint_TEST::SpawnJointOptions, 898
- ModelPtr
 - gazebo::physics, 110
- ModelState
 - gazebo::physics::ModelState, 556, 557
- ModelState.hh, 1192
- ModelState_M
 - gazebo::physics, 110
- ModelStdDesv
 - gazebo::sensors::WirelessTransmitter, 1069
- modelTransform
 - gazebo::common::SkeletonNode, 885
- MouseEvent
 - gazebo::common::MouseEvent, 565
- MouseEvent.hh, 1193
- MovableText
 - gazebo::rendering::MovableText, 568
- MovableText.hh, 1195
- moveScale
 - gazebo::common::MouseEvent, 565
- MoveToPosition
 - gazebo::rendering::Camera, 197
 - gazebo::rendering::UserCamera, 985
 - gazebo::rendering::Visual, 1049

- MoveToPositions
 - gazebo::rendering::Camera, 197
 - gazebo::rendering::Visual, 1049
- MoveToVisual
 - gazebo::rendering::UserCamera, 985
- Moved
 - gazebo::rendering::WindowManager, 1059
- MsgFactory.hh, 1195
- MsgFactoryFn
 - gazebo::msgs, 104
- msgs.hh, 1196
- mu1
 - gazebo::physics::SurfaceParams, 936
- mu2
 - gazebo::physics::SurfaceParams, 937
- MultiCameraSensor
 - gazebo::sensors::MultiCameraSensor, 575
- MultiCameraSensor.hh, 1199
- MultiRayShape
 - gazebo::physics::MultiRayShape, 581
- MultiRayShape.hh, 1199
- MultiRayShapePtr
 - gazebo::physics, 110
- mustBeBaseLink
 - gazebo::physics::SimbodyLink, 822
- mustBreakLoopHere
 - gazebo::physics::SimbodyJoint, 813
- mutexLastUpdateTime
 - gazebo::sensors::Sensor, 761

- NAN_D
 - Math, 51
- NAN_I
 - Math, 51
- NEmpty
 - gazebo::sensors::WirelessTransmitter, 1069
- NO_BUTTON
 - gazebo::common::MouseEvent, 564
- NO_EVENT
 - gazebo::common::KeyEvent, 445
 - gazebo::common::MouseEvent, 564
- NODE
 - gazebo::common::SkeletonNode, 879
- NONE
 - gazebo::rendering::RenderEngine, 707
 - gazebo::sensors::Noise, 604
- NObstacle
 - gazebo::sensors::WirelessTransmitter, 1070
- NRealGen
 - gazebo::math, 102
- NSleep
 - gazebo::common::Time, 950
- NULL
 - CommonTypes.hh, 1122

- name
 - gazebo::common::Animation, 143
 - gazebo::common::Material, 506
 - gazebo::common::NodeAnimation, 597
 - gazebo::common::SkeletonAnimation, 877
 - gazebo::common::SkeletonNode, 885
 - gazebo::physics::State, 915
 - gazebo::rendering::Camera, 204
- near
 - gazebo::rendering::GpuLaser, 353
- NewContact
 - gazebo::physics::ContactManager, 259
- newData
 - gazebo::rendering::Camera, 204
- newImageFrame
 - gazebo::rendering::Camera, 204
- newLaserScans
 - gazebo::physics::MultiRayShape, 586
- NewMsg
 - gazebo::msgs::MsgFactory, 573
- NewPhysicsEngine
 - gazebo::physics::PhysicsFactory, 635
- NewSensor
 - gazebo::sensors::SensorFactory, 763
- Node
 - gazebo::transport::Node, 588
- node
 - gazebo::physics::Entity, 304
 - gazebo::physics::Gripper, 370
 - gazebo::physics::PhysicsEngine, 633
 - gazebo::sensors::Sensor, 761
- Node.hh, 1200
- NodeAnimation
 - gazebo::common::NodeAnimation, 595
- nodeIndex
 - gazebo::common::NodeAssignment, 598
- NodeMap
 - gazebo::common, 98
- NodeMapItr
 - gazebo::common, 98
- NodePtr
 - gazebo::transport, 121
- NodeTransform
 - gazebo::common::NodeTransform, 600
- nodes
 - gazebo::common::Skeleton, 872
- Noise
 - gazebo::sensors::Noise, 604
- Noise.hh, 1202
- NoisePtr
 - gazebo::sensors, 118
- NoiseType
 - gazebo::sensors::Noise, 604
- normal

- gazebo::math::Plane, 642
- NormalRealDist
 - gazebo::math, 102
- Normalize
 - gazebo::math::Angle, 134
 - gazebo::math::Quaternion, 682
 - gazebo::math::Vector2d, 990
 - gazebo::math::Vector2i, 999
 - gazebo::math::Vector3, 1011
 - gazebo::math::Vector4, 1021
- normals
 - gazebo::physics::Contact, 256
- Notify
 - gazebo::util::LogRecord, 491
- notifyRenderSingleObject
 - gazebo::rendering::GpuLaser, 350
- nsec
 - gazebo::common::Time, 965
- NullStream
 - Common, 36
- NumTerrainSubdivisions
 - gazebo::rendering::Heightmap, 380
- NumericAnimation
 - gazebo::common::NumericAnimation, 606
- NumericAnimationPtr
 - gazebo::common, 98
- NumericKeyFrame
 - gazebo::common::NumericKeyFrame, 608

- ORDER_MAX
 - STLLoader.hh, 1280
- OTHER
 - gazebo::sensors, 119
- offset
 - gazebo::physics::MultiRayShape, 586
- Ogre, 123
- ogre, 123
- ogre_gazebo.h, 1202
- ogrePathsFromEnv
 - gazebo::common::SystemPaths, 942
- oldAction
 - gazebo::physics::Actor, 130
- onAnimationComplete
 - gazebo::rendering::Camera, 204
- OnPhysicsMsg
 - gazebo::physics::PhysicsEngine, 629
 - gazebo::physics::SimbodyPhysics, 834
- OnPoseChange
 - gazebo::physics::Entity, 301
 - gazebo::physics::Link, 470
 - gazebo::physics::Model, 547
 - gazebo::physics::SimbodyCollision, 788
 - gazebo::physics::SimbodyLink, 820
 - gazebo::rendering::Light, 452
- OnRequest
 - gazebo::physics::PhysicsEngine, 630
 - gazebo::physics::SimbodyPhysics, 834
- One
 - gazebo::math::Vector3, 1017
- Open
 - gazebo::util::LogPlay, 485
- OpenAL.hh, 1204
- OpenALSink
 - gazebo::util::OpenALSink, 612
- OpenALSinkPtr
 - gazebo::util, 122
- OpenALSource
 - gazebo::util::OpenALSource, 613
- OpenALSourcePtr
 - gazebo::util, 122
- operator<
 - gazebo::common::Time, 958
 - gazebo::math::Angle, 136
- operator<<
 - gazebo::common::Color, 236
 - gazebo::common::Exception, 333
 - gazebo::common::Material, 506
 - gazebo::common::Time, 964
 - gazebo::common::Timer, 967
 - gazebo::math::Angle, 138
 - gazebo::math::Box, 168
 - gazebo::math::Matrix3, 511
 - gazebo::math::Matrix4, 518
 - gazebo::math::Pose, 656
 - gazebo::math::Quaternion, 687
 - gazebo::math::Vector2d, 995
 - gazebo::math::Vector2i, 1004
 - gazebo::math::Vector3, 1017
 - gazebo::math::Vector4, 1026
 - gazebo::physics::CollisionState, 226
 - gazebo::physics::Inertial, 407
 - gazebo::physics::JointState, 440
 - gazebo::physics::LinkState, 482
 - gazebo::physics::ModelState, 562
 - gazebo::physics::WorldState, 1089
- operator<=
 - gazebo::common::Time, 959
 - gazebo::math::Angle, 136
- operator>
 - gazebo::common::Time, 961, 962
 - gazebo::math::Angle, 137
- operator>>
 - gazebo::common::Color, 236
 - gazebo::common::Time, 965
 - gazebo::math::Angle, 138
 - gazebo::math::Pose, 657
 - gazebo::math::Quaternion, 688
 - gazebo::math::Vector2d, 995

- gazebo::math::Vector2i, 1004
- gazebo::math::Vector3, 1017
- gazebo::math::Vector4, 1027
- operator>=
 - gazebo::common::Time, 962, 963
 - gazebo::math::Angle, 137
- operator*
 - gazebo::common::Color, 231
 - gazebo::common::NodeTransform, 601
 - gazebo::common::Time, 952
 - gazebo::math::Angle, 134
 - gazebo::math::Matrix3, 509, 511
 - gazebo::math::Matrix4, 515, 516
 - gazebo::math::Pose, 653
 - gazebo::math::Quaternion, 682, 683
 - gazebo::math::Vector2d, 990, 991
 - gazebo::math::Vector2i, 999
 - gazebo::math::Vector3, 1012, 1017
 - gazebo::math::Vector4, 1022
- operator*=
 - gazebo::common::Color, 231
 - gazebo::common::Time, 952, 953
 - gazebo::math::Angle, 134
 - gazebo::math::Quaternion, 683
 - gazebo::math::Vector2d, 991
 - gazebo::math::Vector2i, 1000
 - gazebo::math::Vector3, 1012
 - gazebo::math::Vector4, 1023
- operator()
 - gazebo::common::NodeTransform, 601
 - gazebo::event::EventT, 325–327
- operator+
 - gazebo::common::Color, 232
 - gazebo::common::Time, 953, 954
 - gazebo::math::Angle, 135
 - gazebo::math::Box, 167
 - gazebo::math::Matrix3, 509
 - gazebo::math::Pose, 654
 - gazebo::math::Quaternion, 683
 - gazebo::math::Vector2d, 992
 - gazebo::math::Vector2i, 1000
 - gazebo::math::Vector3, 1013
 - gazebo::math::Vector4, 1023
 - gazebo::physics::CollisionState, 225
 - gazebo::physics::Inertial, 404
 - gazebo::physics::JointState, 439
 - gazebo::physics::JointWrench, 443
 - gazebo::physics::LinkState, 481
 - gazebo::physics::ModelState, 561
 - gazebo::physics::WorldState, 1087
- operator+=
 - gazebo::common::Color, 232
 - gazebo::common::Time, 954
 - gazebo::math::Angle, 135
- gazebo::math::Box, 167
- gazebo::math::Pose, 654
- gazebo::math::Quaternion, 683
- gazebo::math::Vector2d, 992
- gazebo::math::Vector2i, 1000
- gazebo::math::Vector3, 1013
- gazebo::math::Vector4, 1023
- gazebo::physics::Inertial, 404
- operator-
 - gazebo::common::Color, 232, 233
 - gazebo::common::Time, 955
 - gazebo::math::Angle, 135
 - gazebo::math::Box, 168
 - gazebo::math::Matrix3, 510
 - gazebo::math::Pose, 654
 - gazebo::math::Quaternion, 684
 - gazebo::math::Vector2d, 992
 - gazebo::math::Vector2i, 1001
 - gazebo::math::Vector3, 1013
 - gazebo::math::Vector4, 1024
 - gazebo::physics::CollisionState, 225
 - gazebo::physics::JointState, 440
 - gazebo::physics::JointWrench, 443
 - gazebo::physics::LinkState, 481
 - gazebo::physics::ModelState, 561
 - gazebo::physics::State, 913
 - gazebo::physics::WorldState, 1087
- operator-=
 - gazebo::common::Color, 233
 - gazebo::common::Time, 955, 956
 - gazebo::math::Angle, 135
 - gazebo::math::Pose, 655
 - gazebo::math::Quaternion, 684
 - gazebo::math::Vector2d, 992
 - gazebo::math::Vector2i, 1001
 - gazebo::math::Vector3, 1014
 - gazebo::math::Vector4, 1024
- operator/
 - gazebo::common::Color, 233
 - gazebo::common::Time, 956, 957
 - gazebo::math::Angle, 136
 - gazebo::math::Vector2d, 992, 993
 - gazebo::math::Vector2i, 1001
 - gazebo::math::Vector3, 1014
 - gazebo::math::Vector4, 1024
- operator/=
 - gazebo::common::Color, 234
 - gazebo::common::Time, 957
 - gazebo::math::Angle, 136
 - gazebo::math::Vector2d, 993
 - gazebo::math::Vector2i, 1002
 - gazebo::math::Vector3, 1014
 - gazebo::math::Vector4, 1025
- operator=

- gazebo::common::Color, 234
- gazebo::common::PID, 638
- gazebo::common::Time, 960
- gazebo::math::Box, 168
- gazebo::math::Matrix4, 516
- gazebo::math::Plane, 641
- gazebo::math::Pose, 655
- gazebo::math::Quaternion, 684
- gazebo::math::Vector2d, 994
- gazebo::math::Vector2i, 1002, 1003
- gazebo::math::Vector3, 1015
- gazebo::math::Vector4, 1025, 1026
- gazebo::physics::CollisionState, 226
- gazebo::physics::Contact, 255
- gazebo::physics::Inertial, 404
- gazebo::physics::JointState, 440
- gazebo::physics::JointWrench, 443
- gazebo::physics::LinkState, 482
- gazebo::physics::ModelState, 562
- gazebo::physics::State, 914
- gazebo::physics::WorldState, 1088
- operator==
 - gazebo::common::Color, 234
 - gazebo::common::Time, 960, 961
 - gazebo::math::Angle, 137
 - gazebo::math::Box, 168
 - gazebo::math::Matrix3, 510
 - gazebo::math::Matrix4, 517
 - gazebo::math::Pose, 655
 - gazebo::math::Quaternion, 685
 - gazebo::math::Vector2d, 994
 - gazebo::math::Vector2i, 1003
 - gazebo::math::Vector3, 1015
 - gazebo::math::Vector4, 1026
 - gazebo::physics::Base, 161
- operator[]
 - gazebo::common::Color, 234
 - gazebo::math::Matrix3, 510
 - gazebo::math::Matrix4, 517
 - gazebo::math::Vector2d, 994
 - gazebo::math::Vector2i, 1003
 - gazebo::math::Vector3, 1015
 - gazebo::math::Vector4, 1026
- OrbitViewController
 - gazebo::rendering::OrbitViewController, 618
- OrbitViewController.hh, 1204
- PHONG
 - gazebo::common::Material, 501
- PID
 - gazebo::common::PID, 637
- PID.hh, 1212
- PIXEL_FORMAT_COUNT
 - gazebo::common::Image, 391
- PLANE_SHAPE
 - gazebo::physics::Base, 157
- POINTS
 - gazebo::common::SubMesh, 919
- PRESS
 - gazebo::common::KeyEvent, 445
 - gazebo::common::MouseEvent, 564
- Param_V
 - gazebo::common, 98
- parent
 - gazebo::common::SkeletonNode, 886
 - gazebo::physics::Base, 164
 - gazebo::rendering::Visual, 1055
- parentEntity
 - gazebo::physics::Entity, 304
 - gazebo::sensors::WirelessTransceiver, 1066
- parentId
 - gazebo::sensors::Sensor, 761
- parentLink
 - gazebo::physics::Joint, 429
- parentLinkPose
 - Joint_TEST::SpawnJointOptions, 898
- parentName
 - gazebo::sensors::Sensor, 761
- ParseArgs
 - gazebo::Server, 770
- parseFloat
 - Math, 50
- parseInt
 - Math, 50
- pathLength
 - gazebo::physics::Actor, 130
- Pause
 - gazebo::util::OpenALSource, 615
- pause
 - gazebo::event::Events, 321
- pause_incoming
 - Transport, 89
- pause_world
 - Classes for physics and dynamics, 68
- pause_worlds
 - Classes for physics and dynamics, 68
- PauseIncoming
 - gazebo::transport::TopicManager, 972
- PhysicsEngine
 - gazebo::physics::PhysicsEngine, 624
- physicsEngine
 - Joint_TEST, 434
- PhysicsEngine.hh, 1205
- PhysicsEnginePtr
 - gazebo::physics, 110
- PhysicsFactory.hh, 1206
- PhysicsFactoryFn
 - Classes for physics and dynamics, 67

- PhysicsIface.hh, 1207
- physicsInitialized
 - gazebo::physics::SimbodyJoint, 813
 - gazebo::physics::SimbodyLink, 822
- physicsSub
 - gazebo::physics::PhysicsEngine, 633
- PhysicsTypes.hh, 1209
 - GZ_ALL_COLLIDE, 1212
 - GZ_FIXED_COLLIDE, 1212
 - GZ_GHOST_COLLIDE, 1212
 - GZ_NONE_COLLIDE, 1212
 - GZ_SENSOR_COLLIDE, 1212
- physicsUpdateMutex
 - gazebo::physics::PhysicsEngine, 634
- Pi
 - gazebo::math::Angle, 139
- pitchNode
 - gazebo::rendering::Camera, 204
- PixelFormat
 - gazebo::common::Image, 391
- PixelFormatNames
 - Common, 41
- PlaceOnEntity
 - gazebo::physics::Entity, 301
- PlaceOnNearestEntityBelow
 - gazebo::physics::Entity, 301
- placeable
 - gazebo::physics::Collision, 222
- Plane
 - gazebo::math::Plane, 641
- Plane.hh, 1213
- PlaneShape
 - gazebo::physics::PlaneShape, 644
- PlaneShape.hh, 1214
- Play
 - gazebo::physics::Actor, 128
 - gazebo::util::OpenALSource, 615
- playStartTime
 - gazebo::physics::Actor, 130
- Plugin.hh, 1215
 - GZ_REGISTER_MODEL_PLUGIN, 1217
 - GZ_REGISTER_SENSOR_PLUGIN, 1218
 - GZ_REGISTER_SYSTEM_PLUGIN, 1218
 - GZ_REGISTER_VISUAL_PLUGIN, 1218
 - GZ_REGISTER_WORLD_PLUGIN, 1219
- pluginPathsFromEnv
 - gazebo::common::SystemPaths, 942
- PluginT
 - gazebo::PluginT, 647
- PluginType
 - Common, 36
- plugins
 - gazebo::sensors::Sensor, 761
- pointSize
 - gazebo::common::Material, 506
- points
 - gazebo::math::RotationSpline, 724
 - gazebo::math::Spline, 910
- pos
 - gazebo::common::MouseEvent, 565
 - gazebo::math::Pose, 657
- Pose
 - gazebo::math::Pose, 650, 651
- pose
 - gazebo::sensors::Sensor, 761
- Pose.hh, 1219
- Pose2Transform
 - gazebo::physics::SimbodyPhysics, 834
- PoseAnimation
 - gazebo::common::PoseAnimation, 658
- PoseAnimationPtr
 - gazebo::common, 98
- PoseKeyFrame
 - gazebo::common::PoseKeyFrame, 661
- poseSub
 - gazebo::sensors::Sensor, 761
- positions
 - gazebo::physics::Contact, 256
- PostRender
 - gazebo::rendering::Camera, 197
 - gazebo::rendering::DepthCamera, 276
 - gazebo::rendering::GpuLaser, 350
 - gazebo::rendering::UserCamera, 985
- postRender
 - gazebo::event::Events, 321
- power
 - gazebo::sensors::WirelessTransceiver, 1066
- PreLoad
 - gazebo::Server, 770
- PreRender
 - gazebo::rendering::Scene, 742
- preRender
 - gazebo::event::Events, 321
- precision
 - Math, 50
- PrepareHardwareBuffers
 - gazebo::rendering::DynamicRenderable, 293
- prepareProceduralPage
 - gazebo::rendering::DummyPageProvider, 285
- pressPos
 - gazebo::common::MouseEvent, 565
- prevAnimTime
 - gazebo::rendering::Camera, 205
- prevAnimationTime
 - gazebo::physics::Entity, 304
- prevFrameTime
 - gazebo::physics::Actor, 130
- prevPos

- gazebo::common::MouseEvent, 565
- PrimitiveType
 - gazebo::common::SubMesh, 919
- Print
 - gazebo::common::Exception, 333
 - gazebo::physics::Base, 161
- print_version
 - gazebo, 95
- PrintEntityTree
 - gazebo::physics::World, 1079
- PrintSceneGraph
 - gazebo::rendering::Scene, 742
- PrintSource
 - gazebo::common::NodeTransform, 601
- PrintTransforms
 - gazebo::common::Skeleton, 871
- PrintUsage
 - gazebo::Server, 770
- ProcessIncoming
 - gazebo::transport::Node, 591
- ProcessMsg
 - gazebo::physics::BoxShape, 171
 - gazebo::physics::Collision, 220
 - gazebo::physics::CylinderShape, 271
 - gazebo::physics::HeightmapShape, 385
 - gazebo::physics::Inertial, 404
 - gazebo::physics::Link, 470
 - gazebo::physics::MapShape, 496
 - gazebo::physics::MeshShape, 536
 - gazebo::physics::Model, 547
 - gazebo::physics::MultiRayShape, 585
 - gazebo::physics::PlaneShape, 645
 - gazebo::physics::RayShape, 704
 - gazebo::physics::Shape, 777
 - gazebo::physics::SphereShape, 900
 - gazebo::physics::SurfaceParams, 935
- ProcessNodes
 - gazebo::transport::TopicManager, 972
- ProcessPublishers
 - gazebo::transport::Node, 591
- ProcessWriteQueue
 - gazebo::transport::Connection, 246
- Projector
 - gazebo::rendering::Projector, 663
- Projector.hh, 1220
- provideFeedback
 - gazebo::physics::Joint, 429
- pub
 - gazebo::sensors::WirelessTransceiver, 1066
- Publication
 - gazebo::transport::Publication, 665
- Publication.hh, 1220
- PublicationPtr
 - gazebo::transport, 121
- PublicationTransport
 - gazebo::transport::PublicationTransport, 669
- PublicationTransport.hh, 1223
- PublicationTransportPtr
 - gazebo::transport, 121
- Publish
 - gazebo::transport::Node, 592
 - gazebo::transport::Publication, 668
 - gazebo::transport::Publisher, 673
 - gazebo::transport::TopicManager, 972
- publish
 - Transport, 89
- PublishContacts
 - gazebo::physics::ContactManager, 259
- PublishModelPose
 - gazebo::physics::World, 1079
- Publisher
 - gazebo::transport::Publisher, 672
- publisher
 - gazebo::physics::ContactPublisher, 260
- Publisher.hh, 1225
- PublisherPtr
 - gazebo::transport, 121
- Purple
 - gazebo::common::Color, 237
- QuadNode, 674
- QuadToQuad
 - gazebo::physics::SimbodyPhysics, 835
- Quaternion
 - gazebo::math::Quaternion, 678
- Quaternion.hh, 1227
- r
 - gazebo::common::Color, 237
- R_FLOAT16
 - gazebo::common::Image, 391
- R_FLOAT32
 - gazebo::common::Image, 391
- RAY
 - gazebo::sensors, 119
- RAY_SHAPE
 - gazebo::physics::Base, 157
- RELEASE
 - gazebo::common::KeyEvent, 445
 - gazebo::common::MouseEvent, 564
- RENDER_PATH_COUNT
 - gazebo::rendering::RenderEngine, 707
- RENDERING_LINE_LIST
 - gazebo::rendering, 115
- RENDERING_LINE_STRIP
 - gazebo::rendering, 115
- RENDERING_MESH_RESOURCE
 - gazebo::rendering, 115
- RENDERING_POINT_LIST

- gazebo::rendering, 115
- RENDERING_TRIANGLE_FAN
 - gazebo::rendering, 115
- RENDERING_TRIANGLE_LIST
 - gazebo::rendering, 115
- RENDERING_TRIANGLE_STRIP
 - gazebo::rendering, 115
- REPLACE
 - gazebo::common::Material, 500
- RFIDSensor
 - gazebo::sensors::RFIDSensor, 711
- RFIDSensor.hh, 1236
- RFIDSensor_V
 - gazebo::sensors, 118
- RFIDSensorPtr
 - gazebo::sensors, 118
- RFIDTag
 - gazebo::sensors::RFIDTag, 714
- RFIDTag.hh, 1237
- RFIDTag_V
 - gazebo::sensors, 118
- RFIDTagPtr
 - gazebo::sensors, 118
- RFIDTagVisual
 - gazebo::rendering::RFIDTagVisual, 716
- RFIDTagVisual.hh, 1237
- RFIDTagVisualPtr
 - gazebo::rendering, 114
- RFIDVisual
 - gazebo::rendering::RFIDVisual, 717
- RFIDVisual.hh, 1238
- RFIDVisualPtr
 - gazebo::rendering, 114
- RGB_FLOAT16
 - gazebo::common::Image, 391
- RGB_FLOAT32
 - gazebo::common::Image, 391
- RGB_INT16
 - gazebo::common::Image, 391
- RGB_INT32
 - gazebo::common::Image, 391
- RGB_INT8
 - gazebo::common::Image, 391
- RGBA
 - gazebo::common::Color, 229
- RGBA_INT8
 - gazebo::common::Image, 391
- RIGHT
 - gazebo::common::MouseEvent, 564
- ROT
 - Rendering, 76
- ROT_X
 - Rendering, 76
- ROT_Y
 - Rendering, 76
- ROT_Z
 - Rendering, 76
- ROTATE
 - gazebo::common::NodeTransform, 600
- RTShaderSystem.hh, 1242
- Radian
 - gazebo::math::Angle, 137
- Raise
 - gazebo::rendering::Heightmap, 379
- Rand.hh, 1228
- rangeCountRatio
 - gazebo::sensors::GpuRaySensor, 364
- rangeElem
 - gazebo::physics::MultiRayShape, 586
 - gazebo::sensors::GpuRaySensor, 364
- RawCallbackHelper
 - gazebo::transport::RawCallbackHelper, 691
- rawNW
 - gazebo::common::Skeleton, 872
- RawNodeAnim
 - gazebo::common, 98
- RawNodeWeights
 - gazebo::common, 98
- RawSkeletonAnim
 - gazebo::common, 98
- rawTransforms
 - gazebo::common::SkeletonNode, 886
- rayCountRatio
 - gazebo::rendering::GpuLaser, 353
- rayElem
 - gazebo::physics::MultiRayShape, 586
- RaySensor
 - gazebo::sensors::RaySensor, 695
- RaySensor.hh, 1229
- RaySensor_V
 - gazebo::sensors, 118
- RaySensorPtr
 - gazebo::sensors, 118
- RayShape
 - gazebo::physics::RayShape, 702
- RayShape.hh, 1230
- RayShapePtr
 - gazebo::physics, 110
- rays
 - gazebo::physics::MultiRayShape, 586
- Read
 - gazebo::transport::Connection, 246
- ReadCallback
 - gazebo::transport::Connection, 243
- ReadPixelBuffer
 - gazebo::rendering::Camera, 198
- realTime
 - gazebo::common::UpdateInfo, 978

- gazebo::physics::State, 915
- realTimeUpdateRate
 - gazebo::physics::PhysicsEngine, 634
- RecalcTangents
 - gazebo::math::RotationSpline, 723
 - gazebo::math::Spline, 908
- RecalculateMatrix
 - gazebo::common::NodeTransform, 602
- RecalculateNormals
 - gazebo::common::Mesh, 525
 - gazebo::common::SubMesh, 924
- Red
 - gazebo::common::Color, 237
- referencePose
 - gazebo::sensors::WirelessTransceiver, 1066
- RegisterAll
 - gazebo::physics::PhysicsFactory, 635
 - gazebo::sensors::SensorFactory, 763
- RegisterMsg
 - gazebo::msgs::MsgFactory, 573
- RegisterPhysicsEngine
 - gazebo::physics::PhysicsFactory, 635
- RegisterSensor
 - gazebo::sensors::SensorFactory, 763
- RegisterTopicNamespace
 - gazebo::transport::ConnectionManager, 250
 - gazebo::transport::TopicManager, 973
- relativeEndPos
 - gazebo::physics::RayShape, 705
- relativeStartPos
 - gazebo::physics::RayShape, 705
- Remove
 - gazebo::util::LogRecord, 491
- remove_scene
 - Rendering, 78
- remove_sensor
 - Sensors, 83
- remove_sensors
 - Sensors, 83
- remove_worlds
 - Classes for physics and dynamics, 69
- RemoveCallback
 - gazebo::transport::Node, 592
- RemoveCamera
 - gazebo::rendering::Scene, 742
- RemoveChild
 - gazebo::physics::Base, 162
 - gazebo::physics::Link, 470
 - gazebo::physics::Model, 547
- RemoveChildJoint
 - gazebo::physics::Link, 470
- RemoveChildren
 - gazebo::physics::Base, 162
- RemoveCollision
 - gazebo::physics::Link, 470
- RemoveConnection
 - gazebo::transport::ConnectionManager, 250
- RemoveNode
 - gazebo::transport::TopicManager, 973
- RemoveParentJoint
 - gazebo::physics::Link, 470
- RemovePlugin
 - gazebo::physics::World, 1079
 - gazebo::rendering::Visual, 1049
- RemoveScene
 - gazebo::rendering::RenderEngine, 709
 - gazebo::rendering::RTShaderSystem, 727
- removeScene
 - gazebo::rendering::Events, 310
- RemoveSensor
 - gazebo::sensors::SensorManager, 766
- RemoveSensors
 - gazebo::sensors::SensorManager, 766
- RemoveShadows
 - gazebo::rendering::RTShaderSystem, 728
- RemoveSubscription
 - gazebo::transport::Publication, 668
- RemoveTransport
 - gazebo::transport::Publication, 668
- RemoveVisual
 - gazebo::rendering::Scene, 742
- Render
 - gazebo::rendering::Camera, 198
- render
 - gazebo::event::Events, 321
- RenderEngine.hh, 1231
- RenderEvents.hh, 1232
- RenderImpl
 - gazebo::rendering::Camera, 198
- RenderOpType
 - gazebo::rendering, 115
- RenderPathType
 - gazebo::rendering::RenderEngine, 707
- renderTarget
 - gazebo::rendering::Camera, 205
- renderTexture
 - gazebo::rendering::Camera, 205
- RenderTypes.hh, 1234
 - GZ_VISIBILITY_ALL, 1236
 - GZ_VISIBILITY_GUI, 1236
 - GZ_VISIBILITY_SELECTABLE, 1236
 - GZ_VISIBILITY_SELECTION, 1236
- Rendering, 73
 - ~SelectionObj, 76
 - Attach, 76
 - create_scene, 77
 - Detach, 77
 - fini, 77

- get_scene, 77
- GetMode, 77
- GetState, 77
- init, 77
- Load, 78
- load, 77
- ROT, 76
- ROT_X, 76
- ROT_Y, 76
- ROT_Z, 76
- remove_scene, 78
- SCALE, 76
- SCALE_X, 76
- SCALE_Y, 76
- SCALE_Z, 76
- SELECTION_NONE, 76
- SelectionMode, 76
- SelectionObj, 76
- SetGlobal, 78
- SetMode, 78
- SetState, 78
- TRANS, 76
- TRANS_X, 76
- TRANS_Y, 76
- TRANS_Z, 76
- UpdateSize, 79
- RenderingIface.hh, 1233
- request
 - Transport, 89
- requestNoReply
 - Transport, 89, 90
- requestPub
 - gazebo::physics::Entity, 304
- requestSub
 - gazebo::physics::PhysicsEngine, 634
- requests
 - gazebo::rendering::Camera, 205
- Rescale
 - gazebo::common::Image, 394
- Reset
 - gazebo::common::Color, 235
 - gazebo::common::PID, 638
 - gazebo::common::SkeletonNode, 883
 - gazebo::math::Pose, 655
 - gazebo::ModelPlugin, 554
 - gazebo::physics::Base, 162
 - gazebo::physics::Contact, 255
 - gazebo::physics::Entity, 301
 - gazebo::physics::Inertial, 405
 - gazebo::physics::Joint, 425
 - gazebo::physics::JointController, 435
 - gazebo::physics::Link, 470
 - gazebo::physics::Model, 547
 - gazebo::physics::PhysicsEngine, 630
 - gazebo::physics::SimbodyJoint, 810
 - gazebo::physics::SimbodyPhysics, 835
 - gazebo::physics::World, 1079
 - gazebo::SensorPlugin, 769
 - gazebo::SystemPlugin, 944
 - gazebo::VisualPlugin, 1056
 - gazebo::WorldPlugin, 1083
- ResetCount
 - gazebo::physics::ContactManager, 259
- ResetEntities
 - gazebo::physics::World, 1079
- ResetLastUpdateTime
 - gazebo::sensors::Sensor, 759
- ResetLastUpdateTimes
 - gazebo::sensors::SensorManager, 766
- ResetPhysicsStates
 - gazebo::physics::Link, 471
- ResetTime
 - gazebo::physics::World, 1080
- Resize
 - gazebo::rendering::GUIOverlay, 374
 - gazebo::rendering::UserCamera, 985
 - gazebo::rendering::WindowManager, 1059
- responsePub
 - gazebo::physics::PhysicsEngine, 634
- RestoreSimbodyState
 - gazebo::physics::SimbodyHingeJoint, 803
 - gazebo::physics::SimbodyJoint, 810
 - gazebo::physics::SimbodyLink, 820
- Rewind
 - gazebo::util::OpenALSource, 615
- Road, 718
 - gazebo::physics::Road, 719
- Road.hh, 1239
- Road2d
 - gazebo::rendering::Road2d, 720
- Road2d.hh, 1240
- RoadPtr
 - gazebo::physics, 110
- root
 - gazebo::common::Skeleton, 872
 - gazebo::rendering::RenderEngine, 710
- rot
 - gazebo::math::Pose, 657
- Rotate
 - gazebo::physics::Inertial, 405
- rotate
 - gazebo::common::PoseKeyFrame, 662
- RotatePitch
 - gazebo::rendering::Camera, 198
- RotatePositionAboutOrigin
 - gazebo::math::Pose, 655
- RotateVector
 - gazebo::math::Quaternion, 685

- RotateVectorReverse
 - gazebo::math::Quaternion, 685
- RotateYaw
 - gazebo::rendering::Camera, 198
- RotationSpline
 - gazebo::math::RotationSpline, 721
- RotationSpline.hh, 1240
- Round
 - gazebo::math::Pose, 656
 - gazebo::math::Quaternion, 685
 - gazebo::math::Vector3, 1016
- Run
 - gazebo::Master, 497
 - gazebo::physics::World, 1080
 - gazebo::Server, 770
 - gazebo::transport::ConnectionManager, 250
- run
 - gazebo, 95
 - Transport, 90
- run_once
 - Sensors, 83
- run_threads
 - Sensors, 84
- run_world
 - Classes for physics and dynamics, 69
- run_worlds
 - Classes for physics and dynamics, 69
- RunOnce
 - gazebo::Master, 497
- RunThread
 - gazebo::Master, 497
- RunThreads
 - gazebo::sensors::SensorManager, 767
- SCALE
 - gazebo::common::NodeTransform, 600
 - Rendering, 76
- SCALE_X
 - Rendering, 76
- SCALE_Y
 - Rendering, 76
- SCALE_Z
 - Rendering, 76
- SCREW_JOINT
 - gazebo::physics::Base, 156
- SCROLL
 - gazebo::common::MouseEvent, 564
- SELECTION_NONE
 - Rendering, 76
- SENSOR_COLLISION
 - gazebo::physics::Base, 157
- SENSOR_PLUGIN
 - Common, 36
- SHADE_COUNT
 - gazebo::common::Material, 501
- SHAPE
 - gazebo::physics::Base, 156
- SLIDER_JOINT
 - gazebo::physics::Base, 156
- SM2Profile
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 890
- SPHERE_SHAPE
 - gazebo::physics::Base, 157
- SSLM_NormalMapLightingObjectSpace
 - gazebo::rendering::RTShaderSystem, 726
- SSLM_NormalMapLightingTangentSpace
 - gazebo::rendering::RTShaderSystem, 726
- SSLM_PerPixelLighting
 - gazebo::rendering::RTShaderSystem, 726
- SSLM_PerVertexLighting
 - gazebo::rendering::RTShaderSystem, 726
- STLLoader
 - gazebo::common::STLLoader, 916
- STLLoader.hh, 1279
 - COR3_MAX, 1280
 - FACE_MAX, 1280
 - LINE_MAX_LEN, 1280
 - ORDER_MAX, 1280
- STOP_CFM
 - gazebo::physics::Joint, 415
- STOP_ERP
 - gazebo::physics::Joint, 415
- SUSPENSION_CFM
 - gazebo::physics::Joint, 415
- SUSPENSION_ERP
 - gazebo::physics::Joint, 415
- SYSTEM_PLUGIN
 - Common, 36
- Save
 - gazebo::physics::World, 1080
- saveCount
 - gazebo::rendering::Camera, 205
- SaveFrame
 - gazebo::rendering::Camera, 198, 199
 - gazebo::sensors::CameraSensor, 209
 - gazebo::sensors::DepthCameraSensor, 279
 - gazebo::sensors::MultiCameraSensor, 577
- saveFrameBuffer
 - gazebo::rendering::Camera, 205
- SavePNG
 - gazebo::common::Image, 394
- SaveSimbodyState
 - gazebo::physics::SimbodyHingeJoint, 803
 - gazebo::physics::SimbodyJoint, 810
 - gazebo::physics::SimbodyLink, 820
- Scale
 - gazebo::common::Mesh, 525

- gazebo::common::NodeAnimation, 597
- gazebo::common::Skeleton, 871
- gazebo::common::SkeletonAnimation, 876
- gazebo::common::SubMesh, 924
- gazebo::math::Quaternion, 685
- scale
 - gazebo::physics::Entity, 304
 - gazebo::physics::Shape, 778
- ScaleXAxis
 - gazebo::rendering::AxisVisual, 150
- ScaleYAxis
 - gazebo::rendering::AxisVisual, 150
- ScaleZAxis
 - gazebo::rendering::AxisVisual, 150
- scanElem
 - gazebo::physics::MultiRayShape, 586
 - gazebo::sensors::GpuRaySensor, 364
- Scene
 - gazebo::rendering::Scene, 733
- scene
 - gazebo::rendering::Camera, 205
 - gazebo::rendering::Visual, 1055
 - gazebo::sensors::Sensor, 762
- Scene.hh, 1242
- SceneFromSDF
 - Messages, 59
- sceneNode
 - gazebo::rendering::Camera, 205
 - gazebo::rendering::Visual, 1055
- ScenePtr
 - gazebo::rendering, 114
- screenshotPath
 - gazebo::rendering::Camera, 205
- ScrewJoint
 - gazebo::physics::ScrewJoint, 747
- ScrewJoint.hh, 1244
- scriptLength
 - gazebo::physics::Actor, 130
- scroll
 - gazebo::common::MouseEvent, 565
- sdf
 - gazebo::physics::Base, 164
 - gazebo::physics::PhysicsEngine, 634
 - gazebo::rendering::Camera, 205
 - gazebo::sensors::Sensor, 762
- sec
 - gazebo::common::Time, 965
- SecToNano
 - gazebo::common::Time, 963
- SelectVisual
 - gazebo::rendering::Scene, 742
- SelectionMode
 - Rendering, 76
- SelectionObj
 - Rendering, 76
- SelectionObj.hh, 1245
- SelectionObjPtr
 - gazebo::rendering, 114
- SendMessage
 - gazebo::transport::Publisher, 673
- Sensor
 - gazebo::sensors::Sensor, 754
- Sensor.hh, 1245
- Sensor_V
 - gazebo::sensors, 118
- SensorCategory
 - gazebo::sensors, 119
- SensorFactor, 762
- SensorFactory.hh, 1247
- SensorFactoryFn
 - gazebo::sensors, 118
- SensorManager.hh, 1248
- SensorPlugin
 - gazebo::SensorPlugin, 768
- SensorPluginPtr
 - gazebo, 95
- SensorPtr
 - gazebo::sensors, 118
- SensorTypes.hh, 1250
- Sensors, 80
 - create_sensor, 82
 - fini, 82
 - GZ_REGISTER_STATIC_SENSOR, 82
 - get_sensor, 82
 - init, 83
 - load, 83
 - remove_sensor, 83
 - remove_sensors, 83
 - run_once, 83
 - run_threads, 84
 - stop, 84
- SensorsIface.hh, 1248
- SensorsInitialized
 - gazebo::sensors::SensorManager, 767
- Server
 - gazebo::Server, 770
- Server.hh, 1252
- Set
 - gazebo::common::Color, 235
 - gazebo::common::NodeTransform, 602
 - gazebo::common::Time, 964
 - gazebo::math::Matrix4, 517
 - gazebo::math::Plane, 642
 - gazebo::math::Pose, 656
 - gazebo::math::Quaternion, 686
 - gazebo::math::Vector2d, 995
 - gazebo::math::Vector2i, 1003
 - gazebo::math::Vector3, 1016

- gazebo::math::Vector4, 1026
- Messages, 59–61
- SetActive
 - gazebo::sensors::DepthCameraSensor, 279
 - gazebo::sensors::Sensor, 759
- SetAltitude
 - gazebo::physics::PlaneShape, 645
 - gazebo::physics::SimbodyPlaneShape, 839
- SetAmbient
 - gazebo::common::Material, 503
 - gazebo::rendering::Visual, 1049
- SetAmbientColor
 - gazebo::rendering::Scene, 743
- SetAnchor
 - gazebo::physics::Joint, 425
 - gazebo::physics::ScrewJoint, 748
 - gazebo::physics::SimbodyJoint, 810
 - gazebo::physics::SliderJoint, 888
- SetAngle
 - gazebo::physics::Joint, 425
- SetAngleMax
 - gazebo::sensors::GpuRaySensor, 363
- SetAngleMin
 - gazebo::sensors::GpuRaySensor, 363
- SetAngularAccel
 - gazebo::physics::Link, 471
 - gazebo::physics::Model, 547
- SetAngularDamping
 - gazebo::physics::Link, 471
 - gazebo::physics::SimbodyLink, 820
- SetAngularVel
 - gazebo::physics::Link, 471
 - gazebo::physics::Model, 548
 - gazebo::physics::SimbodyLink, 820
- SetAnimation
 - gazebo::physics::Entity, 301, 302
- SetAspectRatio
 - gazebo::rendering::Camera, 199
- SetAttenuation
 - gazebo::rendering::Light, 452
- SetAttribute
 - gazebo::physics::Joint, 425
 - gazebo::physics::SimbodyJoint, 810, 811
- SetAutoCalculate
 - gazebo::math::RotationSpline, 723
 - gazebo::math::Spline, 909
- SetAutoDisable
 - gazebo::physics::Link, 471
 - gazebo::physics::Model, 548
 - gazebo::physics::SimbodyLink, 820
- SetAutoDisableFlag
 - gazebo::physics::PhysicsEngine, 630
- SetAxis
 - gazebo::physics::BallJoint, 153
 - gazebo::physics::Joint, 426
 - gazebo::physics::SimbodyHinge2Joint, 797
 - gazebo::physics::SimbodyHingeJoint, 803
 - gazebo::physics::SimbodyJoint, 811
 - gazebo::physics::SimbodyScrewJoint, 847
 - gazebo::physics::SimbodySliderJoint, 853
 - gazebo::physics::SimbodyUniversalJoint, 862
- SetAxisMaterial
 - gazebo::rendering::AxisVisual, 151
- SetBackgroundColor
 - gazebo::rendering::Scene, 743
- SetBasePath
 - gazebo::util::LogRecord, 491
- SetBaseline
 - gazebo::rendering::MovableText, 570
- SetBindShapeTransform
 - gazebo::common::Skeleton, 871
- SetBlendFactors
 - gazebo::common::Material, 504
- SetBlendMode
 - gazebo::common::Material, 504
- SetCallbackId
 - gazebo::transport::Subscriber, 930
- SetCamera
 - gazebo::rendering::WindowManager, 1059
- SetCameraCount
 - gazebo::rendering::GpuLaser, 350
- SetCanonicalLink
 - gazebo::physics::Entity, 302
- SetCaptureData
 - gazebo::rendering::Camera, 199
- SetCaptureDataOnce
 - gazebo::rendering::Camera, 199
- SetCastShadows
 - gazebo::rendering::Light, 452
 - gazebo::rendering::Visual, 1049
- SetCategoryBits
 - gazebo::physics::Collision, 220
 - gazebo::physics::SimbodyCollision, 788
- SetCellCount
 - gazebo::rendering::Grid, 367
- SetCellLength
 - gazebo::rendering::Grid, 367
- SetCharHeight
 - gazebo::rendering::MovableText, 571
- SetClipDist
 - gazebo::rendering::Camera, 199
- SetCmd
 - gazebo::common::PID, 638
- SetCmdMax
 - gazebo::common::PID, 638
- SetCmdMin
 - gazebo::common::PID, 638
- SetCoG

- gazebo::physics::Inertial, 405
- SetCol
 - gazebo::math::Matrix3, 510
- SetCollideBits
 - gazebo::physics::Collision, 220
 - gazebo::physics::SimbodyCollision, 788
- SetCollideMode
 - gazebo::physics::Link, 471
 - gazebo::physics::Model, 548
- SetCollision
 - gazebo::physics::Collision, 221
- SetCollisionShape
 - gazebo::physics::SimbodyCollision, 788
- SetColor
 - gazebo::rendering::DynamicLines, 289
 - gazebo::rendering::Grid, 368
 - gazebo::rendering::MovableText, 571
- SetComponent
 - gazebo::common::NodeTransform, 602
- SetContactMaxCorrectingVel
 - gazebo::physics::PhysicsEngine, 630
- SetContactSurfaceLayer
 - gazebo::physics::PhysicsEngine, 630
- SetContactsEnabled
 - gazebo::physics::Collision, 221
- SetCosHorzFOV
 - gazebo::rendering::GpuLaser, 350
- SetCosVertFOV
 - gazebo::rendering::GpuLaser, 350
- SetDGain
 - gazebo::common::PID, 639
- SetDamping
 - gazebo::physics::Joint, 426
 - gazebo::physics::SimbodyBallJoint, 782
 - gazebo::physics::SimbodyHinge2Joint, 797
 - gazebo::physics::SimbodyHingeJoint, 803
 - gazebo::physics::SimbodyJoint, 811
 - gazebo::physics::SimbodyScrewJoint, 847
 - gazebo::physics::SimbodySliderJoint, 854
 - gazebo::physics::SimbodyUniversalJoint, 862
- SetDepthTarget
 - gazebo::rendering::DepthCamera, 276
- SetDepthWrite
 - gazebo::common::Material, 504
- SetDiffuse
 - gazebo::common::Material, 504
 - gazebo::rendering::Visual, 1049
- SetDiffuseColor
 - gazebo::rendering::Light, 452
- SetDirection
 - gazebo::rendering::Light, 453
- SetDirtyPose
 - gazebo::physics::SimbodyLink, 821
- SetDistance
 - gazebo::rendering::OrbitViewController, 620
- SetElevationReference
 - gazebo::common::SphericalCoordinates, 905
- SetEmissive
 - gazebo::common::Material, 504
 - gazebo::rendering::LaserVisual, 448
 - gazebo::rendering::Visual, 1050
- SetEnabled
 - gazebo::physics::Link, 472
 - gazebo::physics::Model, 548
 - gazebo::physics::SimbodyLink, 821
 - gazebo::rendering::ContactVisual, 266
 - gazebo::rendering::Projector, 664
 - gazebo::rendering::ViewController, 1033
 - gazebo::rendering::WrenchVisual, 1091
- SetFarClip
 - gazebo::rendering::GpuLaser, 351
- SetFiducial
 - gazebo::physics::RayShape, 704
- SetFile
 - gazebo::common::AudioDecoder, 148
- SetFocalPoint
 - gazebo::rendering::OrbitViewController, 620
 - gazebo::rendering::UserCamera, 986
- SetFog
 - gazebo::rendering::Scene, 743
- SetFontName
 - gazebo::rendering::MovableText, 571
- SetForce
 - gazebo::physics::Joint, 426
 - gazebo::physics::Link, 472
 - gazebo::physics::SimbodyJoint, 811
 - gazebo::physics::SimbodyLink, 821
- SetForceImpl
 - gazebo::physics::SimbodyBallJoint, 782
 - gazebo::physics::SimbodyHinge2Joint, 797
 - gazebo::physics::SimbodyHingeJoint, 803
 - gazebo::physics::SimbodyJoint, 812
 - gazebo::physics::SimbodyScrewJoint, 847
 - gazebo::physics::SimbodySliderJoint, 854
 - gazebo::physics::SimbodyUniversalJoint, 862
- SetFromABGR
 - gazebo::common::Color, 235
- SetFromARGB
 - gazebo::common::Color, 235
- SetFromAxes
 - gazebo::math::Matrix3, 511
- SetFromAxis
 - gazebo::math::Matrix3, 511
 - gazebo::math::Quaternion, 686
- SetFromBGRA
 - gazebo::common::Color, 235
- SetFromData
 - gazebo::common::Image, 394

- SetFromDegree
 - gazebo::math::Angle, 138
- SetFromEuler
 - gazebo::math::Quaternion, 686
- SetFromHSV
 - gazebo::common::Color, 235
- SetFromRGBA
 - gazebo::common::Color, 236
- SetFromRadian
 - gazebo::math::Angle, 138
- SetFromYUV
 - gazebo::common::Color, 236
- SetGain
 - gazebo::util::OpenALSource, 615
- SetGlobal
 - Rendering, 78
- SetGravity
 - gazebo::physics::PhysicsEngine, 631
 - gazebo::physics::SimbodyPhysics, 835
- SetGravityMode
 - gazebo::physics::Link, 472
 - gazebo::physics::Model, 548
 - gazebo::physics::SimbodyLink, 821
- SetGrid
 - gazebo::rendering::Scene, 743
- SetHFOV
 - gazebo::rendering::Camera, 200
- SetHandle
 - gazebo::common::SkeletonNode, 883
- SetHeadingOffset
 - gazebo::common::SphericalCoordinates, 905
- SetHeight
 - gazebo::rendering::Grid, 368
- SetHighStop
 - gazebo::physics::BallJoint, 153
 - gazebo::physics::Joint, 426
 - gazebo::physics::SimbodyBallJoint, 783
 - gazebo::physics::SimbodyHinge2Joint, 797
 - gazebo::physics::SimbodyHingeJoint, 804
 - gazebo::physics::SimbodyScrewJoint, 848
 - gazebo::physics::SimbodySliderJoint, 854
 - gazebo::physics::SimbodyUniversalJoint, 863
- SetHighlighted
 - gazebo::rendering::Visual, 1050
- SetHorzFOV
 - gazebo::rendering::GpuLaser, 351
- SetHorzHalfAngle
 - gazebo::rendering::GpuLaser, 351
- SetIGain
 - gazebo::common::PID, 639
- SetIMax
 - gazebo::common::PID, 639
- SetIMin
 - gazebo::common::PID, 639
- SetIXX
 - gazebo::physics::Inertial, 406
- SetIXY
 - gazebo::physics::Inertial, 406
- SetIXZ
 - gazebo::physics::Inertial, 406
- SetIYY
 - gazebo::physics::Inertial, 406
- SetIYZ
 - gazebo::physics::Inertial, 407
- SetIZZ
 - gazebo::physics::Inertial, 407
- SetId
 - gazebo::common::SkeletonNode, 883
 - gazebo::rendering::Visual, 1050
- SetImageHeight
 - gazebo::rendering::Camera, 200
- SetImageSize
 - gazebo::rendering::Camera, 200
- SetImageWidth
 - gazebo::rendering::Camera, 200
- SetIndexCount
 - gazebo::common::SubMesh, 924
- SetInertiaMatrix
 - gazebo::physics::Inertial, 406
- SetInertial
 - gazebo::physics::Link, 472
- SetInitialRelativePose
 - gazebo::physics::Entity, 302
- SetInitialTransform
 - gazebo::common::SkeletonNode, 884
- SetInverseBindTransform
 - gazebo::common::SkeletonNode, 884
- SetIsHorizontal
 - gazebo::rendering::GpuLaser, 351
- SetJointAnimation
 - gazebo::physics::Model, 548
- SetJointPosition
 - gazebo::physics::JointController, 435
 - gazebo::physics::Model, 549
- SetJointPositions
 - gazebo::physics::JointController, 435
 - gazebo::physics::Model, 549
- SetKinematic
 - gazebo::physics::Link, 472
- SetLaserRetro
 - gazebo::physics::Collision, 221
 - gazebo::physics::Link, 473
 - gazebo::physics::Model, 549
- SetLatitudeReference
 - gazebo::common::SphericalCoordinates, 905
- SetLength
 - gazebo::common::Animation, 142
 - gazebo::physics::CylinderShape, 271

- gazebo::physics::RayShape, 704
- SetLightType
 - gazebo::rendering::Light, 453
- SetLighting
 - gazebo::common::Material, 505
- SetLineWidth
 - gazebo::rendering::Grid, 368
- SetLinearAccel
 - gazebo::physics::Link, 473
 - gazebo::physics::Model, 549
- SetLinearDamping
 - gazebo::physics::Link, 473
 - gazebo::physics::SimbodyLink, 821
- SetLinearVel
 - gazebo::physics::Link, 473
 - gazebo::physics::Model, 550
 - gazebo::physics::SimbodyLink, 821
- SetLinkStatic
 - gazebo::physics::Link, 473
 - gazebo::physics::SimbodyLink, 822
- SetLinkWorldPose
 - gazebo::physics::Model, 550
- SetLocallyAdvertised
 - gazebo::transport::Publication, 668
- SetLongitudeReference
 - gazebo::common::SphericalCoordinates, 905
- SetLoop
 - gazebo::util::OpenALSource, 616
- SetLowStop
 - gazebo::physics::BallJoint, 153
 - gazebo::physics::Joint, 427
 - gazebo::physics::SimbodyBallJoint, 783
 - gazebo::physics::SimbodyHinge2Joint, 798
 - gazebo::physics::SimbodyHingeJoint, 804
 - gazebo::physics::SimbodyScrewJoint, 848
 - gazebo::physics::SimbodySliderJoint, 854
 - gazebo::physics::SimbodyUniversalJoint, 863
- SetMOI
 - gazebo::physics::Inertial, 407
- SetMass
 - gazebo::physics::Inertial, 407
- SetMaterial
 - gazebo::rendering::Visual, 1050
- SetMaterialIndex
 - gazebo::common::SubMesh, 925
- SetMaxContacts
 - gazebo::physics::Collision, 221
 - gazebo::physics::PhysicsEngine, 631
- SetMaxForce
 - gazebo::physics::Joint, 427
 - gazebo::physics::SimbodyBallJoint, 783
 - gazebo::physics::SimbodyHinge2Joint, 798
 - gazebo::physics::SimbodyHingeJoint, 804
 - gazebo::physics::SimbodyScrewJoint, 848
- gazebo::physics::SimbodySliderJoint, 855
- gazebo::physics::SimbodyUniversalJoint, 863
- SetMaxStepSize
 - gazebo::physics::PhysicsEngine, 631
- SetMesh
 - gazebo::physics::MeshShape, 536
- setMinimalComms
 - Transport, 90
- SetMode
 - Rendering, 78
- SetModel
 - gazebo::physics::Joint, 427
- SetModelTransform
 - gazebo::common::SkeletonNode, 884
- SetName
 - gazebo::common::Mesh, 525
 - gazebo::common::NodeAnimation, 597
 - gazebo::common::SkeletonAnimation, 876
 - gazebo::common::SkeletonNode, 884
 - gazebo::common::SubMesh, 925
 - gazebo::physics::Base, 162
 - gazebo::physics::Entity, 302
 - gazebo::physics::State, 914
 - gazebo::rendering::Camera, 200
 - gazebo::rendering::Light, 453
 - gazebo::rendering::Visual, 1050
- SetNearClip
 - gazebo::rendering::GpuLaser, 351
- SetNode
 - gazebo::transport::Publisher, 674
- SetNormal
 - gazebo::common::SubMesh, 925
 - gazebo::physics::PlaneShape, 645
- SetNormalCount
 - gazebo::common::SubMesh, 925
- SetNormalMap
 - gazebo::rendering::Visual, 1050
- SetNumVertAttached
 - gazebo::common::Skeleton, 872
- SetOperationType
 - gazebo::rendering::DynamicRenderable, 293
- SetPGain
 - gazebo::common::PID, 639
- SetParam
 - gazebo::physics::PhysicsEngine, 631
- SetParams
 - gazebo::Server, 770
- SetParent
 - gazebo::common::SkeletonNode, 884
 - gazebo::physics::Base, 162
 - gazebo::sensors::Sensor, 759, 760
- SetPath
 - gazebo::common::Mesh, 525
- SetPaused

- gazebo::physics::World, 1080
- gazebo::util::LogRecord, 491
- SetPerPixelLighting
 - gazebo::rendering::RTShaderSystem, 728
- SetPitch
 - gazebo::util::OpenALSource, 616
- SetPoint
 - gazebo::rendering::DynamicLines, 289
- SetPointSize
 - gazebo::common::Material, 505
- SetPoints
 - gazebo::physics::RayShape, 704
 - gazebo::physics::SimbodyRayShape, 841
- SetPose
 - gazebo::rendering::Visual, 1051
 - gazebo::util::OpenALSink, 612
 - gazebo::util::OpenALSource, 616
- SetPosition
 - gazebo::rendering::Light, 453
 - gazebo::rendering::Visual, 1051
- SetPrimitiveType
 - gazebo::common::SubMesh, 925
- SetProvideFeedback
 - gazebo::physics::Joint, 427
- SetPublication
 - gazebo::transport::Publisher, 674
- SetPublishData
 - gazebo::physics::Link, 473
- SetQuiet
 - Common, 41
- SetRadius
 - gazebo::physics::CylinderShape, 271
 - gazebo::physics::SimbodySphereShape, 857
 - gazebo::physics::SphereShape, 901
- SetRange
 - gazebo::rendering::Light, 453
- SetRangeCount
 - gazebo::rendering::GpuLaser, 351
- SetRayCountRatio
 - gazebo::rendering::GpuLaser, 352
- SetRealTime
 - gazebo::physics::LinkState, 482
 - gazebo::physics::ModelState, 562
 - gazebo::physics::State, 914
 - gazebo::physics::WorldState, 1088
- SetRealTimeUpdateRate
 - gazebo::physics::PhysicsEngine, 631
- SetReferencePose
 - gazebo::sensors::ImuSensor, 398
- SetRelativePose
 - gazebo::physics::Entity, 302
- SetRenderRate
 - gazebo::rendering::Camera, 200
- SetRenderTarget
 - gazebo::rendering::Camera, 201
 - gazebo::rendering::UserCamera, 986
- SetRetro
 - gazebo::physics::RayShape, 704
- SetRibbonTrail
 - gazebo::rendering::Visual, 1051
- SetRootNode
 - gazebo::common::Skeleton, 872
- SetRotation
 - gazebo::common::PoseKeyFrame, 661
 - gazebo::rendering::Visual, 1051
- SetSID
 - gazebo::common::NodeTransform, 602
- SetSORPGSIters
 - gazebo::physics::PhysicsEngine, 632
- SetSORPGSPreconIters
 - gazebo::physics::PhysicsEngine, 632
- SetSORPGSW
 - gazebo::physics::PhysicsEngine, 632
- SetSaveFramePathname
 - gazebo::rendering::Camera, 201
- SetSaveable
 - gazebo::physics::Base, 163
- SetScale
 - gazebo::common::Mesh, 525
 - gazebo::common::SubMesh, 926
 - gazebo::math::Matrix4, 518
 - gazebo::physics::BoxShape, 172
 - gazebo::physics::Collision, 221
 - gazebo::physics::CylinderShape, 271
 - gazebo::physics::HeightmapShape, 385
 - gazebo::physics::Link, 474
 - gazebo::physics::MapShape, 496
 - gazebo::physics::MeshShape, 536
 - gazebo::physics::Model, 550
 - gazebo::physics::MultiRayShape, 585
 - gazebo::physics::PlaneShape, 646
 - gazebo::physics::RayShape, 705
 - gazebo::physics::Shape, 778
 - gazebo::physics::SphereShape, 901
 - gazebo::rendering::Visual, 1051
- SetScene
 - gazebo::rendering::Camera, 201
 - gazebo::rendering::Visual, 1052
- SetSceneNode
 - gazebo::rendering::Camera, 201
- SetSeed
 - gazebo::math::Rand, 690
 - gazebo::physics::PhysicsEngine, 632
 - gazebo::physics::SimbodyPhysics, 835
- SetSelected
 - gazebo::physics::Base, 163
 - gazebo::physics::Link, 474
 - gazebo::rendering::Light, 453

- setSelectedEntity
 - gazebo::event::Events, 321
- SetSelfCollide
 - gazebo::physics::Link, 474
 - gazebo::physics::SimbodyLink, 822
- SetShadeMode
 - gazebo::common::Material, 505
- SetShaderType
 - gazebo::rendering::Visual, 1052
- SetShadowsEnabled
 - gazebo::rendering::Scene, 743
- SetShape
 - gazebo::physics::Collision, 221
- SetShininess
 - gazebo::common::Material, 505
- SetShowOnTop
 - gazebo::rendering::MovableText, 571
- SetSimTime
 - gazebo::physics::LinkState, 482
 - gazebo::physics::ModelState, 562
 - gazebo::physics::State, 914
 - gazebo::physics::World, 1080
 - gazebo::physics::WorldState, 1088
- SetSize
 - gazebo::physics::BoxShape, 172
 - gazebo::physics::CylinderShape, 271
 - gazebo::physics::PlaneShape, 646
 - gazebo::physics::SimbodyBoxShape, 785
 - gazebo::physics::SimbodyCylinderShape, 790
- SetSkeleton
 - gazebo::common::Mesh, 525
- SetSkeletonPose
 - gazebo::rendering::Visual, 1052
- SetSkyXMode
 - gazebo::rendering::Scene, 744
- SetSourceValues
 - gazebo::common::NodeTransform, 602
- SetSpaceWidth
 - gazebo::rendering::MovableText, 571
- SetSpecular
 - gazebo::common::Material, 505
 - gazebo::rendering::Visual, 1052
- SetSpecularColor
 - gazebo::rendering::Light, 454
- SetSpotFalloff
 - gazebo::rendering::Light, 454
- SetSpotInnerAngle
 - gazebo::rendering::Light, 454
- SetSpotOuterAngle
 - gazebo::rendering::Light, 454
- SetState
 - gazebo::physics::Collision, 222
 - gazebo::physics::Joint, 428
 - gazebo::physics::Link, 474
 - gazebo::physics::Model, 550
 - gazebo::physics::World, 1080
 - Rendering, 78
- SetStatic
 - gazebo::physics::Entity, 303
- SetSubMeshCenter
 - gazebo::common::SubMesh, 926
- SetSurfaceType
 - gazebo::common::SphericalCoordinates, 905
- SetTargetRealTimeFactor
 - gazebo::physics::PhysicsEngine, 632
- SetTension
 - gazebo::math::Spline, 909
- SetTexCoord
 - gazebo::common::SubMesh, 926
- SetTexCoordCount
 - gazebo::common::SubMesh, 926
- SetText
 - gazebo::rendering::MovableText, 571
- SetTextAlignment
 - gazebo::rendering::MovableText, 572
- SetTexture
 - gazebo::rendering::Projector, 664
- SetTextureImage
 - gazebo::common::Material, 505
- SetThreadPitch
 - gazebo::physics::ScrewJoint, 748
 - gazebo::physics::SimbodyScrewJoint, 848
- SetTime
 - gazebo::common::Animation, 142
- SetToldentity
 - gazebo::math::Quaternion, 687
- SetToMax
 - gazebo::math::Vector3, 1016
- SetToMin
 - gazebo::math::Vector3, 1016
- SetToWallTime
 - gazebo::common::Time, 964
- SetTorque
 - gazebo::physics::Link, 474
 - gazebo::physics::SimbodyLink, 822
- SetTransform
 - gazebo::common::SkeletonNode, 884
- SetTranslate
 - gazebo::math::Matrix4, 518
- SetTranslation
 - gazebo::common::PoseKeyFrame, 661
- SetTransparency
 - gazebo::common::Material, 506
 - gazebo::rendering::Visual, 1052
- SetTransparent
 - gazebo::rendering::Scene, 744
- SetType
 - gazebo::common::NodeTransform, 603

- gazebo::common::SkeletonNode, 885
- SetUp
 - Joint_TEST, 432
- SetUpdateRate
 - gazebo::sensors::Sensor, 760
- SetUserData
 - gazebo::rendering::Grid, 368
- SetValue
 - gazebo::common::NumericKeyFrame, 609
- SetVelocity
 - gazebo::physics::Joint, 428
 - gazebo::physics::SimbodyBallJoint, 783
 - gazebo::physics::SimbodyHinge2Joint, 798
 - gazebo::physics::SimbodyHingeJoint, 804
 - gazebo::physics::SimbodyScrewJoint, 849
 - gazebo::physics::SimbodySliderJoint, 855
 - gazebo::physics::SimbodyUniversalJoint, 863
 - gazebo::util::OpenALSink, 612
 - gazebo::util::OpenALSOURCE, 616
- SetVertFOV
 - gazebo::rendering::GpuLaser, 352
- SetVertHalfAngle
 - gazebo::rendering::GpuLaser, 352
- SetVertex
 - gazebo::common::SubMesh, 926
- SetVertexCount
 - gazebo::common::SubMesh, 926
- SetVerticalAngleMax
 - gazebo::sensors::GpuRaySensor, 363
- SetVerticalAngleMin
 - gazebo::sensors::GpuRaySensor, 363
- SetViewController
 - gazebo::rendering::UserCamera, 986
- SetViewportDimensions
 - gazebo::rendering::UserCamera, 986
- SetVisibilityFlags
 - gazebo::rendering::Visual, 1052
- SetVisible
 - gazebo::rendering::Scene, 744
 - gazebo::rendering::Visual, 1053
 - gazebo::rendering::WireBox, 1060
- SetWallTime
 - gazebo::physics::LinkState, 482
 - gazebo::physics::ModelState, 562
 - gazebo::physics::State, 914
 - gazebo::physics::WorldState, 1088
- SetWindowId
 - gazebo::rendering::Camera, 201
- SetWireframe
 - gazebo::rendering::Heightmap, 379
 - gazebo::rendering::Scene, 744
 - gazebo::rendering::Visual, 1053
- SetWorld
 - gazebo::physics::Base, 163
 - gazebo::physics::WorldState, 1089
- SetWorldCFM
 - gazebo::physics::PhysicsEngine, 633
- SetWorldERP
 - gazebo::physics::PhysicsEngine, 633
- SetWorldPose
 - gazebo::physics::Entity, 303
 - gazebo::rendering::Camera, 201
 - gazebo::rendering::UserCamera, 987
 - gazebo::rendering::Visual, 1053
- SetWorldPosition
 - gazebo::rendering::Camera, 202
 - gazebo::rendering::Visual, 1053
- SetWorldRotation
 - gazebo::rendering::Camera, 202
 - gazebo::rendering::Visual, 1053
- SetWorldTwist
 - gazebo::physics::Entity, 303
- ShadeMode
 - gazebo::common::Material, 500
- shadeMode
 - gazebo::common::Material, 507
- ShadeModeStr
 - gazebo::common::Material, 507
- Shape
 - gazebo::physics::Shape, 777
- shape
 - gazebo::physics::Collision, 222
- Shape.hh, 1253
- ShapePtr
 - gazebo::physics, 110
- shift
 - gazebo::common::MouseEvent, 566
- shininess
 - gazebo::common::Material, 507
- Show
 - gazebo::rendering::GUIOverlay, 374
- ShowBoundingBox
 - gazebo::rendering::Visual, 1054
- ShowCOM
 - gazebo::rendering::Visual, 1054
- ShowCOMs
 - gazebo::rendering::Scene, 745
- ShowClouds
 - gazebo::rendering::Scene, 744
- ShowCollision
 - gazebo::rendering::Visual, 1054
- ShowCollisions
 - gazebo::rendering::Scene, 745
- ShowContacts
 - gazebo::rendering::Scene, 745
- ShowJoints
 - gazebo::rendering::Scene, 745
 - gazebo::rendering::Visual, 1054

- ShowRotation
 - gazebo::rendering::ArrowVisual, 145
 - gazebo::rendering::AxisVisual, 151
- ShowSkeleton
 - gazebo::rendering::Visual, 1054
- ShowVisual
 - gazebo::rendering::Light, 454
- ShowWireframe
 - gazebo::rendering::Camera, 202
- Shutdown
 - gazebo::transport::Connection, 247
- sid
 - gazebo::common::NodeTransform, 603
- sigInt
 - gazebo::event::Events, 321
- Signal
 - gazebo::event::EventT, 328–330
- SimTK, 123
- simTime
 - gazebo::common::UpdateInfo, 978
 - gazebo::physics::State, 915
- Simbody Physics, 71
- simbody_inc.h, 1254
- SimbodyBallJoint
 - gazebo::physics::SimbodyBallJoint, 780
- SimbodyBallJoint.hh, 1254
- SimbodyBoxShape
 - gazebo::physics::SimbodyBoxShape, 785
- SimbodyBoxShape.hh, 1255
- SimbodyCollision
 - gazebo::physics::SimbodyCollision, 787
- SimbodyCollision.hh, 1255
- SimbodyCollisionPtr
 - gazebo::physics, 110
- SimbodyCylinderShape
 - gazebo::physics::SimbodyCylinderShape, 790
- SimbodyCylinderShape.hh, 1256
- SimbodyHeightmapShape
 - gazebo::physics::SimbodyHeightmapShape, 792
- SimbodyHeightmapShape.hh, 1257
- SimbodyHinge2Joint
 - gazebo::physics::SimbodyHinge2Joint, 794
- SimbodyHinge2Joint.hh, 1257
- SimbodyHingeJoint
 - gazebo::physics::SimbodyHingeJoint, 800
- SimbodyHingeJoint.hh, 1258
- SimbodyJoint
 - gazebo::physics::SimbodyJoint, 807
- SimbodyJoint.hh, 1259
- SimbodyLink
 - gazebo::physics::SimbodyLink, 816
- SimbodyLink.hh, 1259
- SimbodyLinkPtr
 - gazebo::physics, 110
- SimbodyMeshShape
 - gazebo::physics::SimbodyMeshShape, 824
- SimbodyMeshShape.hh, 1260
- SimbodyModel
 - gazebo::physics::SimbodyModel, 826
- SimbodyModel.hh, 1261
- SimbodyModelPtr
 - gazebo::physics, 110
- SimbodyMultiRayShape
 - gazebo::physics::SimbodyMultiRayShape, 828
- SimbodyMultiRayShape.hh, 1261
- SimbodyPhysics
 - gazebo::physics::SimbodyPhysics, 831
- simbodyPhysics
 - gazebo::physics::SimbodyJoint, 813
- SimbodyPhysics.hh, 1262
- simbodyPhysicsInitialized
 - gazebo::physics::SimbodyPhysics, 837
- SimbodyPhysicsPtr
 - gazebo::physics, 110
- simbodyPhysicsStepped
 - gazebo::physics::SimbodyPhysics, 837
- SimbodyPlaneShape
 - gazebo::physics::SimbodyPlaneShape, 839
- SimbodyPlaneShape.hh, 1263
- SimbodyRayShape
 - gazebo::physics::SimbodyRayShape, 841
- SimbodyRayShape.hh, 1263
- SimbodyRayShapePtr
 - gazebo::physics, 110
- SimbodyScrewJoint
 - gazebo::physics::SimbodyScrewJoint, 844
- SimbodyScrewJoint.hh, 1264
- SimbodySliderJoint
 - gazebo::physics::SimbodySliderJoint, 851
- SimbodySliderJoint.hh, 1265
- SimbodySphereShape
 - gazebo::physics::SimbodySphereShape, 856
- SimbodySphereShape.hh, 1265
- SimbodyTypes.hh, 1266
- SimbodyUniversalJoint
 - gazebo::physics::SimbodyUniversalJoint, 859
- SimbodyUniversalJoint.hh, 1267
- SingletonT
 - ~SingletonT, 866
 - Instance, 866
 - SingletonT, 866
 - SingletonT, 866
 - SingletonT < T >, 864
 - SingletonT.hh, 1268
- size
 - gazebo::math::Plane, 642
- skelAnimation
 - gazebo::physics::Actor, 130

- skelNodesMap
 - gazebo::physics::Actor, 130
- Skeleton
 - gazebo::common::Skeleton, 868
- skeleton
 - gazebo::physics::Actor, 130
- Skeleton.hh, 1268
- SkeletonAnimation
 - gazebo::common::SkeletonAnimation, 874
- SkeletonAnimation.hh, 1270
- SkeletonNode
 - gazebo::common::SkeletonNode, 879, 880
- SkeletonNodeType
 - gazebo::common::SkeletonNode, 879
- skinFile
 - gazebo::physics::Actor, 131
- skinScale
 - gazebo::physics::Actor, 131
- SkyX, 123
- SkyXMode
 - gazebo::rendering::Scene, 732
- skyx
 - gazebo::rendering::Scene, 746
- slaveMobods
 - gazebo::physics::SimbodyLink, 822
- slaveWelds
 - gazebo::physics::SimbodyLink, 822
- Sleep
 - gazebo::common::Time, 964
- Slerp
 - gazebo::math::Quaternion, 687
- SliderJoint
 - gazebo::physics::SliderJoint, 887
- SliderJoint.hh, 1271
- slip1
 - gazebo::physics::SurfaceParams, 937
- slip2
 - gazebo::physics::SurfaceParams, 937
- Smooth
 - gazebo::rendering::Heightmap, 379
- SnapVisualToNearestBelow
 - gazebo::rendering::Scene, 745
- SonarSensor
 - gazebo::sensors::SonarSensor, 892
- SonarSensor.hh, 1272
- SonarSensorPtr
 - gazebo::sensors, 118
- SonarVisual
 - gazebo::rendering::SonarVisual, 896
- SonarVisual.hh, 1273
- SonarVisualPtr
 - gazebo::rendering, 114
- source
 - gazebo::common::NodeTransform, 603
- SpawnJoint
 - Joint_TEST, 432, 433
- SpawnJointOptions
 - Joint_TEST::SpawnJointOptions, 897
- SpawnJointRotational
 - Joint_TEST, 433
- SpawnJointRotationalWorld
 - Joint_TEST, 433
- SpawnJointTypes
 - Joint_TEST, 433
- specular
 - gazebo::common::Material, 507
- SpeedOfLight
 - gazebo::common, 99
- SphereShape
 - gazebo::physics::SphereShape, 900
- SphereShape.hh, 1274
- SphereShapePtr
 - gazebo::physics, 110
- SphericalCoordinates
 - gazebo::common::SphericalCoordinates, 903
- SphericalCoordinates.hh, 1275
- SphericalCoordinatesPtr
 - gazebo::common, 98
- SphericalFromLocal
 - gazebo::common::SphericalCoordinates, 905
- Spline
 - gazebo::math::Spline, 907
- Spline.hh, 1276
- SplitHeights
 - gazebo::rendering::Heightmap, 380
- Squad
 - gazebo::math::Quaternion, 687
- Stamp
 - Messages, 61
- Start
 - Common, 41
 - gazebo::common::Timer, 967
 - gazebo::util::DiagnosticTimer, 284
 - gazebo::util::LogRecord, 492
- startDelay
 - gazebo::physics::Actor, 131
- StartRead
 - gazebo::transport::Connection, 247
- startTime
 - gazebo::physics::TrajectoryInfo, 974
- StartTimer
 - gazebo::util::DiagnosticManager, 282
- State
 - gazebo::physics::State, 912
- State.hh, 1278
- std_string2
 - Joint_TEST.hh, 1165
- Step

- gazebo::util::LogPlay, 486
- step
 - gazebo::event::Events, 322
- StepWorld
 - gazebo::physics::World, 1081
- Stop
 - gazebo::common::Timer, 967
 - gazebo::Master, 498
 - gazebo::physics::Actor, 128
 - gazebo::physics::World, 1081
 - gazebo::sensors::SensorManager, 767
 - gazebo::Server, 770
 - gazebo::transport::ConnectionManager, 251
 - gazebo::transport::IOManager, 410
 - gazebo::util::DiagnosticTimer, 284
 - gazebo::util::LogRecord, 492
 - gazebo::util::OpenALSource, 617
- stop
 - gazebo, 95
 - gazebo::event::Events, 322
 - Sensors, 84
 - Transport, 90
- stop_world
 - Classes for physics and dynamics, 69
- stop_worlds
 - Classes for physics and dynamics, 69
- StopAnimation
 - gazebo::physics::Entity, 303
 - gazebo::physics::Model, 550
- StopRead
 - gazebo::transport::Connection, 247
- StopTimer
 - gazebo::util::DiagnosticManager, 282
- StrStr_M
 - gazebo::common, 98
- StripSceneName
 - gazebo::rendering::Scene, 745
- StripWorldName
 - gazebo::physics::World, 1081
- SubMesh
 - gazebo::common::SubMesh, 919
- SubNodeMap
 - gazebo::transport::TopicManager, 969
- subSampling
 - gazebo::physics::HeightmapShape, 385
- submesh
 - gazebo::physics::MeshShape, 537
- Subscribe
 - gazebo::transport::ConnectionManager, 251
 - gazebo::transport::Node, 592, 593
 - gazebo::transport::TopicManager, 973
- SubscribeOptions
 - gazebo::transport::SubscribeOptions, 928
- SubscribeOptions.hh, 1280
- Subscriber
 - gazebo::transport::Subscriber, 929
- Subscriber.hh, 1282
- SubscriberPtr
 - gazebo::transport, 121
- SubscriptionTransport
 - gazebo::transport::SubscriptionTransport, 931
- SubscriptionTransport.hh, 1284
- SubscriptionTransportPtr
 - gazebo::transport, 121
- SurfaceParams
 - gazebo::physics::SurfaceParams, 934
- SurfaceParams.hh, 1286
- SurfaceParamsPtr
 - gazebo::physics, 110
- SurfaceType
 - gazebo::common::SphericalCoordinates, 902
- system
 - gazebo::physics::SimbodyPhysics, 837
- SystemPaths.hh, 1287
 - GetCurrentDir, 1288
 - LINUX, 1288
- SystemPlugin
 - gazebo::SystemPlugin, 943
- SystemPluginPtr
 - gazebo, 95
- systemPluginsArgc
 - gazebo::Server, 770
- systemPluginsArgv
 - gazebo::Server, 771
- TPtr
 - gazebo::PluginT, 647
- TRANS
 - Rendering, 76
- TRANS_X
 - Rendering, 76
- TRANS_Y
 - Rendering, 76
- TRANS_Z
 - Rendering, 76
- TRANSLATE
 - gazebo::common::NodeTransform, 600
- TRIANGLES
 - gazebo::common::SubMesh, 919
- TRIFANS
 - gazebo::common::SubMesh, 919
- TRISTRIPS
 - gazebo::common::SubMesh, 919
- tangents
 - gazebo::math::RotationSpline, 724
 - gazebo::math::Spline, 910
- targetRealTimeFactor
 - gazebo::physics::PhysicsEngine, 634

- tension
 - gazebo::math::Spline, 910
- texImage
 - gazebo::common::Material, 507
- textureHeight
 - gazebo::rendering::Camera, 205
- textureWidth
 - gazebo::rendering::Camera, 205
- threadPitch
 - gazebo::physics::ScrewJoint, 749
- Time
 - gazebo::common::Time, 948, 949
- time
 - gazebo::common::KeyFrame, 447
 - gazebo::physics::Contact, 256
- Time.hh, 1288
- timePos
 - gazebo::common::Animation, 143
- Timer
 - gazebo::common::Timer, 966
- Timer.hh, 1289
- Toggle
 - gazebo::rendering::Projector, 664
- ToggleShowVisual
 - gazebo::rendering::Light, 454
- ToggleShowWireframe
 - gazebo::rendering::Camera, 202
- ToggleVisible
 - gazebo::rendering::Visual, 1054
- TopicManager.hh, 1290
- TrackVisual
 - gazebo::rendering::Camera, 202
- TrackVisualFromSDF
 - Messages, 61
- TrackVisualImpl
 - gazebo::rendering::Camera, 202, 203
 - gazebo::rendering::UserCamera, 987
- tracker
 - gazebo::physics::SimbodyPhysics, 837
- trajInfo
 - gazebo::physics::Actor, 131
- trajectories
 - gazebo::physics::Actor, 131
- transform
 - gazebo::common::NodeTransform, 603
 - gazebo::common::SkeletonNode, 886
- Transform2Pose
 - gazebo::physics::SimbodyPhysics, 836
- TransformAffine
 - gazebo::math::Matrix4, 518
- TransformType
 - gazebo::common::NodeTransform, 600
- Translate
 - gazebo::common::Mesh, 526
 - gazebo::common::SubMesh, 927
- gazebo::rendering::Camera, 203
- translate
 - gazebo::common::PoseKeyFrame, 662
- translated
 - gazebo::physics::TrajectoryInfo, 974
- TransmitterVisual
 - gazebo::rendering::TransmitterVisual, 976
- TransmitterVisual.hh, 1292
- transparency
 - gazebo::common::Material, 507
- Transport, 85
 - CallbackHelperPtr, 87
 - clear_buffers, 87
 - fini, 87
 - get_master_uri, 87
 - get_topic_namespaces, 87
 - getAdvertisedTopics, 88
 - getMinimalComms, 88
 - getTopicMsgType, 88
 - init, 88
 - is_stopped, 89
 - pause_incoming, 89
 - publish, 89
 - request, 89
 - requestNoReply, 89, 90
 - run, 90
 - setMinimalComms, 90
 - stop, 90
- TransportIface.hh, 1293
- TransportTypes.hh, 1295
- TriggerUpdate
 - gazebo::transport::ConnectionManager, 251
- TwoPi
 - gazebo::math::Angle, 139
- type
 - gazebo::common::KeyEvent, 445
 - gazebo::common::MouseEvent, 566
 - gazebo::common::NodeTransform, 603
 - gazebo::common::SkeletonNode, 886
 - gazebo::physics::TrajectoryInfo, 974
 - gazebo::PluginT, 648
 - Joint_TEST::SpawnJointOptions, 898
- typeString
 - gazebo::rendering::ViewController, 1034
- UIntGen
 - gazebo::math, 102
- UNION
 - gazebo::common::MeshCSG, 527
- UNIVERSAL_JOINT
 - gazebo::physics::Base, 156
- UNKNOWN_PIXEL_FORMAT
 - gazebo::common::Image, 391

- URealGen
 - gazebo::math, 102
- Unadvertise
 - gazebo::transport::ConnectionManager, 251
 - gazebo::transport::TopicManager, 973
- UniformIntDist
 - gazebo::math, 102
- UniformRealDist
 - gazebo::math, 102
- UnitX
 - gazebo::math::Vector3, 1017
- UnitY
 - gazebo::math::Vector3, 1017
- UnitZ
 - gazebo::math::Vector3, 1018
- UniversalJoint
 - gazebo::physics::UniversalJoint, 977
- UniversalJoint.hh, 1296
- unloadProceduralPage
 - gazebo::rendering::DummyPageProvider, 285
- unprepareProceduralPage
 - gazebo::rendering::DummyPageProvider, 286
- Unsubscribe
 - gazebo::transport::ConnectionManager, 251
 - gazebo::transport::Subscriber, 930
 - gazebo::transport::TopicManager, 973
- Update
 - gazebo::common::PID, 639
 - gazebo::physics::Actor, 129
 - gazebo::physics::Base, 163
 - gazebo::physics::Joint, 428
 - gazebo::physics::JointController, 436
 - gazebo::physics::Link, 474
 - gazebo::physics::MapShape, 496
 - gazebo::physics::MeshShape, 537
 - gazebo::physics::Model, 551
 - gazebo::physics::MultiRayShape, 585
 - gazebo::physics::RayShape, 705
 - gazebo::physics::SimbodyRayShape, 842
 - gazebo::rendering::Camera, 203
 - gazebo::rendering::DynamicLines, 289
 - gazebo::rendering::FPSViewController, 340
 - gazebo::rendering::GUIOverlay, 374
 - gazebo::rendering::MovableText, 572
 - gazebo::rendering::OrbitViewController, 620
 - gazebo::rendering::TransmitterVisual, 976
 - gazebo::rendering::UserCamera, 987
 - gazebo::rendering::ViewController, 1034
 - gazebo::rendering::Visual, 1054
 - gazebo::sensors::Sensor, 760
 - gazebo::sensors::SensorManager, 767
- update
 - gazebo::sensors::ForceTorqueSensor, 337
 - gazebo::sensors::SonarSensor, 895
- UpdateChildrenTransforms
 - gazebo::common::SkeletonNode, 885
- UpdateCollision
 - gazebo::physics::PhysicsEngine, 633
 - gazebo::physics::SimbodyPhysics, 836
- UpdateFromMsg
 - gazebo::rendering::Light, 454
 - gazebo::rendering::Visual, 1054
- UpdateImpl
 - gazebo::sensors::CameraSensor, 209
 - gazebo::sensors::ContactSensor, 264
 - gazebo::sensors::DepthCameraSensor, 279
 - gazebo::sensors::ForceTorqueSensor, 337
 - gazebo::sensors::GpsSensor, 343
 - gazebo::sensors::GpuRaySensor, 364
 - gazebo::sensors::ImuSensor, 398
 - gazebo::sensors::MultiCameraSensor, 578
 - gazebo::sensors::RaySensor, 699
 - gazebo::sensors::RFIDSensor, 712
 - gazebo::sensors::RFIDTag, 714
 - gazebo::sensors::Sensor, 760
 - gazebo::sensors::SonarSensor, 894
 - gazebo::sensors::WirelessTransmitter, 1069
- UpdateInfo.hh, 1297
- UpdateMass
 - gazebo::physics::Link, 475
- UpdateParameters
 - gazebo::physics::Actor, 129
 - gazebo::physics::Base, 163
 - gazebo::physics::Collision, 222
 - gazebo::physics::Entity, 303
 - gazebo::physics::Inertial, 407
 - gazebo::physics::Joint, 428
 - gazebo::physics::Link, 475
 - gazebo::physics::Model, 551
- UpdateParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 890
- updateParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 775
- UpdateParamsForCompositeMap
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 890
- updatePeriod
 - gazebo::sensors::Sensor, 762
- UpdatePhysics
 - gazebo::physics::PhysicsEngine, 633
 - gazebo::physics::SimbodyPhysics, 836
- UpdatePoint
 - gazebo::math::RotationSpline, 723
 - gazebo::math::Spline, 909
- UpdatePublications
 - gazebo::transport::TopicManager, 974

- UpdateRays
 - gazebo::physics::MultiRayShape, 585
 - gazebo::physics::SimbodyMultiRayShape, 828
- UpdateShaders
 - gazebo::rendering::RTShaderSystem, 728
- UpdateSize
 - Rendering, 79
- UpdateStateSDF
 - gazebo::physics::World, 1081
- UpdateSurface
 - gazebo::physics::Link, 475
- updateVpParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 775
- upperLimit
 - gazebo::physics::Joint, 429
- useCFMDamping
 - gazebo::physics::Joint, 430
- UserCamera
 - gazebo::rendering::UserCamera, 981
- UserCamera.hh, 1298
- UserCameraPtr
 - gazebo::rendering, 114
- UtilTypes.hh, 1299
- Utility, 91
 - DIAG_TIMER_LAP, 91
 - DIAG_TIMER_START, 91
 - DIAG_TIMER_STOP, 91
- V_ABOVE
 - gazebo::rendering::MovableText, 568
- V_BELOW
 - gazebo::rendering::MovableText, 568
- VEL
 - gazebo::physics::Joint, 415
- VERTEX
 - gazebo::rendering::RenderEngine, 707
- VISUAL
 - gazebo::physics::Base, 156
- VISUAL_PLUGIN
 - Common, 36
- Valid
 - gazebo::common::Image, 394
- ValidateIP
 - gazebo::transport::Connection, 247
- value
 - gazebo::common::NumericKeyFrame, 609
- variance
 - Math, 50
- Vec3ToVector3
 - gazebo::physics::SimbodyPhysics, 836
- Vector2d
 - gazebo::math::Vector2d, 989
- Vector2d.hh, 1300
- Vector2i
 - gazebo::math::Vector2i, 997, 998
- Vector2i.hh, 1301
- Vector3
 - gazebo::math::Vector3, 1007, 1008
- Vector3.hh, 1302
- Vector3ToVec3
 - gazebo::physics::SimbodyPhysics, 836
- Vector4
 - gazebo::math::Vector4, 1020
- Vector4.hh, 1303
- velocityLimit
 - gazebo::physics::Joint, 430
- VertAlign
 - gazebo::rendering::MovableText, 568
- vertElem
 - gazebo::physics::MultiRayShape, 586
 - gazebo::sensors::GpuRaySensor, 364
- vertHalfAngle
 - gazebo::rendering::GpuLaser, 353
- vertRangeCount
 - gazebo::sensors::GpuRaySensor, 364
- vertRayCount
 - gazebo::sensors::GpuRaySensor, 365
- vertSize
 - gazebo::physics::HeightmapShape, 385
- vertexBufferCapacity
 - gazebo::rendering::DynamicRenderable, 293
- vertexIndex
 - gazebo::common::NodeAssignment, 598
- vfov
 - gazebo::rendering::GpuLaser, 353
- Video
 - gazebo::common::Video, 1028
- Video.hh, 1305
- VideoVisual
 - gazebo::rendering::VideoVisual, 1030
- VideoVisual.hh, 1306
- ViewController
 - gazebo::rendering::ViewController, 1032
- ViewController.hh, 1307
- viewport
 - gazebo::rendering::Camera, 206
- visPub
 - gazebo::physics::Entity, 304
- visitRenderables
 - gazebo::rendering::MovableText, 572
- Visual
 - gazebo::rendering::Visual, 1040
- Visual.hh, 1308
- VisualFromSDF
 - Messages, 62
- visualId
 - gazebo::physics::Actor, 131

- visualMsg
 - gazebo::physics::Entity, 305
- visualName
 - gazebo::physics::Actor, 131
- VisualPlugin
 - gazebo::VisualPlugin, 1056
- VisualPluginPtr
 - gazebo, 95
- VisualPtr
 - gazebo::rendering, 114
- visuals
 - gazebo::physics::Link, 476
- Visuals_M
 - gazebo::physics::Link, 460
- w
 - gazebo::math::Quaternion, 688
 - gazebo::math::Vector4, 1027
- WORLD_PLUGIN
 - Common, 36
- wait
 - Joint_TEST::SpawnJointOptions, 898
- WaitForConnection
 - gazebo::transport::Publisher, 674
- wallTime
 - gazebo::physics::State, 915
- weight
 - gazebo::common::NodeAssignment, 598
- White
 - gazebo::common::Color, 237
- windowId
 - gazebo::rendering::Camera, 206
- WindowManager
 - gazebo::rendering::WindowManager, 1057
- WindowManager.hh, 1309
- WindowManagerPtr
 - gazebo::rendering, 114
- WireBox
 - gazebo::rendering::WireBox, 1060
- WireBox.hh, 1310
- WirelessReceiver
 - gazebo::sensors::WirelessReceiver, 1062
- WirelessReceiver.hh, 1311
- WirelessReceiver_V
 - gazebo::sensors, 118
- WirelessReceiverPtr
 - gazebo::sensors, 119
- WirelessTransceiver
 - gazebo::sensors::WirelessTransceiver, 1064
- WirelessTransceiver.hh, 1311
- WirelessTransceiver_V
 - gazebo::sensors, 119
- WirelessTransceiverPtr
 - gazebo::sensors, 119
- WirelessTransmitter
 - gazebo::sensors::WirelessTransmitter, 1068
- WirelessTransmitter.hh, 1312
- WirelessTransmitter_V
 - gazebo::sensors, 119
- WirelessTransmitterPtr
 - gazebo::sensors, 119
- World
 - gazebo::physics::World, 1073
- world
 - gazebo::physics::Base, 164
 - gazebo::physics::Contact, 256
 - gazebo::physics::PhysicsEngine, 634
 - gazebo::physics::SimbodyJoint, 813
 - gazebo::sensors::Sensor, 762
- World.hh, 1313
- worldChild
 - Joint_TEST::SpawnJointOptions, 898
- worldCreated
 - gazebo::event::Events, 322
- worldName
 - gazebo::common::UpdateInfo, 979
- worldParent
 - Joint_TEST::SpawnJointOptions, 898
- WorldPlugin
 - gazebo::WorldPlugin, 1082
- WorldPluginPtr
 - gazebo, 95
- WorldPtr
 - gazebo::physics, 110
- WorldState
 - gazebo::physics::WorldState, 1085
- WorldState.hh, 1314
- worldUpdateBegin
 - gazebo::event::Events, 322
- worldUpdateEnd
 - gazebo::event::Events, 322
- worlds_running
 - Classes for physics and dynamics, 69
- wrench
 - gazebo::physics::Contact, 256
 - gazebo::physics::Joint, 430
- WrenchVisual
 - gazebo::rendering::WrenchVisual, 1090
- WrenchVisual.hh, 1316
- WrenchVisualPtr
 - gazebo::rendering, 115
- Write
 - gazebo::util::LogRecord, 492
- x
 - gazebo::math::Quaternion, 688
 - gazebo::math::Vector2d, 995
 - gazebo::math::Vector2i, 1004

- gazebo::math::Vector3, 1018
- gazebo::math::Vector4, 1027
- X_POSITION
 - BVHLoader.hh, 1110
- X_ROTATION
 - BVHLoader.hh, 1110
- xCB
 - gazebo::physics::SimbodyJoint, 813
- xPA
 - gazebo::physics::SimbodyJoint, 813
- y
 - gazebo::math::Quaternion, 688
 - gazebo::math::Vector2d, 995
 - gazebo::math::Vector2i, 1004
 - gazebo::math::Vector3, 1018
 - gazebo::math::Vector4, 1027
- Y_POSITION
 - BVHLoader.hh, 1110
- Y_ROTATION
 - BVHLoader.hh, 1110
- Yellow
 - gazebo::common::Color, 237
- z
 - gazebo::math::Quaternion, 688
 - gazebo::math::Vector3, 1018
 - gazebo::math::Vector4, 1027
- Z_POSITION
 - BVHLoader.hh, 1110
- Z_ROTATION
 - BVHLoader.hh, 1110
- ZERO
 - gazebo::math::Matrix4, 519
- Zero
 - gazebo::common::Time, 965
 - gazebo::math::Angle, 139
 - gazebo::math::Pose, 657
 - gazebo::math::Vector3, 1018