

Gazebo
1.5.0

Generated by Doxygen 1.8.2

Fri Mar 8 2013 10:59:25

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	23
7.1	File List	23
8	Module Documentation	27
8.1	Common	27
8.1.1	Detailed Description	31
8.1.2	Macro Definition Documentation	31
8.1.2.1	gzclr_end	31
8.1.2.2	gzclr_start	31
8.1.2.3	gzdbg	31
8.1.2.4	gzerr	31
8.1.2.5	gzlog	31
8.1.2.6	gzmsg	31
8.1.2.7	gzthrow	31
8.1.2.8	gzwarn	32

8.1.3	Enumeration Type Documentation	32
8.1.3.1	PluginType	32
8.1.4	Function Documentation	32
8.1.4.1	NullStream	32
8.1.4.2	add_search_path_suffix	32
8.1.4.3	ColorErr	32
8.1.4.4	ColorMsg	33
8.1.4.5	DownloadDependencies	33
8.1.4.6	find_file	33
8.1.4.7	find_file_path	33
8.1.4.8	GetDBConfig	34
8.1.4.9	GetManifest	34
8.1.4.10	GetModelConfig	34
8.1.4.11	GetModelFile	34
8.1.4.12	GetModelName	34
8.1.4.13	GetModelPath	35
8.1.4.14	GetModels	35
8.1.4.15	GetModels	35
8.1.4.16	GetURI	35
8.1.4.17	HasModel	35
8.1.4.18	Init	36
8.1.4.19	Log	36
8.1.4.20	SetQuiet	36
8.1.5	Variable Documentation	36
8.1.5.1	PixelFormatNames	36
8.2	Events	38
8.2.1	Detailed Description	38
8.2.2	Function Documentation	38
8.2.2.1	~EventT	38
8.2.2.2	Connect	38
8.2.2.3	ConnectionCount	39
8.2.2.4	Disconnect	39
8.2.2.5	Disconnect	40
8.3	Classes for physics and dynamics	41
8.3.1	Detailed Description	44
8.3.2	Macro Definition Documentation	44
8.3.2.1	GZ_REGISTER_PHYSICS_ENGINE	44

8.3.3	Typedef Documentation	44
8.3.3.1	PhysicsFactoryFn	44
8.3.4	Function Documentation	44
8.3.4.1	create_world	44
8.3.4.2	fini	44
8.3.4.3	get_world	44
8.3.4.4	init_world	45
8.3.4.5	init_worlds	45
8.3.4.6	load	45
8.3.4.7	load_world	45
8.3.4.8	load_worlds	45
8.3.4.9	pause_world	45
8.3.4.10	pause_worlds	46
8.3.4.11	remove_worlds	46
8.3.4.12	run_world	46
8.3.4.13	run_worlds	46
8.3.4.14	stop_world	46
8.3.4.15	stop_worlds	46
8.3.5	Variable Documentation	46
8.3.5.1	EntityTypename	46
8.4	Math	48
8.4.1	Detailed Description	49
8.4.2	Function Documentation	50
8.4.2.1	clamp	50
8.4.2.2	equal	50
8.4.2.3	isnan	50
8.4.2.4	isnan	50
8.4.2.5	isPowerOfTwo	51
8.4.2.6	max	51
8.4.2.7	mean	51
8.4.2.8	min	51
8.4.2.9	parseFloat	52
8.4.2.10	parseInt	52
8.4.2.11	precision	52
8.4.2.12	variance	53
8.4.3	Variable Documentation	53
8.4.3.1	NAN_D	53

8.4.3.2	NAN_I	53
8.5	Messages	54
8.5.1	Detailed Description	55
8.5.2	Macro Definition Documentation	56
8.5.2.1	GZ_REGISTER_STATIC_MSG	56
8.5.3	Function Documentation	56
8.5.3.1	Convert	56
8.5.3.2	Convert	56
8.5.3.3	Convert	57
8.5.3.4	Convert	57
8.5.3.5	Convert	57
8.5.3.6	Convert	57
8.5.3.7	Convert	58
8.5.3.8	Convert	58
8.5.3.9	Convert	58
8.5.3.10	Convert	58
8.5.3.11	Convert	59
8.5.3.12	Convert	59
8.5.3.13	CreateRequest	59
8.5.3.14	FogFromSDF	59
8.5.3.15	GetHeader	60
8.5.3.16	GUIFromSDF	60
8.5.3.17	Init	60
8.5.3.18	LightFromSDF	60
8.5.3.19	SceneFromSDF	60
8.5.3.20	Set	61
8.5.3.21	Set	61
8.5.3.22	Set	61
8.5.3.23	Set	61
8.5.3.24	Set	61
8.5.3.25	Set	62
8.5.3.26	Set	62
8.5.3.27	Set	62
8.5.3.28	Set	62
8.5.3.29	Stamp	62
8.5.3.30	Stamp	63
8.5.3.31	TrackVisualFromSDF	63

8.5.3.32	VisualFromSDF	63
8.6	Rendering	64
8.6.1	Detailed Description	66
8.6.2	Function Documentation	66
8.6.2.1	create_scene	66
8.6.2.2	fini	66
8.6.2.3	get_scene	66
8.6.2.4	init	66
8.6.2.5	load	66
8.6.2.6	remove_scene	66
8.7	Gazebo_parser	68
8.7.1	Detailed Description	68
8.7.2	Typedef Documentation	68
8.7.2.1	ConstUrdfLinkPtr	68
8.7.2.2	UrdfCollisionPtr	68
8.7.2.3	UrdfLinkPtr	68
8.7.2.4	UrdfVisualPtr	68
8.8	Sensors	69
8.8.1	Detailed Description	70
8.8.2	Macro Definition Documentation	70
8.8.2.1	GZ_REGISTER_STATIC_SENSOR	70
8.8.3	Function Documentation	71
8.8.3.1	create_sensor	71
8.8.3.2	fini	71
8.8.3.3	get_sensor	71
8.8.3.4	init	71
8.8.3.5	load	72
8.8.3.6	remove_sensor	72
8.8.3.7	remove_sensors	72
8.8.3.8	run	72
8.8.3.9	run_once	72
8.8.3.10	run_threads	72
8.8.3.11	stop	73
8.9	Transport	74
8.9.1	Detailed Description	75
8.9.2	Typedef Documentation	75
8.9.2.1	CallbackHelperPtr	75

8.9.3	Function Documentation	76
8.9.3.1	clear_buffers	76
8.9.3.2	fini	76
8.9.3.3	get_master_uri	76
8.9.3.4	get_topic_namespaces	76
8.9.3.5	getAdvertisedTopics	76
8.9.3.6	getAdvertisedTopics	76
8.9.3.7	getTopicMsgType	77
8.9.3.8	init	77
8.9.3.9	is_stopped	77
8.9.3.10	pause_incoming	77
8.9.3.11	request	78
8.9.3.12	requestNoReply	78
8.9.3.13	requestNoReply	78
8.9.3.14	run	78
8.9.3.15	stop	78
8.10	Utility	79
8.10.1	Detailed Description	79
8.10.2	Macro Definition Documentation	79
8.10.2.1	DIAG_TIMER_LAP	79
8.10.2.2	DIAG_TIMER_START	79
8.10.2.3	DIAG_TIMER_STOP	79
9	Namespace Documentation	81
9.1	boost Namespace Reference	81
9.2	gazebo Namespace Reference	81
9.2.1	Detailed Description	82
9.2.2	Typedef Documentation	83
9.2.2.1	GUIPluginPtr	83
9.2.2.2	ModelPluginPtr	83
9.2.2.3	SensorPluginPtr	83
9.2.2.4	SystemPluginPtr	83
9.2.2.5	VisualPluginPtr	83
9.2.2.6	WorldPluginPtr	83
9.2.3	Function Documentation	83
9.2.3.1	add_plugin	83
9.2.3.2	find_file	83

9.2.3.3	fini	83
9.2.3.4	init	83
9.2.3.5	load	83
9.2.3.6	print_version	83
9.2.3.7	run	83
9.2.3.8	stop	83
9.3	gazebo::common Namespace Reference	83
9.3.1	Detailed Description	86
9.3.2	Typedef Documentation	86
9.3.2.1	AnimationPtr	86
9.3.2.2	DiagnosticTimerPtr	86
9.3.2.3	NodeMap	86
9.3.2.4	NodeMapIter	86
9.3.2.5	NumericAnimationPtr	86
9.3.2.6	Param_V	86
9.3.2.7	PoseAnimationPtr	86
9.3.2.8	RawNodeAnim	86
9.3.2.9	RawNodeWeights	86
9.3.2.10	RawSkeletonAnim	86
9.3.2.11	StrStr_M	86
9.4	gazebo::event Namespace Reference	86
9.4.1	Detailed Description	87
9.4.2	Typedef Documentation	87
9.4.2.1	Connection_V	87
9.4.2.2	ConnectionPtr	87
9.5	gazebo::math Namespace Reference	87
9.5.1	Detailed Description	89
9.5.2	Typedef Documentation	89
9.5.2.1	GeneratorType	89
9.5.2.2	NormalRealDist	89
9.5.2.3	NRealGen	89
9.5.2.4	UIntGen	89
9.5.2.5	UniformIntDist	89
9.5.2.6	UniformRealDist	89
9.5.2.7	URealGen	89
9.6	gazebo::msgs Namespace Reference	89
9.6.1	Detailed Description	91

9.6.2	Typedef Documentation	91
9.6.2.1	MsgFactoryFn	91
9.7	gazebo::physics Namespace Reference	91
9.7.1	Detailed Description	95
9.7.2	Typedef Documentation	95
9.7.2.1	Actor_V	95
9.7.2.2	ActorPtr	95
9.7.2.3	Base_V	95
9.7.2.4	BasePtr	95
9.7.2.5	BoxShapePtr	95
9.7.2.6	Collision_V	95
9.7.2.7	CollisionPtr	95
9.7.2.8	ContactPtr	95
9.7.2.9	CylinderShapePtr	95
9.7.2.10	EntityPtr	95
9.7.2.11	HeightmapShapePtr	95
9.7.2.12	InertialPtr	95
9.7.2.13	Joint_V	95
9.7.2.14	JointController_V	95
9.7.2.15	JointControllerPtr	95
9.7.2.16	JointPtr	95
9.7.2.17	Link_V	95
9.7.2.18	LinkPtr	95
9.7.2.19	MeshShapePtr	95
9.7.2.20	Model_V	96
9.7.2.21	ModelPtr	96
9.7.2.22	MultiRayShapePtr	96
9.7.2.23	PhysicsEnginePtr	96
9.7.2.24	RayShapePtr	96
9.7.2.25	RoadPtr	96
9.7.2.26	ShapePtr	96
9.7.2.27	SphereShapePtr	96
9.7.2.28	SurfaceParamsPtr	96
9.7.2.29	WorldPtr	96
9.8	gazebo::rendering Namespace Reference	96
9.8.1	Detailed Description	98
9.8.2	Typedef Documentation	99

9.8.2.1	ArrowVisualPtr	99
9.8.2.2	AxisVisualPtr	99
9.8.2.3	CameraPtr	99
9.8.2.4	CameraVisualPtr	99
9.8.2.5	COMVisualPtr	99
9.8.2.6	ContactVisualPtr	99
9.8.2.7	DepthCameraPtr	99
9.8.2.8	DynamicLinesPtr	99
9.8.2.9	GpuLaserPtr	99
9.8.2.10	JointVisualPtr	99
9.8.2.11	LaserVisualPtr	99
9.8.2.12	LightPtr	99
9.8.2.13	RFIDTagVisualPtr	99
9.8.2.14	RFIDVisualPtr	99
9.8.2.15	ScenePtr	99
9.8.2.16	UserCameraPtr	99
9.8.2.17	VisualPtr	99
9.8.3	Enumeration Type Documentation	99
9.8.3.1	RenderOpType	99
9.9	gazebo::sensors Namespace Reference	100
9.9.1	Detailed Description	101
9.9.2	Typedef Documentation	102
9.9.2.1	CameraSensor_V	102
9.9.2.2	CameraSensorPtr	102
9.9.2.3	ContactSensor_V	102
9.9.2.4	ContactSensorPtr	102
9.9.2.5	DepthCameraSensor_V	102
9.9.2.6	DepthCameraSensorPtr	102
9.9.2.7	GpuRaySensor_V	102
9.9.2.8	GpuRaySensorPtr	102
9.9.2.9	ImuSensor_V	102
9.9.2.10	ImuSensorPtr	102
9.9.2.11	RaySensor_V	102
9.9.2.12	RaySensorPtr	102
9.9.2.13	RFIDSensor_V	102
9.9.2.14	RFIDSensorPtr	102
9.9.2.15	RFIDTag_V	102

9.9.2.16	RFIDTagPtr	102
9.9.2.17	Sensor_V	102
9.9.2.18	SensorFactoryFn	102
9.9.2.19	SensorPtr	102
9.9.3	Enumeration Type Documentation	102
9.9.3.1	SensorCategory	102
9.10	gazebo::transport Namespace Reference	103
9.10.1	Typedef Documentation	104
9.10.1.1	ConnectionPtr	104
9.10.1.2	NodePtr	104
9.10.1.3	PublicationPtr	104
9.10.1.4	PublicationTransportPtr	104
9.10.1.5	PublisherPtr	104
9.10.1.6	SubscriberPtr	104
9.10.1.7	SubscriptionTransportPtr	105
9.11	gazebo::util Namespace Reference	105
9.11.1	Typedef Documentation	105
9.11.1.1	DiagnosticTimerPtr	105
9.12	google Namespace Reference	105
9.13	google::protobuf Namespace Reference	105
9.14	google::protobuf::compiler Namespace Reference	105
9.15	google::protobuf::compiler::cpp Namespace Reference	105
9.16	Ogre Namespace Reference	106
9.17	ogre Namespace Reference	106
9.18	sdf Namespace Reference	106
9.18.1	Detailed Description	107
9.18.2	Typedef Documentation	107
9.18.2.1	ElementPtr	107
9.18.2.2	ElementPtr_V	107
9.18.2.3	Param_V	107
9.18.2.4	ParamPtr	107
9.18.2.5	SDFPtr	107
9.18.3	Function Documentation	107
9.18.3.1	addNestedModel	107
9.18.3.2	copyChildren	107
9.18.3.3	init	107
9.18.3.4	initDoc	107

9.18.3.5	initDoc	107
9.18.3.6	initFile	107
9.18.3.7	initFile	107
9.18.3.8	initString	107
9.18.3.9	initXml	107
9.18.3.10	readDoc	107
9.18.3.11	readDoc	108
9.18.3.12	readFile	108
9.18.3.13	readString	108
9.18.3.14	readString	108
9.18.3.15	readXml	108
9.19	SkyX Namespace Reference	108
9.20	urdf2gazebo Namespace Reference	108
9.20.1	Detailed Description	108
10	Class Documentation	109
10.1	A Class Reference	109
10.1.1	Detailed Description	109
10.2	gazebo::physics::Actor Class Reference	109
10.2.1	Detailed Description	112
10.2.2	Constructor & Destructor Documentation	112
10.2.2.1	Actor	112
10.2.2.2	~Actor	112
10.2.3	Member Function Documentation	112
10.2.3.1	Fini	112
10.2.3.2	GetSDF	112
10.2.3.3	Init	113
10.2.3.4	IsActive	113
10.2.3.5	Load	113
10.2.3.6	Play	113
10.2.3.7	Stop	113
10.2.3.8	Update	113
10.2.3.9	UpdateParameters	113
10.2.4	Member Data Documentation	114
10.2.4.1	active	114
10.2.4.2	autoStart	114
10.2.4.3	bonePosePub	114

10.2.4.4	interpolateX	114
10.2.4.5	lastPos	114
10.2.4.6	lastScriptTime	114
10.2.4.7	lastTraj	114
10.2.4.8	loop	114
10.2.4.9	mainLink	114
10.2.4.10	mesh	114
10.2.4.11	oldAction	114
10.2.4.12	pathLength	115
10.2.4.13	playStartTime	115
10.2.4.14	prevFrameTime	115
10.2.4.15	scriptLength	115
10.2.4.16	skelAnimation	115
10.2.4.17	skeleton	115
10.2.4.18	skelNodesMap	115
10.2.4.19	skinFile	115
10.2.4.20	skinScale	115
10.2.4.21	startDelay	115
10.2.4.22	trajectories	115
10.2.4.23	trajInfo	116
10.2.4.24	visualName	116
10.3	gazebo::math::Angle Class Reference	116
10.3.1	Detailed Description	117
10.3.2	Constructor & Destructor Documentation	117
10.3.2.1	Angle	117
10.3.2.2	Angle	117
10.3.2.3	Angle	118
10.3.2.4	~Angle	118
10.3.3	Member Function Documentation	118
10.3.3.1	Degree	118
10.3.3.2	Normalize	118
10.3.3.3	operator!=	118
10.3.3.4	operator*	118
10.3.3.5	operator*	118
10.3.3.6	operator*= operator+=	119
10.3.3.7	operator+	119
10.3.3.8	operator+=	119

10.3.3.9	operator-	119
10.3.3.10	operator==	120
10.3.3.11	operator/	120
10.3.3.12	operator/=	120
10.3.3.13	operator<	120
10.3.3.14	operator<=	121
10.3.3.15	operator==	121
10.3.3.16	operator>	121
10.3.3.17	operator>=	121
10.3.3.18	Radian	122
10.3.3.19	SetFromDegree	122
10.3.3.20	SetFromRadian	122
10.3.4	Friends And Related Function Documentation	122
10.3.4.1	operator<<	122
10.3.4.2	operator>>	123
10.4	gazebo::common::Animation Class Reference	123
10.4.1	Detailed Description	124
10.4.2	Member Typedef Documentation	124
10.4.2.1	KeyFrame_V	124
10.4.3	Constructor & Destructor Documentation	125
10.4.3.1	Animation	125
10.4.3.2	~Animation	125
10.4.4	Member Function Documentation	125
10.4.4.1	AddTime	125
10.4.4.2	GetKeyFrame	125
10.4.4.3	GetKeyFrameCount	125
10.4.4.4	GetKeyFramesAtTime	125
10.4.4.5	GetLength	126
10.4.4.6	GetTime	126
10.4.4.7	SetLength	126
10.4.4.8	SetTime	126
10.4.5	Member Data Documentation	126
10.4.5.1	build	126
10.4.5.2	keyFrames	127
10.4.5.3	length	127
10.4.5.4	loop	127
10.4.5.5	name	127

10.4.5.6	timePos	127
10.5	gazebo::rendering::ArrowVisual Class Reference	127
10.5.1	Detailed Description	128
10.5.2	Constructor & Destructor Documentation	128
10.5.2.1	ArrowVisual	128
10.5.2.2	~ArrowVisual	129
10.5.3	Member Function Documentation	129
10.5.3.1	Load	129
10.5.3.2	ShowRotation	129
10.6	gazebo::common::AssertionInternalError Class Reference	129
10.6.1	Detailed Description	130
10.6.2	Constructor & Destructor Documentation	130
10.6.2.1	AssertionInternalError	130
10.6.2.2	~AssertionInternalError	131
10.7	gazebo::rendering::AxisVisual Class Reference	131
10.7.1	Detailed Description	132
10.7.2	Constructor & Destructor Documentation	132
10.7.2.1	AxisVisual	132
10.7.2.2	~AxisVisual	132
10.7.3	Member Function Documentation	132
10.7.3.1	Load	132
10.7.3.2	ScaleXAxis	132
10.7.3.3	ScaleYAxis	133
10.7.3.4	ScaleZAxis	133
10.7.3.5	SetAxisMaterial	133
10.7.3.6	ShowRotation	133
10.8	gazebo::physics::BallJoint< T > Class Template Reference	133
10.8.1	Detailed Description	134
10.8.2	Constructor & Destructor Documentation	134
10.8.2.1	BallJoint	134
10.8.2.2	~BallJoint	135
10.8.3	Member Function Documentation	135
10.8.3.1	GetAngleCount	135
10.8.3.2	GetHighStop	135
10.8.3.3	GetLowStop	135
10.8.3.4	Load	135
10.8.3.5	SetAxis	135

10.8.3.6	SetHighStop	135
10.8.3.7	SetLowStop	135
10.9	gazebo::physics::Base Class Reference	135
10.9.1	Detailed Description	138
10.9.2	Member Enumeration Documentation	138
10.9.2.1	EntityType	138
10.9.3	Constructor & Destructor Documentation	139
10.9.3.1	Base	139
10.9.3.2	~Base	139
10.9.4	Member Function Documentation	139
10.9.4.1	AddChild	139
10.9.4.2	AddType	140
10.9.4.3	Fini	140
10.9.4.4	GetByld	140
10.9.4.5	GetByName	140
10.9.4.6	GetChild	140
10.9.4.7	GetChild	141
10.9.4.8	GetChildCount	141
10.9.4.9	GetId	141
10.9.4.10	GetName	141
10.9.4.11	GetParent	141
10.9.4.12	GetParentId	142
10.9.4.13	GetSaveable	142
10.9.4.14	GetScopedName	142
10.9.4.15	GetSDF	142
10.9.4.16	GetType	142
10.9.4.17	GetWorld	142
10.9.4.18	HasType	143
10.9.4.19	Init	143
10.9.4.20	IsSelected	143
10.9.4.21	Load	143
10.9.4.22	operator==	143
10.9.4.23	Print	144
10.9.4.24	RemoveChild	144
10.9.4.25	RemoveChild	144
10.9.4.26	RemoveChildren	144
10.9.4.27	Reset	144

10.9.4.28	Reset	144
10.9.4.29	SetName	145
10.9.4.30	SetParent	145
10.9.4.31	SetSaveable	145
10.9.4.32	SetSelected	145
10.9.4.33	SetWorld	145
10.9.4.34	Update	146
10.9.4.35	UpdateParameters	146
10.9.5	Member Data Documentation	146
10.9.5.1	children	146
10.9.5.2	childrenEnd	146
10.9.5.3	parent	146
10.9.5.4	sdf	146
10.9.5.5	world	146
10.10	gazebo::math::Box Class Reference	146
10.10.1	Detailed Description	148
10.10.2	Constructor & Destructor Documentation	148
10.10.2.1	Box	148
10.10.2.2	Box	148
10.10.2.3	Box	148
10.10.2.4	~Box	148
10.10.3	Member Function Documentation	148
10.10.3.1	GetCenter	148
10.10.3.2	GetSize	148
10.10.3.3	GetXLength	149
10.10.3.4	GetYLength	149
10.10.3.5	GetZLength	149
10.10.3.6	Merge	149
10.10.3.7	operator+	149
10.10.3.8	operator+=	149
10.10.3.9	operator-	150
10.10.3.10	operator=	150
10.10.3.11	operator==	150
10.10.4	Friends And Related Function Documentation	150
10.10.4.1	operator<<	151
10.10.5	Member Data Documentation	151
10.10.5.1	max	151

10.10.5.2 min	151
10.11 gazebo::physics::BoxShape Class Reference	151
10.11.1 Detailed Description	152
10.11.2 Constructor & Destructor Documentation	153
10.11.2.1 BoxShape	153
10.11.2.2 ~BoxShape	153
10.11.3 Member Function Documentation	153
10.11.3.1 FillMsg	153
10.11.3.2 GetSize	153
10.11.3.3 Init	153
10.11.3.4 ProcessMsg	153
10.11.3.5 SetSize	154
10.12 gazebo::common::BVHLoader Class Reference	154
10.12.1 Detailed Description	154
10.12.2 Constructor & Destructor Documentation	154
10.12.2.1 BVHLoader	154
10.12.2.2 ~BVHLoader	154
10.12.3 Member Function Documentation	154
10.12.3.1 Load	154
10.13 gazebo::transport::CallbackHelper Class Reference	155
10.13.1 Detailed Description	156
10.13.2 Constructor & Destructor Documentation	156
10.13.2.1 CallbackHelper	156
10.13.2.2 ~CallbackHelper	156
10.13.3 Member Function Documentation	156
10.13.3.1 GetId	156
10.13.3.2 GetLatching	156
10.13.3.3 GetMsgType	157
10.13.3.4 HandleData	157
10.13.3.5 IsLocal	157
10.13.4 Member Data Documentation	157
10.13.4.1 latching	157
10.14 gazebo::transport::CallbackHelperT< M > Class Template Reference	157
10.14.1 Detailed Description	158
10.14.2 Constructor & Destructor Documentation	158
10.14.2.1 CallbackHelperT	158
10.14.3 Member Function Documentation	159

10.14.3.1 GetMsgType	159
10.14.3.2 HandleData	159
10.14.3.3 IsLocal	159
10.15 gazebo::rendering::Camera Class Reference	159
10.15.1 Detailed Description	165
10.15.2 Constructor & Destructor Documentation	165
10.15.2.1 Camera	165
10.15.2.2 ~Camera	166
10.15.3 Member Function Documentation	166
10.15.3.1 AnimationComplete	166
10.15.3.2 AttachToVisual	166
10.15.3.3 AttachToVisualImpl	166
10.15.3.4 AttachToVisualImpl	167
10.15.3.5 ConnectNewImageFrame	167
10.15.3.6 CreateRenderTexture	167
10.15.3.7 DisconnectNewImageFrame	167
10.15.3.8 EnableSaveFrame	168
10.15.3.9 Fini	168
10.15.3.10 GetAspectRatio	168
10.15.3.11 GetAvgFPS	168
10.15.3.12 GetCameraToViewportRay	168
10.15.3.13 GetDirection	169
10.15.3.14 GetFarClip	169
10.15.3.15 GetFrameFilename	169
10.15.3.16 GetHFOV	169
10.15.3.17 GetImageByteSize	169
10.15.3.18 GetImageByteSize	169
10.15.3.19 GetImageData	170
10.15.3.20 GetImageDepth	170
10.15.3.21 GetImageFormat	170
10.15.3.22 GetImageHeight	170
10.15.3.23 GetImageWidth	170
10.15.3.24 GetInitialized	171
10.15.3.25 GetLastRenderWallTime	171
10.15.3.26 GetName	171
10.15.3.27 GetNearClip	171
10.15.3.28 GetOgreCamera	171

10.15.3.29GetPitchNode	171
10.15.3.30GetRenderRate	172
10.15.3.31GetRenderTexture	172
10.15.3.32GetRight	172
10.15.3.33GetScene	172
10.15.3.34GetSceneNode	172
10.15.3.35GetTextureHeight	172
10.15.3.36GetTextureWidth	173
10.15.3.37GetTriangleCount	173
10.15.3.38GetUp	173
10.15.3.39GetVFOV	173
10.15.3.40GetViewport	173
10.15.3.41GetViewportHeight	173
10.15.3.42GetViewportWidth	174
10.15.3.43GetWindowId	174
10.15.3.44GetWorldPointOnPlane	174
10.15.3.45GetWorldPose	174
10.15.3.46GetWorldPosition	174
10.15.3.47GetWorldRotation	175
10.15.3.48GetZValue	175
10.15.3.49Init	175
10.15.3.50IsAnimating	175
10.15.3.51IsInitialized	175
10.15.3.52IsVisible	175
10.15.3.53IsVisible	176
10.15.3.54Load	176
10.15.3.55Load	176
10.15.3.56MoveToPosition	176
10.15.3.57MoveToPositions	177
10.15.3.58PostRender	177
10.15.3.59Render	177
10.15.3.60RenderImpl	177
10.15.3.61RotatePitch	177
10.15.3.62RotateYaw	177
10.15.3.63SaveFrame	178
10.15.3.64SaveFrame	178
10.15.3.65SetAspectRatio	178

10.15.3.66SetCaptureData	178
10.15.3.67SetClipDist	178
10.15.3.68SetHFOV	179
10.15.3.69SetImageHeight	179
10.15.3.70SetImageSize	179
10.15.3.71SetImageWidth	179
10.15.3.72SetName	179
10.15.3.73SetRenderRate	180
10.15.3.74SetRenderTarget	180
10.15.3.75SetSaveFramePathname	180
10.15.3.76SetScene	180
10.15.3.77SetSceneNode	180
10.15.3.78SetWindowId	180
10.15.3.79SetWorldPose	180
10.15.3.80SetWorldPosition	181
10.15.3.81SetWorldRotation	181
10.15.3.82ShowWireframe	181
10.15.3.83ToggleShowWireframe	181
10.15.3.84TrackVisual	181
10.15.3.85TrackVisualImpl	181
10.15.3.86TrackVisualImpl	182
10.15.3.87Translate	182
10.15.3.88Update	182
10.15.4 Member Data Documentation	182
10.15.4.1 animState	182
10.15.4.2 bayerFrameBuffer	182
10.15.4.3 camera	182
10.15.4.4 captureData	182
10.15.4.5 connections	183
10.15.4.6 imageFormat	183
10.15.4.7 imageHeight	183
10.15.4.8 imageWidth	183
10.15.4.9 initialized	183
10.15.4.10lastRenderWallTime	183
10.15.4.11name	183
10.15.4.12newData	183
10.15.4.13newImageFrame	183

10.15.4.14	onAnimationComplete	183
10.15.4.15	pitchNode	183
10.15.4.16	prevAnimTime	184
10.15.4.17	renderTarget	184
10.15.4.18	renderTexture	184
10.15.4.19	requests	184
10.15.4.20	saveCount	184
10.15.4.21	saveFrameBuffer	184
10.15.4.22	scene	184
10.15.4.23	sceneNode	184
10.15.4.24	sdf	184
10.15.4.25	textureHeight	184
10.15.4.26	textureWidth	184
10.15.4.27	viewport	184
10.15.4.28	windowId	185
10.16	gazebo::sensors::CameraSensor Class Reference	185
10.16.1	Detailed Description	186
10.16.2	Constructor & Destructor Documentation	186
10.16.2.1	CameraSensor	186
10.16.2.2	~CameraSensor	186
10.16.3	Member Function Documentation	186
10.16.3.1	Fini	186
10.16.3.2	GetCamera	187
10.16.3.3	GetImageData	187
10.16.3.4	GetImageHeight	187
10.16.3.5	GetImageWidth	187
10.16.3.6	GetTopic	187
10.16.3.7	Init	187
10.16.3.8	IsActive	188
10.16.3.9	Load	188
10.16.3.10	Load	188
10.16.3.11	SaveFrame	188
10.16.3.12	SetParent	188
10.16.3.13	UpdateImpl	189
10.17	gazebo::rendering::CameraVisual Class Reference	189
10.17.1	Detailed Description	190
10.17.2	Constructor & Destructor Documentation	190

10.17.2.1 CameraVisual	190
10.17.2.2 ~CameraVisual	190
10.17.3 Member Function Documentation	190
10.17.3.1 Load	190
10.18 gazebo::common::ColladaLoader Class Reference	190
10.18.1 Detailed Description	191
10.18.2 Constructor & Destructor Documentation	191
10.18.2.1 ColladaLoader	191
10.18.2.2 ~ColladaLoader	191
10.18.3 Member Function Documentation	191
10.18.3.1 Load	191
10.19 gazebo::physics::Collision Class Reference	192
10.19.1 Detailed Description	194
10.19.2 Constructor & Destructor Documentation	194
10.19.2.1 Collision	194
10.19.2.2 ~Collision	195
10.19.3 Member Function Documentation	195
10.19.3.1 AddContact	195
10.19.3.2 ConnectContact	195
10.19.3.3 DisconnectContact	195
10.19.3.4 FillMsg	195
10.19.3.5 Fini	195
10.19.3.6 GetBoundingBox	195
10.19.3.7 GetContactsEnabled	196
10.19.3.8 GetLaserRetro	196
10.19.3.9 GetLink	196
10.19.3.10 GetModel	196
10.19.3.11 GetRelativeAngularAccel	196
10.19.3.12 GetRelativeAngularVel	196
10.19.3.13 GetRelativeLinearAccel	197
10.19.3.14 GetRelativeLinearVel	197
10.19.3.15 GetShape	197
10.19.3.16 GetShapeType	197
10.19.3.17 GetState	197
10.19.3.18 GetSurface	198
10.19.3.19 GetWorldAngularAccel	198
10.19.3.20 GetWorldAngularVel	198

10.19.3.21	GetWorldLinearAccel	198
10.19.3.22	GetWorldLinearVel	198
10.19.3.23	Init	198
10.19.3.24	IsPlaceable	199
10.19.3.25	Load	199
10.19.3.26	ProcessMsg	199
10.19.3.27	SetCategoryBits	199
10.19.3.28	SetCollideBits	199
10.19.3.29	SetCollision	199
10.19.3.30	SetContactsEnabled	200
10.19.3.31	SetLaserRetro	200
10.19.3.32	SetShape	200
10.19.3.33	SetState	200
10.19.3.34	UpdateParameters	200
10.19.4	Member Data Documentation	200
10.19.4.1	link	201
10.19.4.2	placeable	201
10.19.4.3	shape	201
10.20	gazebo::physics::CollisionState Class Reference	201
10.20.1	Detailed Description	202
10.20.2	Constructor & Destructor Documentation	202
10.20.2.1	CollisionState	202
10.20.2.2	CollisionState	202
10.20.2.3	CollisionState	202
10.20.2.4	~CollisionState	203
10.20.3	Member Function Documentation	203
10.20.3.1	FillSDF	203
10.20.3.2	GetPose	203
10.20.3.3	IsZero	203
10.20.3.4	Load	203
10.20.3.5	operator+	204
10.20.3.6	operator-	204
10.20.3.7	operator=	204
10.20.4	Friends And Related Function Documentation	204
10.20.4.1	operator<<	204
10.21	gazebo::common::Color Class Reference	205
10.21.1	Detailed Description	207

10.21.2 Member Typedef Documentation	207
10.21.2.1 ABGR	207
10.21.2.2 ARGB	207
10.21.2.3 BGRA	207
10.21.2.4 RGBA	207
10.21.3 Constructor & Destructor Documentation	207
10.21.3.1 Color	208
10.21.3.2 Color	208
10.21.3.3 Color	208
10.21.3.4 ~Color	208
10.21.4 Member Function Documentation	208
10.21.4.1 GetAsABGR	208
10.21.4.2 GetAsARGB	208
10.21.4.3 GetAsBGRA	208
10.21.4.4 GetAsHSV	209
10.21.4.5 GetAsRGBA	209
10.21.4.6 GetAsYUV	209
10.21.4.7 operator!=	209
10.21.4.8 operator*	209
10.21.4.9 operator*	210
10.21.4.10operator*=	210
10.21.4.11operator+	210
10.21.4.12operator+	210
10.21.4.13operator+=	211
10.21.4.14operator-	211
10.21.4.15operator-	211
10.21.4.16operator-=	211
10.21.4.17operator/	212
10.21.4.18operator/	212
10.21.4.19operator/=	212
10.21.4.20operator=	212
10.21.4.21operator==	213
10.21.4.22operator[]	213
10.21.4.23Reset	213
10.21.4.24Set	213
10.21.4.25SetFromABGR	213
10.21.4.26SetFromARGB	214

10.21.4.27SetFromBGRA	214
10.21.4.28SetFromHSV	214
10.21.4.29SetFromRGBA	214
10.21.4.30SetFromYUV	214
10.21.5 Friends And Related Function Documentation	215
10.21.5.1 operator<<	215
10.21.5.2 operator>>	215
10.21.6 Member Data Documentation	215
10.21.6.1 a	215
10.21.6.2 b	215
10.21.6.3 Black	215
10.21.6.4 Blue	215
10.21.6.5 g	215
10.21.6.6 Green	215
10.21.6.7 Purple	215
10.21.6.8 r	216
10.21.6.9 Red	216
10.21.6.10White	216
10.21.6.11Yellow	216
10.22gazebo::rendering::COMVisual Class Reference	216
10.22.1 Detailed Description	217
10.22.2 Constructor & Destructor Documentation	217
10.22.2.1 COMVisual	217
10.22.2.2 ~COMVisual	217
10.22.3 Member Function Documentation	217
10.22.3.1 Load	217
10.22.3.2 Load	217
10.23gazebo::event::Connection Class Reference	218
10.23.1 Detailed Description	218
10.23.2 Constructor & Destructor Documentation	218
10.23.2.1 Connection	218
10.23.2.2 Connection	218
10.23.2.3 ~Connection	218
10.23.3 Member Function Documentation	219
10.23.3.1 GetId	219
10.24gazebo::transport::Connection Class Reference	219
10.24.1 Detailed Description	221

10.24.2 Member Typedef Documentation	221
10.24.2.1 AcceptCallback	221
10.24.2.2 ReadCallback	221
10.24.3 Constructor & Destructor Documentation	221
10.24.3.1 Connection	221
10.24.3.2 ~Connection	221
10.24.4 Member Function Documentation	221
10.24.4.1 AsyncRead	221
10.24.4.2 Cancel	221
10.24.4.3 Connect	221
10.24.4.4 ConnectToShutdown	222
10.24.4.5 DisconnectShutdown	222
10.24.4.6 EnqueueMsg	222
10.24.4.7 GetId	222
10.24.4.8 GetLocalAddress	223
10.24.4.9 GetLocalHostname	223
10.24.4.10GetLocalPort	223
10.24.4.11GetLocalURI	223
10.24.4.12GetRemoteAddress	223
10.24.4.13GetRemoteHostname	223
10.24.4.14GetRemotePort	224
10.24.4.15GetRemoteURI	224
10.24.4.16IsOpen	224
10.24.4.17Listen	224
10.24.4.18ProcessWriteQueue	224
10.24.4.19Read	224
10.24.4.20Shutdown	225
10.24.4.21StartRead	225
10.24.4.22StopRead	225
10.24.4.23ValidateIP	225
10.25gazebo::transport::ConnectionManager Class Reference	225
10.25.1 Detailed Description	227
10.25.2 Member Function Documentation	227
10.25.2.1 Advertise	227
10.25.2.2 ConnectToRemoteHost	227
10.25.2.3 Fini	227
10.25.2.4 GetAllPublishers	227

10.25.2.5	GetTopicNamespaces	228
10.25.2.6	Init	228
10.25.2.7	IsRunning	228
10.25.2.8	RegisterTopicNamespace	228
10.25.2.9	RemoveConnection	228
10.25.2.10	Run	229
10.25.2.11	RunUpdate	229
10.25.2.12	Stop	229
10.25.2.13	Subscribe	229
10.25.2.14	Unadvertise	229
10.25.2.15	Unsubscribe	229
10.25.2.16	Unsubscribe	229
10.25.3	Member Data Documentation	230
10.25.3.1	eventConnections	230
10.26	gazebo::transport::ConnectionReadTask Class Reference	230
10.26.1	Detailed Description	230
10.26.2	Constructor & Destructor Documentation	230
10.26.2.1	ConnectionReadTask	230
10.26.3	Member Function Documentation	231
10.26.3.1	execute	231
10.27	gazebo::common::Console Class Reference	231
10.27.1	Detailed Description	232
10.28	gazebo::physics::Contact Class Reference	232
10.28.1	Detailed Description	233
10.28.2	Constructor & Destructor Documentation	233
10.28.2.1	Contact	233
10.28.2.2	Contact	233
10.28.2.3	~Contact	233
10.28.3	Member Function Documentation	233
10.28.3.1	DebugString	233
10.28.3.2	FillMsg	233
10.28.3.3	operator=	234
10.28.3.4	operator=	234
10.28.3.5	Reset	234
10.28.4	Member Data Documentation	234
10.28.4.1	collision1	234
10.28.4.2	collision2	234

10.28.4.3	count	234
10.28.4.4	depths	234
10.28.4.5	normals	235
10.28.4.6	positions	235
10.28.4.7	time	235
10.28.4.8	wrench	235
10.29	gazebo::physics::ContactManager Class Reference	235
10.29.1	Detailed Description	236
10.29.2	Constructor & Destructor Documentation	236
10.29.2.1	ContactManager	236
10.29.2.2	~ContactManager	236
10.29.3	Member Function Documentation	236
10.29.3.1	Clear	236
10.29.3.2	GetContact	236
10.29.3.3	GetContactCount	236
10.29.3.4	GetContacts	236
10.29.3.5	Init	237
10.29.3.6	NewContact	237
10.29.3.7	PublishContacts	237
10.29.3.8	ResetCount	237
10.30	gazebo::sensors::ContactSensor Class Reference	237
10.30.1	Detailed Description	239
10.30.2	Constructor & Destructor Documentation	239
10.30.2.1	ContactSensor	239
10.30.2.2	~ContactSensor	239
10.30.3	Member Function Documentation	239
10.30.3.1	Fini	239
10.30.3.2	GetCollisionContactCount	239
10.30.3.3	GetCollisionCount	239
10.30.3.4	GetCollisionName	240
10.30.3.5	GetContacts	240
10.30.3.6	GetContacts	241
10.30.3.7	Init	241
10.30.3.8	IsActive	241
10.30.3.9	Load	241
10.30.3.10	Load	241
10.30.3.11	UpdateImpl	242

10.31gazebo::rendering::ContactVisual Class Reference	242
10.31.1 Detailed Description	243
10.31.2 Constructor & Destructor Documentation	243
10.31.2.1 ContactVisual	243
10.31.2.2 ~ContactVisual	243
10.31.3 Member Function Documentation	243
10.31.3.1 SetEnabled	243
10.32gazebo::rendering::Conversions Class Reference	243
10.32.1 Detailed Description	244
10.32.2 Member Function Documentation	244
10.32.2.1 Convert	244
10.32.2.2 Convert	244
10.32.2.3 Convert	244
10.32.2.4 Convert	245
10.32.2.5 Convert	245
10.32.2.6 Convert	245
10.33sdf::Converter Class Reference	245
10.33.1 Detailed Description	246
10.33.2 Member Function Documentation	246
10.33.2.1 Convert	246
10.33.2.2 Convert	246
10.34gazebo::physics::CylinderShape Class Reference	246
10.34.1 Detailed Description	248
10.34.2 Constructor & Destructor Documentation	248
10.34.2.1 CylinderShape	248
10.34.2.2 ~CylinderShape	248
10.34.3 Member Function Documentation	248
10.34.3.1 FillMsg	248
10.34.3.2 GetLength	248
10.34.3.3 GetRadius	248
10.34.3.4 Init	249
10.34.3.5 ProcessMsg	249
10.34.3.6 SetLength	249
10.34.3.7 SetRadius	249
10.34.3.8 SetSize	249
10.35gazebo::rendering::DepthCamera Class Reference	249
10.35.1 Detailed Description	251

10.35.2 Constructor & Destructor Documentation	251
10.35.2.1 DepthCamera	251
10.35.2.2 ~DepthCamera	251
10.35.3 Member Function Documentation	251
10.35.3.1 ConnectNewDepthFrame	251
10.35.3.2 ConnectNewRGBPointCloud	252
10.35.3.3 CreateDepthTexture	252
10.35.3.4 DisconnectNewDepthFrame	252
10.35.3.5 DisconnectNewRGBPointCloud	252
10.35.3.6 Fini	253
10.35.3.7 GetDepthData	253
10.35.3.8 Init	253
10.35.3.9 Load	253
10.35.3.10 Load	253
10.35.3.11 PostRender	253
10.35.3.12 SetDepthTarget	253
10.35.4 Member Data Documentation	254
10.35.4.1 depthTarget	254
10.35.4.2 depthTexture	254
10.35.4.3 depthViewport	254
10.36 gazebo::sensors::DepthCameraSensor Class Reference	254
10.36.1 Constructor & Destructor Documentation	255
10.36.1.1 DepthCameraSensor	255
10.36.1.2 ~DepthCameraSensor	255
10.36.2 Member Function Documentation	255
10.36.2.1 Fini	255
10.36.2.2 GetDepthCamera	256
10.36.2.3 Init	256
10.36.2.4 Load	256
10.36.2.5 Load	256
10.36.2.6 SaveFrame	256
10.36.2.7 SetActive	256
10.36.2.8 SetParent	257
10.36.2.9 UpdateImpl	257
10.37 gazebo::util::DiagnosticManager Class Reference	257
10.37.1 Detailed Description	258
10.37.2 Member Function Documentation	258

10.37.2.1	GetLabel	258
10.37.2.2	GetLogPath	258
10.37.2.3	GetTime	259
10.37.2.4	GetTime	259
10.37.2.5	GetTimerCount	259
10.37.2.6	Init	259
10.37.2.7	Lap	259
10.37.2.8	StartTimer	260
10.37.2.9	StopTimer	260
10.38	gazebo::util::DiagnosticTimer Class Reference	260
10.38.1	Detailed Description	261
10.38.2	Constructor & Destructor Documentation	261
10.38.2.1	DiagnosticTimer	261
10.38.2.2	~DiagnosticTimer	261
10.38.3	Member Function Documentation	261
10.38.3.1	GetName	261
10.38.3.2	Lap	261
10.38.3.3	Start	262
10.38.3.4	Stop	262
10.39	gazebo::rendering::DynamicLines Class Reference	262
10.39.1	Detailed Description	263
10.39.2	Constructor & Destructor Documentation	263
10.39.2.1	DynamicLines	263
10.39.2.2	~DynamicLines	264
10.39.3	Member Function Documentation	264
10.39.3.1	AddPoint	264
10.39.3.2	AddPoint	264
10.39.3.3	Clear	264
10.39.3.4	CreateVertexDeclaration	264
10.39.3.5	FillHardwareBuffers	264
10.39.3.6	GetMovableType	264
10.39.3.7	getMovableType	265
10.39.3.8	GetPoint	265
10.39.3.9	GetPointCount	265
10.39.3.10	SetPoint	265
10.39.3.11	Update	265
10.40	gazebo::rendering::DynamicRenderable Class Reference	265

10.40.1 Detailed Description	267
10.40.2 Constructor & Destructor Documentation	267
10.40.2.1 DynamicRenderable	267
10.40.2.2 ~DynamicRenderable	267
10.40.3 Member Function Documentation	267
10.40.3.1 CreateVertexDeclaration	267
10.40.3.2 FillHardwareBuffers	267
10.40.3.3 getBoundingRadius	268
10.40.3.4 GetOperationType	268
10.40.3.5 getSquaredViewDepth	268
10.40.3.6 Init	268
10.40.3.7 PrepareHardwareBuffers	268
10.40.3.8 SetOperationType	269
10.40.4 Member Data Documentation	269
10.40.4.1 indexBufferCapacity	269
10.40.4.2 vertexBufferCapacity	269
10.41 sdf::Element Class Reference	269
10.41.1 Detailed Description	272
10.41.2 Constructor & Destructor Documentation	272
10.41.2.1 Element	272
10.41.2.2 ~Element	272
10.41.3 Member Function Documentation	272
10.41.3.1 AddAttribute	272
10.41.3.2 AddElement	272
10.41.3.3 AddElementDescription	272
10.41.3.4 AddValue	272
10.41.3.5 ClearElements	272
10.41.3.6 Clone	273
10.41.3.7 Copy	273
10.41.3.8 GetAttribute	273
10.41.3.9 GetAttribute	273
10.41.3.10 GetAttributeCount	273
10.41.3.11 GetAttributeSet	273
10.41.3.12 GetCopyChildren	273
10.41.3.13 GetDescription	273
10.41.3.14 GetElement	273
10.41.3.15 GetElement	273

10.41.3.16	GetElementDescription	273
10.41.3.17	GetElementDescription	273
10.41.3.18	GetElementDescriptionCount	274
10.41.3.19	GetElementImpl	274
10.41.3.20	GetFirstElement	274
10.41.3.21	GetInclude	274
10.41.3.22	GetName	274
10.41.3.23	GetNextElement	274
10.41.3.24	GetParent	274
10.41.3.25	GetRequired	274
10.41.3.26	GetValue	274
10.41.3.27	GetValueBool	274
10.41.3.28	GetValueChar	274
10.41.3.29	GetValueColor	274
10.41.3.30	GetValueDouble	274
10.41.3.31	GetValueFloat	274
10.41.3.32	GetValueInt	274
10.41.3.33	GetValuePose	274
10.41.3.34	GetValueQuaternion	274
10.41.3.35	GetValueString	274
10.41.3.36	GetValueTime	274
10.41.3.37	GetValueUInt	274
10.41.3.38	GetValueVector2d	274
10.41.3.39	GetValueVector3	274
10.41.3.40	HasAttribute	275
10.41.3.41	HasElement	275
10.41.3.42	HasElementDescription	275
10.41.3.43	InsertElement	275
10.41.3.44	PrintDescription	275
10.41.3.45	PrintDocLeftPane	275
10.41.3.46	PrintDocRightPane	275
10.41.3.47	PrintValues	275
10.41.3.48	PrintWiki	275
10.41.3.49	RemoveChild	275
10.41.3.50	RemoveFromParent	276
10.41.3.51	Reset	276
10.41.3.52	Set	276

10.41.3.53Set	276
10.41.3.54Set	276
10.41.3.55Set	276
10.41.3.56Set	276
10.41.3.57Set	276
10.41.3.58Set	276
10.41.3.59Set	276
10.41.3.60Set	276
10.41.3.61Set	276
10.41.3.62Set	276
10.41.3.63Set	276
10.41.3.64Set	276
10.41.3.65Set	276
10.41.3.66Set	276
10.41.3.67SetCopyChildren	276
10.41.3.68SetDescription	276
10.41.3.69SetInclude	276
10.41.3.70SetName	276
10.41.3.71SetParent	276
10.41.3.72SetRequired	276
10.41.3.73ToString	277
10.41.3.74Update	277
10.42gazebo::physics::Entity Class Reference	277
10.42.1 Detailed Description	280
10.42.2 Constructor & Destructor Documentation	280
10.42.2.1 Entity	280
10.42.2.2 ~Entity	280
10.42.3 Member Function Documentation	280
10.42.3.1 Fini	280
10.42.3.2 GetBoundingBox	280
10.42.3.3 GetChildCollision	280
10.42.3.4 GetChildLink	281
10.42.3.5 GetCollisionBoundingBox	281
10.42.3.6 GetDirtyPose	281
10.42.3.7 GetInitialRelativePose	281
10.42.3.8 GetNearestEntityBelow	282
10.42.3.9 GetParentModel	282

10.42.3.10	GetRelativeAngularAccel	282
10.42.3.11	GetRelativeAngularVel	282
10.42.3.12	GetRelativeLinearAccel	282
10.42.3.13	GetRelativeLinearVel	283
10.42.3.14	GetRelativePose	283
10.42.3.15	GetWorldAngularAccel	283
10.42.3.16	GetWorldAngularVel	283
10.42.3.17	GetWorldLinearAccel	283
10.42.3.18	GetWorldLinearVel	284
10.42.3.19	GetWorldPose	284
10.42.3.20	IsCanonicalLink	284
10.42.3.21	IsStatic	284
10.42.3.22	Load	284
10.42.3.23	OnPoseChange	285
10.42.3.24	PlaceOnEntity	285
10.42.3.25	PlaceOnNearestEntityBelow	285
10.42.3.26	Reset	285
10.42.3.27	SetAnimation	285
10.42.3.28	SetAnimation	285
10.42.3.29	SetCanonicalLink	285
10.42.3.30	SetInitialRelativePose	286
10.42.3.31	SetName	286
10.42.3.32	SetRelativePose	286
10.42.3.33	SetStatic	286
10.42.3.34	SetWorldPose	286
10.42.3.35	SetWorldTwist	287
10.42.3.36	StopAnimation	287
10.42.3.37	UpdateParameters	287
10.42.4	Member Data Documentation	287
10.42.4.1	animation	287
10.42.4.2	animationConnection	287
10.42.4.3	animationStartPose	287
10.42.4.4	connections	288
10.42.4.5	dirtyPose	288
10.42.4.6	node	288
10.42.4.7	parentEntity	288
10.42.4.8	poseMsg	288

10.42.4.9 prevAnimationTime	288
10.42.4.10 requestPub	288
10.42.4.11 visPub	288
10.42.4.12 visualMsg	288
10.43 gazebo::event::Event Class Reference	288
10.43.1 Detailed Description	290
10.43.2 Constructor & Destructor Documentation	290
10.43.2.1 ~Event	290
10.43.3 Member Function Documentation	290
10.43.3.1 Disconnect	290
10.43.3.2 Disconnect	290
10.44 gazebo::rendering::Events Class Reference	291
10.44.1 Detailed Description	291
10.44.2 Member Function Documentation	291
10.44.2.1 ConnectCreateScene	291
10.44.2.2 ConnectRemoveScene	292
10.44.2.3 DisconnectCreateScene	292
10.44.2.4 DisconnectRemoveScene	292
10.44.3 Member Data Documentation	292
10.44.3.1 createScene	292
10.44.3.2 removeScene	293
10.45 gazebo::event::Events Class Reference	293
10.45.1 Detailed Description	295
10.45.2 Member Function Documentation	295
10.45.2.1 ConnectAddEntity	295
10.45.2.2 ConnectCreateEntity	296
10.45.2.3 ConnectDeleteEntity	296
10.45.2.4 ConnectDiagTimerStart	296
10.45.2.5 ConnectDiagTimerStop	297
10.45.2.6 ConnectPause	297
10.45.2.7 ConnectPostRender	297
10.45.2.8 ConnectPreRender	297
10.45.2.9 ConnectRender	298
10.45.2.10 ConnectSetSelectedEntity	298
10.45.2.11 ConnectStep	298
10.45.2.12 ConnectStop	299
10.45.2.13 ConnectWorldCreated	299

10.45.2.14	ConnectWorldUpdateBegin	299
10.45.2.15	ConnectWorldUpdateEnd	299
10.45.2.16	ConnectWorldUpdateStart	300
10.45.2.17	DisconnectAddEntity	300
10.45.2.18	DisconnectCreateEntity	300
10.45.2.19	DisconnectDeleteEntity	300
10.45.2.20	DisconnectDiagTimerStart	301
10.45.2.21	DisconnectDiagTimerStop	301
10.45.2.22	DisconnectPause	301
10.45.2.23	DisconnectPostRender	301
10.45.2.24	DisconnectPreRender	302
10.45.2.25	DisconnectRender	302
10.45.2.26	DisconnectSetSelectedEntity	302
10.45.2.27	DisconnectStep	302
10.45.2.28	DisconnectStop	302
10.45.2.29	DisconnectWorldCreated	303
10.45.2.30	DisconnectWorldUpdateBegin	303
10.45.2.31	DisconnectWorldUpdateEnd	303
10.45.2.32	DisconnectWorldUpdateStart	303
10.45.3	Member Data Documentation	303
10.45.3.1	addEntity	303
10.45.3.2	deleteEntity	303
10.45.3.3	diagTimerStart	303
10.45.3.4	diagTimerStop	304
10.45.3.5	entityCreated	304
10.45.3.6	pause	304
10.45.3.7	postRender	304
10.45.3.8	preRender	304
10.45.3.9	render	304
10.45.3.10	setSelectedEntity	304
10.45.3.11	step	304
10.45.3.12	stop	305
10.45.3.13	worldCreated	305
10.45.3.14	worldUpdateBegin	305
10.45.3.15	worldUpdateEnd	305
10.45.3.16	worldUpdateStart	305
10.46	gazebo::event::EventT< T > Class Template Reference	305

10.46.1 Detailed Description	308
10.46.2 Member Function Documentation	308
10.46.2.1 operator()	308
10.46.2.2 operator()	308
10.46.2.3 operator()	308
10.46.2.4 operator()	308
10.46.2.5 operator()	309
10.46.2.6 operator()	309
10.46.2.7 operator()	309
10.46.2.8 operator()	309
10.46.2.9 operator()	310
10.46.2.10 operator()	310
10.46.2.11 operator()	311
10.46.2.12 Signal	311
10.46.2.13 Signal	311
10.46.2.14 Signal	311
10.46.2.15 Signal	311
10.46.2.16 Signal	312
10.46.2.17 Signal	312
10.46.2.18 Signal	312
10.46.2.19 Signal	313
10.46.2.20 Signal	313
10.46.2.21 Signal	313
10.46.2.22 Signal	314
10.47 gazebo::common::Exception Class Reference	314
10.47.1 Detailed Description	315
10.47.2 Constructor & Destructor Documentation	315
10.47.2.1 Exception	315
10.47.2.2 Exception	315
10.47.2.3 ~Exception	315
10.47.3 Member Function Documentation	315
10.47.3.1 GetErrorFile	316
10.47.3.2 GetErrorStr	316
10.47.3.3 Print	316
10.47.4 Friends And Related Function Documentation	316
10.47.4.1 operator<<	316
10.48 gazebo::rendering::FPSViewController Class Reference	316

10.48.1 Detailed Description	317
10.48.2 Constructor & Destructor Documentation	318
10.48.2.1 FPSViewController	318
10.48.2.2 ~FPSViewController	318
10.48.3 Member Function Documentation	318
10.48.3.1 GetTypeString	318
10.48.3.2 HandleKeyPressEvent	318
10.48.3.3 HandleKeyReleaseEvent	318
10.48.3.4 HandleMouseEvent	318
10.48.3.5 Init	319
10.48.3.6 Update	319
10.49urdf2gazebo::GazeboExtension Class Reference	319
10.50google::protobuf::compiler::cpp::GazeboGenerator Class Reference	319
10.50.1 Detailed Description	320
10.50.2 Constructor & Destructor Documentation	320
10.50.2.1 GazeboGenerator	320
10.50.2.2 ~GazeboGenerator	320
10.50.3 Member Function Documentation	320
10.50.3.1 Generate	320
10.51gazebo::rendering::GpuLaser Class Reference	320
10.51.1 Detailed Description	322
10.51.2 Constructor & Destructor Documentation	322
10.51.2.1 GpuLaser	322
10.51.2.2 ~GpuLaser	322
10.51.3 Member Function Documentation	322
10.51.3.1 ConnectNewLaserFrame	322
10.51.3.2 CreateLaserTexture	322
10.51.3.3 DisconnectNewLaserFrame	323
10.51.3.4 Fini	323
10.51.3.5 GetLaserData	323
10.51.3.6 Init	323
10.51.3.7 Load	323
10.51.3.8 Load	323
10.51.3.9 notifyRenderSingleObject	323
10.51.3.10PostRender	323
10.51.3.11SetParentSensor	324
10.51.3.12SetRangeCount	324

10.52gazebo::sensors::GpuRaySensor Class Reference	324
10.52.1 Constructor & Destructor Documentation	327
10.52.1.1 GpuRaySensor	327
10.52.1.2 ~GpuRaySensor	327
10.52.2 Member Function Documentation	328
10.52.2.1 ConnectNewLaserFrame	328
10.52.2.2 DisconnectNewLaserFrame	328
10.52.2.3 Fini	328
10.52.2.4 GetAngleMax	328
10.52.2.5 GetAngleMin	328
10.52.2.6 GetAngleResolution	328
10.52.2.7 GetCameraCount	328
10.52.2.8 GetCosHorzFOV	329
10.52.2.9 GetCosVertFOV	329
10.52.2.10GetFiducial	329
10.52.2.11GetHorzFOV	329
10.52.2.12GetHorzHalfAngle	329
10.52.2.13GetLaserCamera	330
10.52.2.14GetRange	330
10.52.2.15GetRangeCount	330
10.52.2.16GetRangeCountRatio	330
10.52.2.17GetRangeMax	330
10.52.2.18GetRangeMin	331
10.52.2.19GetRangeResolution	331
10.52.2.20GetRanges	331
10.52.2.21GetRayCount	331
10.52.2.22GetRayCountRatio	331
10.52.2.23GetRetro	331
10.52.2.24GetVertFOV	332
10.52.2.25GetVertHalfAngle	332
10.52.2.26GetVerticalAngleMax	332
10.52.2.27GetVerticalAngleMin	332
10.52.2.28GetVerticalRangeCount	332
10.52.2.29GetVerticalRayCount	333
10.52.2.30Init	333
10.52.2.31IsHorizontal	333
10.52.2.32Load	333

10.52.2.33	Load	333
10.52.2.34	SetAngleMax	333
10.52.2.35	SetAngleMin	334
10.52.2.36	SetVerticalAngleMax	334
10.52.2.37	SetVerticalAngleMin	334
10.52.2.38	UpdateImpl	334
10.52.3	Member Data Documentation	334
10.52.3.1	cameraCount	334
10.52.3.2	cameraElem	334
10.52.3.3	chfov	334
10.52.3.4	cvfov	335
10.52.3.5	far	335
10.52.3.6	hfov	335
10.52.3.7	horzElem	335
10.52.3.8	horzHalfAngle	335
10.52.3.9	horzRangeCount	335
10.52.3.10	horzRayCount	335
10.52.3.11	isHorizontal	335
10.52.3.12	near	335
10.52.3.13	rangeCountRatio	335
10.52.3.14	rangeElem	335
10.52.3.15	rayCountRatio	336
10.52.3.16	scanElem	336
10.52.3.17	vertElem	336
10.52.3.18	vertHalfAngle	336
10.52.3.19	vertRangeCount	336
10.52.3.20	vertRayCount	336
10.52.3.21	vfov	336
10.53	gazebo::rendering::Grid Class Reference	336
10.53.1	Detailed Description	337
10.53.2	Constructor & Destructor Documentation	337
10.53.2.1	Grid	337
10.53.2.2	~Grid	338
10.53.3	Member Function Documentation	338
10.53.3.1	Enable	338
10.53.3.2	GetCellCount	338
10.53.3.3	GetCellLength	338

10.53.3.4	GetColor	338
10.53.3.5	GetHeight	338
10.53.3.6	GetLineWidth	338
10.53.3.7	GetSceneNode	339
10.53.3.8	Init	339
10.53.3.9	SetCellCount	339
10.53.3.10	SetCellLength	339
10.53.3.11	SetColor	339
10.53.3.12	SetHeight	339
10.53.3.13	SetLineWidth	339
10.53.3.14	SetUserData	340
10.54	gazebo::physics::Gripper Class Reference	340
10.54.1	Detailed Description	340
10.54.2	Constructor & Destructor Documentation	340
10.54.2.1	Gripper	340
10.54.2.2	~Gripper	341
10.54.3	Member Function Documentation	341
10.54.3.1	Init	341
10.54.3.2	Load	341
10.55	gazebo::rendering::GUIOverlay Class Reference	341
10.55.1	Detailed Description	342
10.55.2	Constructor & Destructor Documentation	342
10.55.2.1	GUIOverlay	342
10.55.2.2	~GUIOverlay	342
10.55.3	Member Function Documentation	342
10.55.3.1	AttachCameraToImage	342
10.55.3.2	AttachCameraToImage	343
10.55.3.3	ButtonCallback	343
10.55.3.4	CreateWindow	343
10.55.3.5	HandleKeyPressEvent	343
10.55.3.6	HandleKeyReleaseEvent	344
10.55.3.7	HandleMouseEvent	344
10.55.3.8	Hide	344
10.55.3.9	Init	344
10.55.3.10	IsInitialized	344
10.55.3.11	LoadLayout	345
10.55.3.12	Resize	345

10.55.3.13Show	345
10.55.3.14Update	345
10.56gazebo::rendering::GzTerrainMatGen Class Reference	345
10.56.1 Constructor & Destructor Documentation	346
10.56.1.1 GzTerrainMatGen	346
10.56.1.2 ~GzTerrainMatGen	346
10.57gazebo::rendering::Heightmap Class Reference	346
10.57.1 Detailed Description	346
10.57.2 Constructor & Destructor Documentation	347
10.57.2.1 Heightmap	347
10.57.2.2 ~Heightmap	347
10.57.3 Member Function Documentation	347
10.57.3.1 GetHeight	347
10.57.3.2 GetOgreTerrain	347
10.57.3.3 Load	347
10.57.3.4 LoadFromMsg	347
10.58gazebo::physics::HeightmapShape Class Reference	348
10.58.1 Detailed Description	349
10.58.2 Constructor & Destructor Documentation	349
10.58.2.1 HeightmapShape	349
10.58.2.2 ~HeightmapShape	350
10.58.3 Member Function Documentation	350
10.58.3.1 FillMsg	350
10.58.3.2 GetHeight	350
10.58.3.3 GetMaxHeight	350
10.58.3.4 GetMinHeight	350
10.58.3.5 GetPos	350
10.58.3.6 GetSize	351
10.58.3.7 GetSubSampling	351
10.58.3.8 GetURI	351
10.58.3.9 GetVertexCount	351
10.58.3.10Init	351
10.58.3.11Load	351
10.58.3.12ProcessMsg	352
10.58.4 Member Data Documentation	352
10.58.4.1 heights	352
10.58.4.2 img	352

10.58.4.3	scale	352
10.58.4.4	subSampling	352
10.58.4.5	vertSize	352
10.59	gazebo::physics::Hinge2Joint< T > Class Template Reference	352
10.59.1	Detailed Description	353
10.59.2	Constructor & Destructor Documentation	353
10.59.2.1	Hinge2Joint	353
10.59.2.2	~Hinge2Joint	354
10.59.3	Member Function Documentation	354
10.59.3.1	GetAngleCount	354
10.59.3.2	Load	354
10.60	gazebo::physics::HingeJoint< T > Class Template Reference	354
10.60.1	Detailed Description	355
10.60.2	Constructor & Destructor Documentation	355
10.60.2.1	HingeJoint	355
10.60.2.2	~HingeJoint	355
10.60.3	Member Function Documentation	355
10.60.3.1	GetAngleCount	355
10.60.3.2	Init	355
10.60.3.3	Load	356
10.61	gazebo::common::Image Class Reference	356
10.61.1	Detailed Description	357
10.61.2	Member Enumeration Documentation	357
10.61.2.1	PixelFormat	357
10.61.3	Constructor & Destructor Documentation	358
10.61.3.1	Image	358
10.61.3.2	~Image	358
10.61.4	Member Function Documentation	358
10.61.4.1	ConvertPixelFormat	358
10.61.4.2	GetAvgColor	358
10.61.4.3	GetBPP	359
10.61.4.4	GetData	359
10.61.4.5	GetFilename	359
10.61.4.6	GetHeight	359
10.61.4.7	GetMaxColor	359
10.61.4.8	GetPitch	359
10.61.4.9	GetPixel	360

10.61.4.10	GetPixelFormat	360
10.61.4.11	GetRGBData	360
10.61.4.12	GetWidth	360
10.61.4.13	Load	360
10.61.4.14	Rescale	360
10.61.4.15	SavePNG	361
10.61.4.16	SetFromData	361
10.61.4.17	Valid	361
10.62	gazebo::sensors::ImuSensor Class Reference	361
10.62.1	Detailed Description	363
10.62.2	Constructor & Destructor Documentation	363
10.62.2.1	ImuSensor	363
10.62.2.2	~ImuSensor	363
10.62.3	Member Function Documentation	363
10.62.3.1	Finalize	363
10.62.3.2	GetAngularVelocity	363
10.62.3.3	GetLinearAcceleration	363
10.62.3.4	GetOrientation	363
10.62.3.5	Init	364
10.62.3.6	Load	364
10.62.3.7	Load	364
10.62.3.8	SetReferencePose	364
10.62.3.9	UpdateImpl	364
10.63	gazebo::physics::Inertial Class Reference	365
10.63.1	Detailed Description	366
10.63.2	Constructor & Destructor Documentation	366
10.63.2.1	Inertial	366
10.63.2.2	Inertial	367
10.63.2.3	Inertial	367
10.63.2.4	~Inertial	367
10.63.3	Member Function Documentation	367
10.63.3.1	GetCoG	367
10.63.3.2	GetIXX	367
10.63.3.3	GetIXY	367
10.63.3.4	GetIXZ	368
10.63.3.5	GetIYY	368
10.63.3.6	GetIYZ	368

10.63.3.7 GetIZZ	368
10.63.3.8 GetMass	368
10.63.3.9 GetPose	368
10.63.3.10GetPrincipalMoments	368
10.63.3.11GetProductsofInertia	369
10.63.3.12Load	369
10.63.3.13operator+	369
10.63.3.14operator+=	369
10.63.3.15operator=	369
10.63.3.16ProcessMsg	370
10.63.3.17Reset	370
10.63.3.18Rotate	370
10.63.3.19SetCoG	370
10.63.3.20SetCoG	370
10.63.3.21SetInertiaMatrix	370
10.63.3.22SetIXX	371
10.63.3.23SetIXY	371
10.63.3.24SetIXZ	371
10.63.3.25SetIYY	371
10.63.3.26SetIYZ	371
10.63.3.27SetIZZ	372
10.63.3.28SetMass	372
10.63.3.29UpdateParameters	372
10.63.4 Friends And Related Function Documentation	372
10.63.4.1 operator<<	372
10.64gazebo::common::InternalError Class Reference	372
10.64.1 Detailed Description	373
10.64.2 Constructor & Destructor Documentation	373
10.64.2.1 InternalError	373
10.64.2.2 InternalError	373
10.64.2.3 ~InternalError	374
10.65gazebo::transport::IOManager Class Reference	374
10.65.1 Detailed Description	374
10.65.2 Constructor & Destructor Documentation	374
10.65.2.1 IOManager	374
10.65.2.2 ~IOManager	375
10.65.3 Member Function Documentation	375

10.65.3.1 DecCount	375
10.65.3.2 GetCount	375
10.65.3.3 GetIO	375
10.65.3.4 IncCount	375
10.65.3.5 Stop	375
10.66gazebo::physics::Joint Class Reference	375
10.66.1 Detailed Description	379
10.66.2 Member Enumeration Documentation	379
10.66.2.1 Attribute	379
10.66.3 Constructor & Destructor Documentation	380
10.66.3.1 Joint	380
10.66.3.2 ~Joint	380
10.66.4 Member Function Documentation	380
10.66.4.1 ApplyDamping	380
10.66.4.2 AreConnected	380
10.66.4.3 Attach	380
10.66.4.4 ConnectJointUpdate	381
10.66.4.5 Detach	381
10.66.4.6 DisconnectJointUpdate	381
10.66.4.7 FillMsg	381
10.66.4.8 GetAnchor	381
10.66.4.9 GetAngle	382
10.66.4.10GetAngleCount	382
10.66.4.11GetAngleImpl	382
10.66.4.12GetAttribute	382
10.66.4.13GetChild	383
10.66.4.14GetEffortLimit	383
10.66.4.15GetForce	383
10.66.4.16GetForce	383
10.66.4.17GetForceTorque	384
10.66.4.18GetForceTorque	384
10.66.4.19GetGlobalAxis	384
10.66.4.20GetHighStop	384
10.66.4.21GetInertiaRatio	385
10.66.4.22GetJointLink	385
10.66.4.23GetLinkForce	385
10.66.4.24GetLinkTorque	385

10.66.4.25	GetLocalAxis	386
10.66.4.26	GetLowerLimit	386
10.66.4.27	GetLowStop	386
10.66.4.28	GetMaxForce	387
10.66.4.29	GetParent	387
10.66.4.30	GetUpperLimit	387
10.66.4.31	GetVelocity	387
10.66.4.32	GetVelocityLimit	388
10.66.4.33	Init	388
10.66.4.34	Load	388
10.66.4.35	Load	388
10.66.4.36	Load	388
10.66.4.37	Reset	389
10.66.4.38	SetAnchor	389
10.66.4.39	SetAngle	389
10.66.4.40	SetAttribute	389
10.66.4.41	SetAxis	389
10.66.4.42	SetDamping	390
10.66.4.43	SetForce	390
10.66.4.44	SetHighStop	390
10.66.4.45	SetLowStop	390
10.66.4.46	SetMaxForce	390
10.66.4.47	SetModel	391
10.66.4.48	SetState	391
10.66.4.49	SetVelocity	391
10.66.4.50	Update	391
10.66.4.51	UpdateParameters	391
10.66.5	Member Data Documentation	391
10.66.5.1	anchorLink	391
10.66.5.2	anchorPos	392
10.66.5.3	anchorPose	392
10.66.5.4	applyDamping	392
10.66.5.5	childLink	392
10.66.5.6	dampingCoefficient	392
10.66.5.7	effortLimit	392
10.66.5.8	forceApplied	392
10.66.5.9	inertiaRatio	392

10.66.5.10lowerLimit	392
10.66.5.11model	392
10.66.5.12parentLink	393
10.66.5.13upperLimit	393
10.66.5.14useCFMDamping	393
10.66.5.15velocityLimit	393
10.67gazebo::physics::JointController Class Reference	393
10.67.1 Detailed Description	393
10.67.2 Constructor & Destructor Documentation	394
10.67.2.1 JointController	394
10.67.3 Member Function Documentation	394
10.67.3.1 AddJoint	394
10.67.3.2 Reset	394
10.67.3.3 SetJointPosition	394
10.67.3.4 SetJointPosition	394
10.67.3.5 SetJointPositions	394
10.67.3.6 Update	395
10.68gazebo::physics::JointState Class Reference	395
10.68.1 Detailed Description	396
10.68.2 Constructor & Destructor Documentation	396
10.68.2.1 JointState	396
10.68.2.2 JointState	396
10.68.2.3 JointState	396
10.68.2.4 ~JointState	397
10.68.3 Member Function Documentation	397
10.68.3.1 FillSDF	397
10.68.3.2 GetAngle	397
10.68.3.3 GetAngleCount	397
10.68.3.4 GetAngles	397
10.68.3.5 IsZero	398
10.68.3.6 Load	398
10.68.3.7 operator+	398
10.68.3.8 operator-	398
10.68.3.9 operator=	398
10.68.4 Friends And Related Function Documentation	399
10.68.4.1 operator<<	399
10.69gazebo::rendering::JointVisual Class Reference	399

10.69.1 Detailed Description	400
10.69.2 Constructor & Destructor Documentation	400
10.69.2.1 JointVisual	400
10.69.2.2 ~JointVisual	401
10.69.3 Member Function Documentation	401
10.69.3.1 Load	401
10.70 gazebo::physics::JointWrench Class Reference	401
10.70.1 Detailed Description	401
10.70.2 Member Function Documentation	402
10.70.2.1 operator=	402
10.70.3 Member Data Documentation	402
10.70.3.1 body1Force	402
10.70.3.2 body1Torque	402
10.70.3.3 body2Force	402
10.70.3.4 body2Torque	402
10.71 gazebo::common::KeyFrame Class Reference	402
10.71.1 Detailed Description	403
10.71.2 Constructor & Destructor Documentation	403
10.71.2.1 KeyFrame	403
10.71.2.2 ~KeyFrame	403
10.71.3 Member Function Documentation	404
10.71.3.1 GetTime	404
10.71.4 Member Data Documentation	404
10.71.4.1 time	404
10.72 gazebo::rendering::LaserVisual Class Reference	404
10.72.1 Detailed Description	405
10.72.2 Constructor & Destructor Documentation	405
10.72.2.1 LaserVisual	405
10.72.2.2 ~LaserVisual	405
10.72.3 Member Function Documentation	405
10.72.3.1 SetEmissive	405
10.73 gazebo::rendering::Light Class Reference	405
10.73.1 Detailed Description	407
10.73.2 Constructor & Destructor Documentation	407
10.73.2.1 Light	407
10.73.2.2 ~Light	407
10.73.3 Member Function Documentation	407

10.73.3.1 FillMsg	407
10.73.3.2 GetDiffuseColor	408
10.73.3.3 GetDirection	408
10.73.3.4 GetName	408
10.73.3.5 GetPosition	408
10.73.3.6 GetSpecularColor	408
10.73.3.7 GetType	408
10.73.3.8 Load	409
10.73.3.9 Load	409
10.73.3.10 LoadFromMsg	409
10.73.3.11 OnPoseChange	409
10.73.3.12 SetAttenuation	409
10.73.3.13 SetCastShadows	409
10.73.3.14 SetDiffuseColor	409
10.73.3.15 SetDirection	410
10.73.3.16 SetLightType	410
10.73.3.17 SetName	410
10.73.3.18 SetPosition	410
10.73.3.19 SetRange	410
10.73.3.20 SetSelected	410
10.73.3.21 SetSpecularColor	411
10.73.3.22 SetSpotFalloff	411
10.73.3.23 SetSpotInnerAngle	411
10.73.3.24 SetSpotOuterAngle	411
10.73.3.25 ShowVisual	411
10.73.3.26 ToggleShowVisual	411
10.73.3.27 UpdateFromMsg	412
10.74 gazebo::physics::Link Class Reference	412
10.74.1 Detailed Description	416
10.74.2 Constructor & Destructor Documentation	416
10.74.2.1 Link	416
10.74.2.2 ~Link	417
10.74.3 Member Function Documentation	417
10.74.3.1 AddChildJoint	417
10.74.3.2 AddForce	417
10.74.3.3 AddForceAtRelativePosition	417
10.74.3.4 AddForceAtWorldPosition	417

10.74.3.5 AddParentJoint 417

10.74.3.6 AddRelativeForce 418

10.74.3.7 AddRelativeTorque 418

10.74.3.8 AddTorque 418

10.74.3.9 AttachStaticModel 418

10.74.3.10ConnectEnabled 418

10.74.3.11DetachAllStaticModels 419

10.74.3.12DetachStaticModel 419

10.74.3.13DisconnectEnabled 419

10.74.3.14FillMsg 419

10.74.3.15Fini 419

10.74.3.16GetAngularDamping 419

10.74.3.17GetBoundingBox 420

10.74.3.18GetChildJointsLinks 420

10.74.3.19GetCollision 420

10.74.3.20GetCollision 420

10.74.3.21GetCollisionById 420

10.74.3.22GetCollisions 421

10.74.3.23GetEnabled 421

10.74.3.24GetGravityMode 421

10.74.3.25GetInertial 421

10.74.3.26GetKinematic 421

10.74.3.27GetLinearDamping 422

10.74.3.28GetModel 422

10.74.3.29GetParentJointsLinks 422

10.74.3.30GetRelativeAngularAccel 422

10.74.3.31GetRelativeAngularVel 422

10.74.3.32GetRelativeForce 422

10.74.3.33GetRelativeLinearAccel 423

10.74.3.34GetRelativeLinearVel 423

10.74.3.35GetRelativeTorque 423

10.74.3.36GetSelfCollide 423

10.74.3.37GetSensorCount 423

10.74.3.38GetSensorName 424

10.74.3.39GetWorldAngularAccel 424

10.74.3.40GetWorldCoGLinearVel 424

10.74.3.41GetWorldCoGPose 424

10.74.3.42	GetWorldForce	424
10.74.3.43	GetWorldLinearAccel	425
10.74.3.44	GetWorldLinearVel	425
10.74.3.45	GetWorldLinearVel	425
10.74.3.46	GetWorldTorque	425
10.74.3.47	Init	425
10.74.3.48	Load	426
10.74.3.49	OnPoseChange	426
10.74.3.50	ProcessMsg	426
10.74.3.51	RemoveChildJoint	426
10.74.3.52	RemoveParentJoint	426
10.74.3.53	Reset	426
10.74.3.54	ResetPhysicsStates	427
10.74.3.55	SetAngularAccel	427
10.74.3.56	SetAngularDamping	427
10.74.3.57	SetAngularVel	427
10.74.3.58	SetAutoDisable	427
10.74.3.59	SetCollideMode	427
10.74.3.60	SetEnabled	428
10.74.3.61	SetForce	428
10.74.3.62	SetGravityMode	428
10.74.3.63	SetInertial	428
10.74.3.64	SetKinematic	428
10.74.3.65	SetLaserRetro	428
10.74.3.66	SetLinearAccel	429
10.74.3.67	SetLinearDamping	429
10.74.3.68	SetLinearVel	429
10.74.3.69	SetSelected	429
10.74.3.70	SetSelfCollide	429
10.74.3.71	SetState	429
10.74.3.72	SetTorque	430
10.74.3.73	Update	430
10.74.3.74	UpdateMass	430
10.74.3.75	UpdateParameters	430
10.74.3.76	UpdateSurface	430
10.74.4	Member Data Documentation	430
10.74.4.1	angularAccel	430

10.74.4.2 attachedModelsOffset	430
10.74.4.3 cgVisuals	430
10.74.4.4 inertial	431
10.74.4.5 linearAccel	431
10.74.4.6 visuals	431
10.75gazebo::physics::LinkState Class Reference	431
10.75.1 Detailed Description	432
10.75.2 Constructor & Destructor Documentation	433
10.75.2.1 LinkState	433
10.75.2.2 LinkState	433
10.75.2.3 LinkState	433
10.75.2.4 ~LinkState	433
10.75.3 Member Function Documentation	433
10.75.3.1 FillSDF	433
10.75.3.2 GetAcceleration	433
10.75.3.3 GetCollisionState	434
10.75.3.4 GetCollisionState	434
10.75.3.5 GetCollisionStateCount	434
10.75.3.6 GetCollisionStates	434
10.75.3.7 GetPose	435
10.75.3.8 GetVelocity	435
10.75.3.9 GetWrench	435
10.75.3.10IsZero	435
10.75.3.11Load	435
10.75.3.12operator+	436
10.75.3.13operator-	436
10.75.3.14operator=	436
10.75.4 Friends And Related Function Documentation	436
10.75.4.1 operator<<	436
10.76gazebo::common::LogPlay Class Reference	437
10.76.1 Member Function Documentation	438
10.76.1.1 GetChunk	438
10.76.1.2 GetChunkCount	438
10.76.1.3 GetEncoding	438
10.76.1.4 GetGazeboVersion	438
10.76.1.5 GetLogVersion	438
10.76.1.6 GetRandSeed	439

10.76.1.7	IsOpen	439
10.76.1.8	Open	439
10.76.1.9	Step	439
10.77	Logplay Class Reference	439
10.77.1	Detailed Description	440
10.78	gazebo::common::LogRecord Class Reference	440
10.78.1	Detailed Description	441
10.78.2	Member Function Documentation	441
10.78.2.1	Add	441
10.78.2.2	Fini	442
10.78.2.3	GetBasePath	442
10.78.2.4	GetEncoding	442
10.78.2.5	GetFilename	442
10.78.2.6	GetFileSize	442
10.78.2.7	GetFirstUpdate	443
10.78.2.8	GetPaused	443
10.78.2.9	GetRunning	443
10.78.2.10	GetRunTime	443
10.78.2.11	Init	443
10.78.2.12	Remove	444
10.78.2.13	SetBasePath	444
10.78.2.14	SetPaused	444
10.78.2.15	Start	444
10.78.2.16	Stop	445
10.79	gazebo::Master Class Reference	445
10.79.1	Detailed Description	445
10.79.2	Constructor & Destructor Documentation	446
10.79.2.1	Master	446
10.79.2.2	~Master	446
10.79.3	Member Function Documentation	446
10.79.3.1	Fini	446
10.79.3.2	Init	446
10.79.3.3	Run	446
10.79.3.4	RunOnce	446
10.79.3.5	RunThread	446
10.79.3.6	Stop	446
10.80	gazebo::common::Material Class Reference	446

10.80.1 Detailed Description	449
10.80.2 Member Enumeration Documentation	449
10.80.2.1 BlendMode	449
10.80.2.2 ShadeMode	449
10.80.3 Constructor & Destructor Documentation	449
10.80.3.1 Material	449
10.80.3.2 ~Material	450
10.80.3.3 Material	450
10.80.4 Member Function Documentation	450
10.80.4.1 GetAmbient	450
10.80.4.2 GetBlendFactors	450
10.80.4.3 GetBlendMode	450
10.80.4.4 GetDepthWrite	450
10.80.4.5 GetDiffuse	450
10.80.4.6 GetEmissive	451
10.80.4.7 GetLighting	451
10.80.4.8 GetName	451
10.80.4.9 GetPointSize	451
10.80.4.10GetShadeMode	451
10.80.4.11GetShininess	451
10.80.4.12GetSpecular	452
10.80.4.13GetTextureImage	452
10.80.4.14GetTransparency	452
10.80.4.15SetAmbient	452
10.80.4.16SetBlendFactors	452
10.80.4.17SetBlendMode	452
10.80.4.18SetDepthWrite	453
10.80.4.19SetDiffuse	453
10.80.4.20SetEmissive	453
10.80.4.21SetLighting	453
10.80.4.22SetPointSize	453
10.80.4.23SetShadeMode	453
10.80.4.24SetShininess	454
10.80.4.25SetSpecular	454
10.80.4.26SetTextureImage	454
10.80.4.27SetTextureImage	454
10.80.4.28SetTransparency	454

10.80.5 Friends And Related Function Documentation	454
10.80.5.1 operator<<	454
10.80.6 Member Data Documentation	455
10.80.6.1 ambient	455
10.80.6.2 blendMode	455
10.80.6.3 BlendModeStr	455
10.80.6.4 diffuse	455
10.80.6.5 emissive	455
10.80.6.6 name	455
10.80.6.7 pointSize	455
10.80.6.8 shadeMode	455
10.80.6.9 ShadeModeStr	455
10.80.6.10shininess	455
10.80.6.11specular	455
10.80.6.12texImage	455
10.80.6.13transparency	456
10.81gazebo::math::Matrix3 Class Reference	456
10.81.1 Detailed Description	457
10.81.2 Constructor & Destructor Documentation	457
10.81.2.1 Matrix3	457
10.81.2.2 Matrix3	457
10.81.2.3 Matrix3	457
10.81.2.4 ~Matrix3	457
10.81.3 Member Function Documentation	457
10.81.3.1 operator==	457
10.81.3.2 operator[]	458
10.81.3.3 operator[]	458
10.81.3.4 SetCol	458
10.81.3.5 SetFromAxes	458
10.81.3.6 SetFromAxis	459
10.81.4 Friends And Related Function Documentation	459
10.81.4.1 operator<<	459
10.81.5 Member Data Documentation	459
10.81.5.1 m	459
10.82gazebo::math::Matrix4 Class Reference	459
10.82.1 Detailed Description	461
10.82.2 Constructor & Destructor Documentation	461

10.82.2.1	Matrix4	461
10.82.2.2	Matrix4	461
10.82.2.3	Matrix4	461
10.82.2.4	~Matrix4	462
10.82.3	Member Function Documentation	462
10.82.3.1	GetAsPose	462
10.82.3.2	GetEulerRotation	462
10.82.3.3	GetRotation	462
10.82.3.4	GetTranslation	462
10.82.3.5	Inverse	462
10.82.3.6	IsAffine	463
10.82.3.7	operator*	463
10.82.3.8	operator*	463
10.82.3.9	operator*	463
10.82.3.10	operator=	463
10.82.3.11	operator=	464
10.82.3.12	operator==	464
10.82.3.13	operator[]	464
10.82.3.14	operator[]	464
10.82.3.15	Set	465
10.82.3.16	SetScale	465
10.82.3.17	SetTranslate	465
10.82.3.18	TransformAffine	466
10.82.4	Friends And Related Function Documentation	466
10.82.4.1	operator<<	466
10.82.5	Member Data Documentation	466
10.82.5.1	IDENTITY	466
10.82.5.2	m	466
10.82.5.3	ZERO	466
10.83	gazebo::common::Mesh Class Reference	466
10.83.1	Detailed Description	468
10.83.2	Constructor & Destructor Documentation	468
10.83.2.1	Mesh	468
10.83.2.2	~Mesh	468
10.83.3	Member Function Documentation	468
10.83.3.1	AddMaterial	468
10.83.3.2	AddSubMesh	468

10.83.3.3 FillArrays	469
10.83.3.4 GenSphericalTexCoord	469
10.83.3.5 GetAABB	469
10.83.3.6 GetIndexCount	469
10.83.3.7 GetMaterial	469
10.83.3.8 GetMaterialCount	470
10.83.3.9 GetMax	470
10.83.3.10GetMin	470
10.83.3.11GetName	470
10.83.3.12GetNormalCount	470
10.83.3.13GetPath	470
10.83.3.14GetSkeleton	471
10.83.3.15GetSubMesh	471
10.83.3.16GetSubMeshCount	471
10.83.3.17GetTexCoordCount	471
10.83.3.18GetVertexCount	471
10.83.3.19HasSkeleton	471
10.83.3.20RecalculateNormals	472
10.83.3.21Scale	472
10.83.3.22SetName	472
10.83.3.23SetPath	472
10.83.3.24SetScale	472
10.83.3.25SetSkeleton	472
10.84gazebo::common::MeshCSG Class Reference	472
10.84.1 Detailed Description	473
10.84.2 Member Enumeration Documentation	473
10.84.2.1 BooleanOperation	473
10.84.3 Constructor & Destructor Documentation	473
10.84.3.1 MeshCSG	473
10.84.3.2 ~MeshCSG	473
10.84.4 Member Function Documentation	473
10.84.4.1 CreateBoolean	473
10.85gazebo::common::MeshLoader Class Reference	474
10.85.1 Detailed Description	474
10.85.2 Constructor & Destructor Documentation	475
10.85.2.1 MeshLoader	475
10.85.2.2 ~MeshLoader	475

10.85.3 Member Function Documentation	475
10.85.3.1 Load	475
10.86 gazebo::common::MeshManager Class Reference	475
10.86.1 Detailed Description	476
10.86.2 Member Function Documentation	477
10.86.2.1 AddMesh	477
10.86.2.2 CreateBox	477
10.86.2.3 CreateCamera	477
10.86.2.4 CreateCone	477
10.86.2.5 CreateCylinder	477
10.86.2.6 CreatePlane	478
10.86.2.7 CreatePlane	478
10.86.2.8 CreateSphere	478
10.86.2.9 CreateTube	478
10.86.2.10 GenSphericalTexCoord	479
10.86.2.11 GetMesh	479
10.86.2.12 GetMeshAABB	479
10.86.2.13 HasMesh	479
10.86.2.14 IsValidFilename	480
10.86.2.15 Load	480
10.87 gazebo::physics::Model Class Reference	480
10.87.1 Detailed Description	484
10.87.2 Constructor & Destructor Documentation	484
10.87.2.1 Model	484
10.87.2.2 ~Model	484
10.87.3 Member Function Documentation	484
10.87.3.1 AttachStaticModel	484
10.87.3.2 DetachStaticModel	484
10.87.3.3 FillMsg	485
10.87.3.4 Fini	485
10.87.3.5 GetAutoDisable	485
10.87.3.6 GetBoundingBox	485
10.87.3.7 GetJoint	485
10.87.3.8 GetJointController	486
10.87.3.9 GetJointCount	486
10.87.3.10 GetJoints	486
10.87.3.11 GetLink	486

10.87.3.12GetLinkById	486
10.87.3.13GetLinks	487
10.87.3.14GetPluginCount	487
10.87.3.15GetRelativeAngularAccel	487
10.87.3.16GetRelativeAngularVel	487
10.87.3.17GetRelativeLinearAccel	487
10.87.3.18GetRelativeLinearVel	488
10.87.3.19GetSDF	488
10.87.3.20GetSensorCount	488
10.87.3.21GetWorldAngularAccel	488
10.87.3.22GetWorldAngularVel	488
10.87.3.23GetWorldLinearAccel	489
10.87.3.24GetWorldLinearVel	489
10.87.3.25Init	489
10.87.3.26Load	489
10.87.3.27LoadJoints	489
10.87.3.28LoadPlugins	489
10.87.3.29OnPoseChange	489
10.87.3.30ProcessMsg	490
10.87.3.31RemoveChild	490
10.87.3.32Reset	490
10.87.3.33SetAngularAccel	490
10.87.3.34SetAngularVel	490
10.87.3.35SetAutoDisable	490
10.87.3.36SetCollideMode	491
10.87.3.37SetEnabled	491
10.87.3.38SetGravityMode	491
10.87.3.39SetJointAnimation	491
10.87.3.40SetJointPosition	491
10.87.3.41SetJointPositions	492
10.87.3.42SetLaserRetro	492
10.87.3.43SetLinearAccel	492
10.87.3.44SetLinearVel	492
10.87.3.45SetLinkWorldPose	492
10.87.3.46SetLinkWorldPose	493
10.87.3.47SetState	493
10.87.3.48StopAnimation	493

10.87.3.49Update	493
10.87.3.50UpdateParameters	493
10.87.4 Member Data Documentation	493
10.87.4.1 attachedModels	493
10.87.4.2 attachedModelsOffset	494
10.88gazebo::common::ModelDatabase Class Reference	494
10.88.1 Detailed Description	495
10.89gazebo::ModelPlugin Class Reference	495
10.89.1 Detailed Description	496
10.89.2 Constructor & Destructor Documentation	496
10.89.2.1 ModelPlugin	496
10.89.2.2 ~ModelPlugin	496
10.89.3 Member Function Documentation	496
10.89.3.1 Init	496
10.89.3.2 Load	496
10.89.3.3 Reset	496
10.90gazebo::physics::ModelState Class Reference	497
10.90.1 Detailed Description	498
10.90.2 Constructor & Destructor Documentation	498
10.90.2.1 ModelState	498
10.90.2.2 ModelState	498
10.90.2.3 ModelState	499
10.90.2.4 ~ModelState	499
10.90.3 Member Function Documentation	499
10.90.3.1 FillSDF	499
10.90.3.2 GetJointState	499
10.90.3.3 GetJointState	499
10.90.3.4 GetJointStateCount	500
10.90.3.5 GetJointStates	500
10.90.3.6 GetLinkState	500
10.90.3.7 GetLinkState	501
10.90.3.8 GetLinkStateCount	501
10.90.3.9 GetLinkStates	501
10.90.3.10GetPose	501
10.90.3.11HasJointState	501
10.90.3.12HasLinkState	502
10.90.3.13sZero	502

10.90.3.14	Load	502
10.90.3.15	operator+	502
10.90.3.16	operator-	503
10.90.3.17	operator=	503
10.90.4	Friends And Related Function Documentation	503
10.90.4.1	operator<<	503
10.91	gazebo::common::MouseEvent Class Reference	503
10.91.1	Detailed Description	504
10.91.2	Member Enumeration Documentation	505
10.91.2.1	Buttons	505
10.91.2.2	EventType	505
10.91.3	Constructor & Destructor Documentation	505
10.91.3.1	MouseEvent	505
10.91.4	Member Data Documentation	505
10.91.4.1	alt	505
10.91.4.2	button	505
10.91.4.3	buttons	505
10.91.4.4	control	505
10.91.4.5	dragging	506
10.91.4.6	moveScale	506
10.91.4.7	pos	506
10.91.4.8	pressPos	506
10.91.4.9	prevPos	506
10.91.4.10	scroll	506
10.91.4.11	shift	506
10.91.4.12	type	506
10.92	gazebo::rendering::MovableText Class Reference	506
10.92.1	Detailed Description	508
10.92.2	Member Enumeration Documentation	508
10.92.2.1	HorizAlign	508
10.92.2.2	VertAlign	509
10.92.3	Constructor & Destructor Documentation	509
10.92.3.1	MovableText	509
10.92.3.2	~MovableText	509
10.92.4	Member Function Documentation	509
10.92.4.1	_setupGeometry	509
10.92.4.2	_updateColors	509

10.92.4.3	GetAABB	509
10.92.4.4	GetBaseline	509
10.92.4.5	getBoundingRadius	509
10.92.4.6	GetCharHeight	509
10.92.4.7	GetColor	510
10.92.4.8	GetFont	510
10.92.4.9	getLights	510
10.92.4.10	getMaterial	510
10.92.4.11	getRenderOperation	510
10.92.4.12	GetShowOnTop	510
10.92.4.13	GetSpaceWidth	510
10.92.4.14	getSquaredViewDepth	510
10.92.4.15	GetText	510
10.92.4.16	getWorldTransforms	511
10.92.4.17	Load	511
10.92.4.18	SetBaseline	511
10.92.4.19	SetCharHeight	511
10.92.4.20	SetColor	511
10.92.4.21	SetFontName	511
10.92.4.22	SetShowOnTop	512
10.92.4.23	SetSpaceWidth	512
10.92.4.24	SetText	512
10.92.4.25	SetTextAlignment	512
10.92.4.26	Update	512
10.92.4.27	visitRenderables	512
10.93	gazebo::msgs::MsgFactory Class Reference	512
10.93.1	Detailed Description	513
10.93.2	Member Function Documentation	513
10.93.2.1	GetMsgTypes	513
10.93.2.2	NewMsg	513
10.93.2.3	RegisterMsg	513
10.94	gazebo::sensors::MultiCameraSensor Class Reference	514
10.94.1	Detailed Description	515
10.94.2	Constructor & Destructor Documentation	515
10.94.2.1	MultiCameraSensor	515
10.94.2.2	~MultiCameraSensor	515
10.94.3	Member Function Documentation	515

10.94.3.1	Fini	515
10.94.3.2	GetCamera	515
10.94.3.3	GetCameraCount	516
10.94.3.4	GetImageData	516
10.94.3.5	GetImageHeight	516
10.94.3.6	GetImageWidth	517
10.94.3.7	GetTopic	517
10.94.3.8	Init	517
10.94.3.9	IsActive	517
10.94.3.10	Load	517
10.94.3.11	SaveFrame	518
10.94.3.12	UpdateImpl	518
10.95	gazebo::physics::MultiRayShape Class Reference	518
10.95.1	Detailed Description	521
10.95.2	Constructor & Destructor Documentation	521
10.95.2.1	MultiRayShape	521
10.95.2.2	~MultiRayShape	521
10.95.3	Member Function Documentation	521
10.95.3.1	AddRay	521
10.95.3.2	ConnectNewLaserScans	521
10.95.3.3	DisconnectNewLaserScans	522
10.95.3.4	FillMsg	522
10.95.3.5	GetFiducial	522
10.95.3.6	GetMaxAngle	522
10.95.3.7	GetMaxRange	522
10.95.3.8	GetMinAngle	523
10.95.3.9	GetMinRange	523
10.95.3.10	GetRange	523
10.95.3.11	GetResRange	523
10.95.3.12	GetRetro	523
10.95.3.13	GetSampleCount	524
10.95.3.14	GetScanResolution	524
10.95.3.15	GetVerticalMaxAngle	524
10.95.3.16	GetVerticalMinAngle	524
10.95.3.17	GetVerticalSampleCount	524
10.95.3.18	GetVerticalScanResolution	524
10.95.3.19	Init	525

10.95.3.20	ProcessMsg	525
10.95.3.21	Update	525
10.95.3.22	UpdateRays	525
10.95.4	Member Data Documentation	525
10.95.4.1	horzElem	525
10.95.4.2	newLaserScans	525
10.95.4.3	offset	525
10.95.4.4	rangeElem	525
10.95.4.5	rayElem	526
10.95.4.6	rays	526
10.95.4.7	scanElem	526
10.95.4.8	vertElem	526
10.96	gazebo::transport::Node Class Reference	526
10.96.1	Detailed Description	527
10.96.2	Constructor & Destructor Documentation	528
10.96.2.1	Node	528
10.96.2.2	~Node	528
10.96.3	Member Function Documentation	528
10.96.3.1	Advertise	528
10.96.3.2	DecodeTopicName	528
10.96.3.3	EncodeTopicName	528
10.96.3.4	Fini	529
10.96.3.5	GetId	529
10.96.3.6	GetMsgType	529
10.96.3.7	GetTopicNamespace	529
10.96.3.8	HandleData	529
10.96.3.9	HasLatchedSubscriber	530
10.96.3.10	Init	530
10.96.3.11	InsertLatchedMsg	530
10.96.3.12	ProcessIncoming	530
10.96.3.13	ProcessPublishers	530
10.96.3.14	RemoveCallback	530
10.96.3.15	Subscribe	531
10.96.3.16	Subscribe	531
10.96.3.17	Subscribe	531
10.96.3.18	Subscribe	532
10.97	gazebo::common::NodeAnimation Class Reference	532

10.97.1 Detailed Description	533
10.97.2 Constructor & Destructor Documentation	533
10.97.2.1 NodeAnimation	533
10.97.2.2 ~NodeAnimation	533
10.97.3 Member Function Documentation	533
10.97.3.1 AddKeyFrame	534
10.97.3.2 AddKeyFrame	534
10.97.3.3 GetFrameAt	534
10.97.3.4 GetFrameCount	534
10.97.3.5 GetKeyFrame	534
10.97.3.6 GetKeyFrame	535
10.97.3.7 GetLength	535
10.97.3.8 GetName	535
10.97.3.9 GetTimeAtX	535
10.97.3.10Scale	535
10.97.3.11SetName	536
10.97.4 Member Data Documentation	536
10.97.4.1 keyFrames	536
10.97.4.2 length	536
10.97.4.3 name	536
10.98gazebo::common::NodeAssignment Struct Reference	536
10.98.1 Detailed Description	536
10.98.2 Member Data Documentation	537
10.98.2.1 nodeIndex	537
10.98.2.2 vertexIndex	537
10.98.2.3 weight	537
10.99gazebo::common::NodeTransform Class Reference	537
10.99.1 Detailed Description	538
10.99.2 Member Enumeration Documentation	538
10.99.2.1 TransformType	538
10.99.3 Constructor & Destructor Documentation	539
10.99.3.1 NodeTransform	539
10.99.3.2 NodeTransform	539
10.99.3.3 ~NodeTransform	539
10.99.4 Member Function Documentation	539
10.99.4.1 Get	539
10.99.4.2 GetSID	539

10.99.4.3	GetType	539
10.99.4.4	operator()	540
10.99.4.5	operator*	540
10.99.4.6	operator*	540
10.99.4.7	PrintSource	540
10.99.4.8	RecalculateMatrix	540
10.99.4.9	Set	540
10.99.4.10	SetComponent	541
10.99.4.11	SetSID	541
10.99.4.12	SetSourceValues	541
10.99.4.13	SetSourceValues	541
10.99.4.14	SetSourceValues	541
10.99.4.15	SetType	541
10.99.5	Member Data Documentation	541
10.99.5.1	sid	541
10.99.5.2	source	542
10.99.5.3	transform	542
10.99.5.4	type	542
10.100	gazebo::common::NumericAnimation Class Reference	542
10.100.1	Detailed Description	543
10.100.2	Constructor & Destructor Documentation	543
10.100.2.1	NumericAnimation	543
10.100.2.2	~NumericAnimation	543
10.100.3	Member Function Documentation	543
10.100.3.1	CreateKeyFrame	543
10.100.3.2	GetInterpolatedKeyFrame	543
10.101	gazebo::common::NumericKeyFrame Class Reference	544
10.101.1	Detailed Description	544
10.101.2	Constructor & Destructor Documentation	544
10.101.2.1	NumericKeyFrame	544
10.101.2.2	~NumericKeyFrame	545
10.101.3	Member Function Documentation	545
10.101.3.1	GetValue	545
10.101.3.2	SetValue	545
10.101.4	Member Data Documentation	545
10.101.4.1	value	545
10.102	gazebo::rendering::OrbitViewController Class Reference	545

10.102.1	Detailed Description	547
10.102.2	Constructor & Destructor Documentation	547
10.102.2.1	OrbitViewController	547
10.102.2.2	~OrbitViewController	547
10.102.3	Member Function Documentation	547
10.102.3.1	GetFocalPoint	547
10.102.3.2	GetTypeString	547
10.102.3.3	HandleKeyPressEvent	547
10.102.3.4	HandleKeyReleaseEvent	548
10.102.3.5	HandleMouseEvent	548
10.102.3.6	Init	548
10.102.3.7	Init	548
10.102.3.8	SetDistance	548
10.102.3.9	SetFocalPoint	548
10.102.3.10	Update	549
10.103	df::Param Class Reference	549
10.103.1	Detailed Description	551
10.103.2	Constructor & Destructor Documentation	551
10.103.2.1	Param	551
10.103.2.2	~Param	551
10.103.3	Member Function Documentation	551
10.103.3.1	Clone	551
10.103.3.2	Get	551
10.103.3.3	Get	551
10.103.3.4	Get	551
10.103.3.5	Get	551
10.103.3.6	Get	551
10.103.3.7	Get	551
10.103.3.8	Get	552
10.103.3.9	Get	552
10.103.3.10	Get	552
10.103.3.11	Get	552
10.103.3.12	Get	552
10.103.3.13	Get	552
10.103.3.14	Get	552
10.103.3.15	Get	552
10.103.3.16	GetAsString	552

10.103.3.10	GetDefaultAsString	552
10.103.3.10	GetDescription	552
10.103.3.10	GetKey	552
10.103.3.20	GetRequired	552
10.103.3.20	GetSet	552
10.103.3.20	GetTypeName	552
10.103.3.23	Bool	552
10.103.3.24	Char	552
10.103.3.25	Color	553
10.103.3.26	Double	553
10.103.3.27	Float	553
10.103.3.28	Int	553
10.103.3.29	Pose	553
10.103.3.30	Quaternion	553
10.103.3.31	Str	553
10.103.3.32	Time	553
10.103.3.33	UInt	553
10.103.3.34	Vector2d	553
10.103.3.35	Vector2i	553
10.103.3.36	Vector3	553
10.103.3.37	Reset	553
10.103.3.38	Set	553
10.103.3.39	Set	553
10.103.3.40	Set	553
10.103.3.41	Set	553
10.103.3.42	Set	553
10.103.3.43	Set	553
10.103.3.44	Set	553
10.103.3.45	Set	553
10.103.3.46	Set	553
10.103.3.47	Set	553
10.103.3.48	Set	554
10.103.3.49	Set	554
10.103.3.50	Set	554
10.103.3.51	Set	554
10.103.3.52	Set	554
10.103.3.53	SetDescription	554

10.103.3.58	SetFromString	554
10.103.3.55	SetUpdateFunc	554
10.103.3.56	Update	554
10.103.4	Member Data Documentation	554
10.103.4.1	description	554
10.103.4.2	key	554
10.103.4.3	required	554
10.103.4.4	set	554
10.103.4.5	typeName	555
10.103.4.6	updateFunc	555
10.104	ParamT< T > Class Template Reference	555
10.105df::	ParamT< T > Class Template Reference	555
10.105.1	Detailed Description	556
10.105.2	Constructor & Destructor Documentation	556
10.105.2.1	ParamT	556
10.105.2.2	~ParamT	556
10.105.3	Member Function Documentation	557
10.105.3.1	Clone	557
10.105.3.2	GetAsString	557
10.105.3.3	GetDefaultAsString	557
10.105.3.4	GetDefaultValue	557
10.105.3.5	GetValue	557
10.105.3.6	operator*	557
10.105.3.7	Reset	557
10.105.3.8	Set	557
10.105.3.9	SetFromString	557
10.105.3.10	SetValue	558
10.105.3.11	Update	558
10.105.4	Friends And Related Function Documentation	558
10.105.4.1	operator<<	558
10.105.5	Member Data Documentation	558
10.105.5.1	defaultValue	558
10.105.5.2	value	558
10.106	Gazebo::physics::PhysicsEngine Class Reference	558
10.106.1	Detailed Description	561
10.106.2	Constructor & Destructor Documentation	561
10.106.2.1	PhysicsEngine	561

10.106.2.2~PhysicsEngine	561
10.106.3 Member Function Documentation	561
10.106.3.1CreateCollision	561
10.106.3.2CreateCollision	561
10.106.3.3CreateJoint	562
10.106.3.4CreateLink	562
10.106.3.5CreateShape	562
10.106.3.6DebugPrint	562
10.106.3.7Fini	562
10.106.3.8GetAutoDisableFlag	562
10.106.3.9GetContactManager	563
10.106.3.10GetContactMaxCorrectingVel	563
10.106.3.11GetContactSurfaceLayer	563
10.106.3.12GetGravity	563
10.106.3.13GetMaxContacts	563
10.106.3.14GetPhysicsUpdateMutex	564
10.106.3.15GetSORPGSIters	564
10.106.3.16GetSORPGSPreconIters	564
10.106.3.17GetSORPGSW	564
10.106.3.18GetStepTime	564
10.106.3.19GetType	565
10.106.3.20GetUpdatePeriod	565
10.106.3.21GetUpdateRate	565
10.106.3.22GetWorldCFM	565
10.106.3.23GetWorldERP	565
10.106.3.24Hit	565
10.106.3.25HitForThread	566
10.106.3.26Load	566
10.106.3.27nPhysicsMsg	566
10.106.3.28nRequest	566
10.106.3.29Reset	566
10.106.3.30SetAutoDisableFlag	566
10.106.3.31SetContactMaxCorrectingVel	566
10.106.3.32SetContactSurfaceLayer	567
10.106.3.33SetGravity	567
10.106.3.34SetMaxContacts	567
10.106.3.35SetSeed	567

10.106.3.36	SetSORPGSIters	567
10.106.3.37	SetSORPGSPreconIters	568
10.106.3.38	SetSORPGSW	568
10.106.3.39	SetStepTime	568
10.106.3.40	SetUpdateRate	568
10.106.3.41	SetWorldCFM	568
10.106.3.42	SetWorldERP	568
10.106.3.43	UpdateCollision	569
10.106.3.44	UpdatePhysics	569
10.106.4	Member Data Documentation	569
10.106.4.1	contactManager	569
10.106.4.2	node	569
10.106.4.3	physicsSub	569
10.106.4.4	physicsUpdateMutex	569
10.106.4.5	requestSub	569
10.106.4.6	responsePub	569
10.106.4.7	sdf	569
10.106.4.8	world	570
10.107	gazebo::physics::PhysicsFactory Class Reference	570
10.107.1	Detailed Description	570
10.107.2	Member Function Documentation	570
10.107.2.1	IsRegistered	570
10.107.2.2	NewPhysicsEngine	570
10.107.2.3	RegisterAll	571
10.107.2.4	RegisterPhysicsEngine	571
10.108	gazebo::common::PID Class Reference	571
10.108.1	Detailed Description	572
10.108.2	Constructor & Destructor Documentation	572
10.108.2.1	PID	572
10.108.2.2	~PID	572
10.108.3	Member Function Documentation	572
10.108.3.1	GetCmd	573
10.108.3.2	GetErrors	573
10.108.3.3	Init	573
10.108.3.4	operator=	573
10.108.3.5	Reset	573
10.108.3.6	SetCmd	574

10.108.3.7	SetCmdMax	574
10.108.3.8	SetCmdMin	574
10.108.3.9	SetDGain	574
10.108.3.10	SetIGain	574
10.108.3.11	SetIMax	574
10.108.3.12	SetIMin	575
10.108.3.13	SetPGain	575
10.108.3.14	Update	575
10.109	gazebo::math::Plane Class Reference	575
10.109.1	Detailed Description	576
10.109.2	Constructor & Destructor Documentation	576
10.109.2.1	Plane	576
10.109.2.2	Plane	576
10.109.2.3	Plane	576
10.109.2.4	~Plane	577
10.109.3	Member Function Documentation	577
10.109.3.1	Distance	577
10.109.3.2	operator=	577
10.109.3.3	Set	577
10.109.4	Member Data Documentation	577
10.109.4.1	id	577
10.109.4.2	normal	578
10.109.4.3	size	578
10.110	gazebo::physics::PlaneShape Class Reference	578
10.110.1	Detailed Description	579
10.110.2	Constructor & Destructor Documentation	579
10.110.2.1	PlaneShape	579
10.110.2.2	~PlaneShape	579
10.110.3	Member Function Documentation	579
10.110.3.1	CreatePlane	579
10.110.3.2	FillMsg	580
10.110.3.3	GetNormal	580
10.110.3.4	GetSize	580
10.110.3.5	Init	580
10.110.3.6	ProcessMsg	580
10.110.3.7	SetAltitude	580
10.110.3.8	SetNormal	581

10.110.3.9	SetSize	581
10.111	df::Plugin Class Reference	581
10.111.1	Constructor & Destructor Documentation	582
10.111.1.1	Plugin	582
10.111.2	Member Function Documentation	582
10.111.2.1	Clear	582
10.111.2.2	Print	582
10.111.3	Member Data Documentation	582
10.111.3.1	data	582
10.111.3.2	filename	582
10.111.3.3	name	582
10.112	gazebo::PluginT< T > Class Template Reference	582
10.112.1	Detailed Description	583
10.112.2	Member Typedef Documentation	583
10.112.2.1	TPtr	583
10.112.3	Member Function Documentation	583
10.112.3.1	Create	583
10.112.3.2	GetFilename	584
10.112.3.3	GetHandle	584
10.112.3.4	GetType	584
10.112.4	Member Data Documentation	584
10.112.4.1	filename	584
10.112.4.2	handle	584
10.112.4.3	type	584
10.113	gazebo::math::Pose Class Reference	584
10.113.1	Detailed Description	586
10.113.2	Constructor & Destructor Documentation	586
10.113.2.1	Pose	586
10.113.2.2	Pose	586
10.113.2.3	Pose	587
10.113.2.4	Pose	587
10.113.2.5	~Pose	587
10.113.3	Member Function Documentation	587
10.113.3.1	CoordPoseSolve	587
10.113.3.2	CoordPositionAdd	587
10.113.3.3	CoordPositionAdd	588
10.113.3.4	CoordPositionSub	588

10.113.3.5	CoordRotationAdd	588
10.113.3.6	CoordRotationSub	588
10.113.3.7	Correct	589
10.113.3.8	GetInverse	589
10.113.3.9	IsFinite	589
10.113.3.10	operator!=	589
10.113.3.11	operator*	589
10.113.3.12	operator+	590
10.113.3.13	operator+=	590
10.113.3.14	operator-	590
10.113.3.15	operator-	590
10.113.3.16	operator-=	590
10.113.3.17	operator==	591
10.113.3.18	Reset	591
10.113.3.19	RotatePositionAboutOrigin	591
10.113.3.20	Round	591
10.113.3.21	Set	591
10.113.3.22	Set	592
10.113.4	Friends And Related Function Documentation	592
10.113.4.1	operator<<	592
10.113.4.2	operator>>	592
10.113.5	Member Data Documentation	592
10.113.5.1	pos	592
10.113.5.2	rot	593
10.113.5.3	Zero	593
10.114	gazebo::common::PoseAnimation Class Reference	593
10.114.1	Detailed Description	594
10.114.2	Constructor & Destructor Documentation	594
10.114.2.1	PoseAnimation	594
10.114.2.2	~PoseAnimation	594
10.114.3	Member Function Documentation	594
10.114.3.1	BuildInterpolationSplines	594
10.114.3.2	CreateKeyFrame	594
10.114.3.3	GetInterpolatedKeyFrame	595
10.114.3.4	GetInterpolatedKeyFrame	595
10.115	gazebo::common::PoseKeyFrame Class Reference	595
10.115.1	Detailed Description	596

10.115.2	Constructor & Destructor Documentation	596
10.115.2.1	PoseKeyFrame	596
10.115.2.2	~PoseKeyFrame	596
10.115.3	Member Function Documentation	596
10.115.3.1	GetRotation	596
10.115.3.2	GetTranslation	596
10.115.3.3	SetRotation	597
10.115.3.4	SetTranslation	597
10.115.4	Member Data Documentation	597
10.115.4.1	rotate	597
10.115.4.2	translate	597
10.116	gazebo::rendering::Projector Class Reference	597
10.116.1	Detailed Description	598
10.116.2	Constructor & Destructor Documentation	598
10.116.2.1	Projector	598
10.116.2.2	~Projector	598
10.116.3	Member Function Documentation	598
10.116.3.1	GetParent	598
10.116.3.2	Load	598
10.116.3.3	Load	599
10.116.3.4	Load	599
10.116.3.5	SetEnabled	599
10.116.3.6	SetTexture	599
10.116.3.7	Toggle	599
10.117	gazebo::transport::Publication Class Reference	600
10.117.1	Detailed Description	601
10.117.2	Constructor & Destructor Documentation	601
10.117.2.1	Publication	601
10.117.2.2	~Publication	601
10.117.3	Member Function Documentation	601
10.117.3.1	AddPublisher	601
10.117.3.2	AddSubscription	601
10.117.3.3	AddSubscription	601
10.117.3.4	AddTransport	602
10.117.3.5	GetCallbackCount	602
10.117.3.6	GetLocallyAdvertised	602
10.117.3.7	GetMsgType	602

10.117.3.8	GetNodeCount	602
10.117.3.9	GetRemoteSubscriptionCount	602
10.117.3.10	GetTransportCount	603
10.117.3.11	HasTransport	603
10.117.3.12	LocalPublish	603
10.117.3.13	Publish	603
10.117.3.14	RemoveSubscription	603
10.117.3.15	RemoveSubscription	604
10.117.3.16	RemoveTransport	604
10.117.3.17	SetLocallyAdvertised	604
10.118	Gazebo::transport::PublicationTransport Class Reference	604
10.118.1	Detailed Description	605
10.118.2	Constructor & Destructor Documentation	605
10.118.2.1	PublicationTransport	605
10.118.2.2	~PublicationTransport	605
10.118.3	Member Function Documentation	605
10.118.3.1	AddCallback	605
10.118.3.2	Fin	605
10.118.3.3	GetConnection	605
10.118.3.4	GetMsgType	606
10.118.3.5	GetTopic	606
10.118.3.6	Init	606
10.119	Gazebo::transport::Publisher Class Reference	606
10.119.1	Detailed Description	607
10.119.2	Constructor & Destructor Documentation	607
10.119.2.1	Publisher	607
10.119.2.2	Publisher	607
10.119.2.3	~Publisher	608
10.119.3	Member Function Documentation	608
10.119.3.1	GetLatching	608
10.119.3.2	GetMsgType	608
10.119.3.3	GetOutgoingCount	608
10.119.3.4	GetPrevMsg	608
10.119.3.5	GetTopic	608
10.119.3.6	HasConnections	608
10.119.3.7	Publish	609
10.119.3.8	Publish	609

10.119.3.9	SendMessage	609
10.119.3.10	SetPublication	609
10.119.3.11	WaitForConnection	609
10.120	gazebo::math::Quaternion Class Reference	609
10.120.1	Detailed Description	612
10.120.2	Constructor & Destructor Documentation	612
10.120.2.1	Quaternion	612
10.120.2.2	Quaternion	613
10.120.2.3	Quaternion	613
10.120.2.4	Quaternion	613
10.120.2.5	Quaternion	613
10.120.2.6	Quaternion	613
10.120.2.7	~Quaternion	614
10.120.3	Member Function Documentation	614
10.120.3.1	Correct	614
10.120.3.2	Dot	614
10.120.3.3	EulerToQuaternion	614
10.120.3.4	EulerToQuaternion	614
10.120.3.5	GetAsAxis	614
10.120.3.6	GetAsEuler	615
10.120.3.7	GetAsMatrix3	615
10.120.3.8	GetAsMatrix4	615
10.120.3.9	GetExp	615
10.120.3.10	GetInverse	615
10.120.3.11	GetLog	615
10.120.3.12	GetPitch	616
10.120.3.13	GetRoll	616
10.120.3.14	GetXAxis	616
10.120.3.15	GetYaw	616
10.120.3.16	GetYAxis	616
10.120.3.17	GetZAxis	616
10.120.3.18	Invert	617
10.120.3.19	IsFinite	617
10.120.3.20	Normalize	617
10.120.3.21	operator!=	617
10.120.3.22	operator*	617
10.120.3.23	operator*	617

10.120.3.24	operator*	618
10.120.3.25	operator*= operator*	618
10.120.3.26	operator+ operator*	618
10.120.3.27	operator+=	618
10.120.3.28	operator- operator*	619
10.120.3.29	operator- operator*	619
10.120.3.30	operator-=	619
10.120.3.31	operator=	619
10.120.3.32	operator==	619
10.120.3.33	RotateVector	620
10.120.3.34	RotateVectorReverse	620
10.120.3.35	Round	620
10.120.3.36	Scale	620
10.120.3.37	Set	621
10.120.3.38	SetFromAxis	621
10.120.3.39	SetFromAxis	621
10.120.3.40	SetFromEuler	621
10.120.3.41	SetFromEuler	621
10.120.3.42	SetToIdentity	622
10.120.3.43	Slerp	622
10.120.3.44	Squad	622
10.120.4	Friends And Related Function Documentation	622
10.120.4.1	operator<<	622
10.120.4.2	operator>>	622
10.120.5	Member Data Documentation	623
10.120.5.1w		623
10.120.5.2x		623
10.120.5.3y		623
10.120.5.4z		623
10.120.6	Gazebo::math::Rand Class Reference	623
10.121.1	Detailed Description	624
10.121.2	Member Function Documentation	624
10.121.2.1	GetDbfNormal	624
10.121.2.2	GetDbfUniform	624
10.121.2.3	GetIntNormal	624
10.121.2.4	GetIntUniform	624
10.121.2.5	GetSeed	625

10.121.2.6	SetSeed	625
10.122	gazebo::transport::RawCallbackHelper Class Reference	625
10.122.1	Detailed Description	626
10.122.2	Constructor & Destructor Documentation	626
10.122.2.1	RawCallbackHelper	626
10.122.3	Member Function Documentation	626
10.122.3.1	GetMsgType	626
10.122.3.2	HandleData	626
10.122.3.3	IsLocal	627
10.123	gazebo::sensors::RaySensor Class Reference	627
10.123.1	Detailed Description	629
10.123.2	Constructor & Destructor Documentation	629
10.123.2.1	RaySensor	629
10.123.2.2	~RaySensor	629
10.123.3	Member Function Documentation	629
10.123.3.1	Fini	629
10.123.3.2	GetAngleMax	629
10.123.3.3	GetAngleMin	629
10.123.3.4	GetAngleResolution	630
10.123.3.5	GetFiducial	630
10.123.3.6	GetLaserShape	630
10.123.3.7	GetRange	630
10.123.3.8	GetRangeCount	631
10.123.3.9	GetRangeMax	631
10.123.3.10	GetRangeMin	631
10.123.3.11	GetRangeResolution	631
10.123.3.12	GetRanges	631
10.123.3.13	GetRayCount	631
10.123.3.14	GetRetro	632
10.123.3.15	GetTopic	632
10.123.3.16	GetVerticalAngleMax	632
10.123.3.17	GetVerticalAngleMin	632
10.123.3.18	GetVerticalRangeCount	633
10.123.3.19	GetVerticalRayCount	633
10.123.3.20	Init	633
10.123.3.21	IsActive	633
10.123.3.22	IsBad	633

10.123.3.23updateImpl	633
10.124 Gazebo::physics::RayShape Class Reference	634
10.124.1 Detailed Description	636
10.124.2 Constructor & Destructor Documentation	636
10.124.2.1 RayShape	636
10.124.2.2 RayShape	636
10.124.2.3 ~RayShape	636
10.124.3 Member Function Documentation	636
10.124.3.1 FillMsg	636
10.124.3.2 GetFiducial	636
10.124.3.3 GetGlobalPoints	637
10.124.3.4 GetIntersection	637
10.124.3.5 GetLength	637
10.124.3.6 GetRelativePoints	637
10.124.3.7 GetRetro	637
10.124.3.8 Init	637
10.124.3.9 ProcessMsg	638
10.124.3.10 SetFiducial	638
10.124.3.11 SetLength	638
10.124.3.12 SetPoints	638
10.124.3.13 SetRetro	638
10.124.3.14 Update	639
10.124.4 Member Data Documentation	639
10.124.4.1 contactFiducial	639
10.124.4.2 contactLen	639
10.124.4.3 contactRetro	639
10.124.4.4 globalEndPos	639
10.124.4.5 globalStartPos	639
10.124.4.6 relativeEndPos	639
10.124.4.7 relativeStartPos	639
10.125 Gazebo::rendering::RenderEngine Class Reference	639
10.125.1 Detailed Description	641
10.125.2 Member Enumeration Documentation	641
10.125.2.1 RenderPathType	641
10.125.3 Member Function Documentation	641
10.125.3.1 AddResourcePath	641
10.125.3.2 CreateScene	642

10.125.3.3	Fin	642
10.125.3.4	GetRenderPathType	642
10.125.3.5	GetScene	642
10.125.3.6	GetScene	642
10.125.3.7	GetSceneCount	643
10.125.3.8	init	643
10.125.3.9	Load	643
10.125.3.10	RemoveScene	643
10.125.4	Member Data Documentation	643
10.125.4.1	dummyContext	643
10.125.4.2	dummyDisplay	643
10.125.4.3	dummyWindowId	643
10.125.4.4	root	643
10.126	gazebo::sensors::RFIDSensor Class Reference	644
10.126.1	Detailed Description	645
10.126.2	Constructor & Destructor Documentation	645
10.126.2.1	RFIDSensor	645
10.126.2.2	~RFIDSensor	645
10.126.3	Member Function Documentation	645
10.126.3.1	AddTag	645
10.126.3.2	Fin	645
10.126.3.3	init	645
10.126.3.4	Load	645
10.126.3.5	Load	645
10.126.3.6	UpdateImpl	646
10.127	gazebo::sensors::RFIDTag Class Reference	646
10.127.1	Detailed Description	647
10.127.2	Constructor & Destructor Documentation	647
10.127.2.1	RFIDTag	647
10.127.2.2	~RFIDTag	647
10.127.3	Member Function Documentation	647
10.127.3.1	Fin	647
10.127.3.2	GetTagPose	648
10.127.3.3	init	648
10.127.3.4	Load	648
10.127.3.5	Load	648
10.127.3.6	UpdateImpl	648

10.128	<code>gazebo::rendering::RFIDTagVisual</code> Class Reference	648
10.128.1	Detailed Description	649
10.128.2	Constructor & Destructor Documentation	649
10.128.2.1	<code>RFIDTagVisual</code>	649
10.128.2.2	<code>~RFIDTagVisual</code>	650
10.129	<code>gazebo::rendering::RFIDVisual</code> Class Reference	650
10.129.1	Detailed Description	651
10.129.2	Constructor & Destructor Documentation	651
10.129.2.1	<code>RFIDVisual</code>	651
10.129.2.2	<code>~RFIDVisual</code>	651
10.130	<code>Road</code> Class Reference	651
10.130.1	Detailed Description	651
10.131	<code>gazebo::physics::Road</code> Class Reference	651
10.131.1	Detailed Description	652
10.131.2	Constructor & Destructor Documentation	652
10.131.2.1	<code>Road</code>	652
10.131.2.2	<code>~Road</code>	653
10.131.3	Member Function Documentation	653
10.131.3.1	<code>Init</code>	653
10.131.3.2	<code>Load</code>	653
10.132	<code>gazebo::rendering::Road2d</code> Class Reference	653
10.132.1	Constructor & Destructor Documentation	653
10.132.1.1	<code>Road2d</code>	653
10.132.1.2	<code>~Road2d</code>	653
10.132.2	Member Function Documentation	654
10.132.2.1	<code>Load</code>	654
10.133	<code>gazebo::math::RotationSpline</code> Class Reference	654
10.133.1	Detailed Description	655
10.133.2	Constructor & Destructor Documentation	655
10.133.2.1	<code>RotationSpline</code>	655
10.133.2.2	<code>~RotationSpline</code>	655
10.133.3	Member Function Documentation	655
10.133.3.1	<code>AddPoint</code>	655
10.133.3.2	<code>Clear</code>	655
10.133.3.3	<code>GetNumPoints</code>	655
10.133.3.4	<code>GetPoint</code>	655
10.133.3.5	<code>Interpolate</code>	656

10.133.3.6	Interpolate	656
10.133.3.7	RecalcTangents	656
10.133.3.8	SetAutoCalculate	657
10.133.3.9	UpdatePoint	657
10.133.4	Member Data Documentation	657
10.133.4.1	autoCalc	657
10.133.4.2	points	657
10.133.4.3	tangents	657
10.134	gazebo::rendering::RTShaderSystem Class Reference	658
10.134.1	Detailed Description	659
10.134.2	Member Enumeration Documentation	659
10.134.2.1	LightingModel	659
10.134.3	Member Function Documentation	659
10.134.3.1	AddScene	659
10.134.3.2	ApplyShadows	660
10.134.3.3	AttachEntity	660
10.134.3.4	AttachViewport	660
10.134.3.5	Clear	660
10.134.3.6	DetachEntity	660
10.134.3.7	DetachViewport	660
10.134.3.8	Finis	661
10.134.3.9	GenerateShaders	661
10.134.3.10	GetPSSMShadowCameraSetup	661
10.134.3.11	hit	661
10.134.3.12	RemoveScene	661
10.134.3.13	RemoveShadows	661
10.134.3.14	SetPerPixelLighting	662
10.134.3.15	UpdateShaders	662
10.135	gazebo::rendering::Scene Class Reference	662
10.135.1	Detailed Description	665
10.135.2	Constructor & Destructor Documentation	665
10.135.2.1	Scene	665
10.135.2.2	~Scene	665
10.135.3	Member Function Documentation	665
10.135.3.1	AddVisual	666
10.135.3.2	Clear	666
10.135.3.3	CloneVisual	666

10.135.3.4	CreateCamera	666
10.135.3.5	CreateDepthCamera	666
10.135.3.6	CreateGrid	667
10.135.3.7	CreateUserCamera	667
10.135.3.8	DrawLine	667
10.135.3.9	GetAmbientColor	667
10.135.3.10	GetBackgroundColor	668
10.135.3.11	GetCamera	668
10.135.3.12	GetCamera	668
10.135.3.13	GetCameraCount	668
10.135.3.14	GetFirstContact	669
10.135.3.15	GetGrid	669
10.135.3.16	GetGridCount	669
10.135.3.17	GetHeightBelowPoint	669
10.135.3.18	GetHeightmap	669
10.135.3.19	GetId	670
10.135.3.20	GetIdString	670
10.135.3.21	GetInitialized	670
10.135.3.22	GetLight	670
10.135.3.23	GetLight	670
10.135.3.24	GetLightCount	671
10.135.3.25	GetManager	671
10.135.3.26	GetModelVisualAt	671
10.135.3.27	GetName	671
10.135.3.28	GetSelectedVisual	671
10.135.3.29	GetShadowsEnabled	671
10.135.3.30	GetUserCamera	672
10.135.3.31	GetUserCameraCount	672
10.135.3.32	GetVisual	672
10.135.3.33	GetVisualAt	672
10.135.3.34	GetVisualAt	672
10.135.3.35	GetVisualBelow	673
10.135.3.36	GetVisualsBelowPoint	673
10.135.3.37	GetWorldVisual	673
10.135.3.38	Hit	673
10.135.3.39	Load	673
10.135.3.40	Load	674

10.135.3.41	PreRender	674
10.135.3.42	PrintSceneGraph	674
10.135.3.43	RemoveVisual	674
10.135.3.44	SelectVisual	674
10.135.3.45	SetAmbientColor	674
10.135.3.46	SetBackgroundColor	674
10.135.3.47	SetFog	675
10.135.3.48	SetGrid	675
10.135.3.49	SetShadowsEnabled	675
10.135.3.50	SetTransparent	675
10.135.3.51	SetVisible	675
10.135.3.52	ShowCollisions	676
10.135.3.53	ShowCOMs	676
10.135.3.54	ShowContacts	676
10.135.3.55	ShowJoints	676
10.135.3.56	SnapVisualToNearestBelow	676
10.135.3.57	StripSceneName	676
10.135.4	Member Data Documentation	677
10.135.4.1	skyx	677
10.136	gazebo::physics::ScrewJoint< T > Class Template Reference	677
10.136.1	Detailed Description	678
10.136.2	Constructor & Destructor Documentation	678
10.136.2.1	ScrewJoint	678
10.136.2.2	~ScrewJoint	678
10.136.3	Member Function Documentation	678
10.136.3.1	GetAnchor	678
10.136.3.2	GetAngleCount	679
10.136.3.3	GetThreadPitch	679
10.136.3.4	Load	679
10.136.3.5	SetAnchor	679
10.136.3.6	SetThreadPitch	679
10.136.4	Member Data Documentation	680
10.136.4.1	fakeAnchor	680
10.136.4.2	threadPitch	680
10.137	df::SDF Class Reference	680
10.137.1	Detailed Description	680
10.137.2	Constructor & Destructor Documentation	681

10.137.2.1	SDF	681
10.137.3	Member Function Documentation	681
10.137.3.1	PrintDescription	681
10.137.3.2	PrintDoc	681
10.137.3.3	PrintValues	681
10.137.3.4	PrintWiki	681
10.137.3.5	SetFromString	681
10.137.3.6	ToString	681
10.137.3.7	Write	681
10.137.4	Member Data Documentation	681
10.137.4.1	root	681
10.137.4.2	version	681
10.138	gazebo::rendering::SelectionObj Class Reference	681
10.138.1	Detailed Description	682
10.138.2	Constructor & Destructor Documentation	682
10.138.2.1	SelectionObj	682
10.138.2.2	~SelectionObj	682
10.138.3	Member Function Documentation	682
10.138.3.1	Attach	682
10.138.3.2	Clear	682
10.138.3.3	GetVisualName	682
10.138.3.4	Init	683
10.138.3.5	IsActive	683
10.138.3.6	SetActive	683
10.138.3.7	SetHighlight	683
10.139	gazebo::sensors::Sensor Class Reference	683
10.139.1	Detailed Description	686
10.139.2	Constructor & Destructor Documentation	686
10.139.2.1	Sensor	686
10.139.2.2	~Sensor	686
10.139.3	Member Function Documentation	686
10.139.3.1	ConnectUpdated	686
10.139.3.2	DisconnectUpdated	687
10.139.3.3	FillMsg	687
10.139.3.4	Finis	687
10.139.3.5	GetCategory	687
10.139.3.6	GetLastMeasurementTime	688

10.139.3.7	GetLastUpdateTime	688
10.139.3.8	GetName	688
10.139.3.9	GetParentName	688
10.139.3.10	GetPose	688
10.139.3.11	GetScopedName	689
10.139.3.12	GetTopic	689
10.139.3.13	GetType	689
10.139.3.14	GetUpdateRate	689
10.139.3.15	GetVisualize	689
10.139.3.16	GetWorldName	689
10.139.3.17	Init	690
10.139.3.18	IsActive	690
10.139.3.19	IsBad	690
10.139.3.20	IsLoad	690
10.139.3.21	ResetLastUpdateTime	690
10.139.3.22	SetActive	691
10.139.3.23	SetParent	691
10.139.3.24	SetUpdateRate	691
10.139.3.25	Update	691
10.139.3.26	UpdateImpl	691
10.139.4	Member Data Documentation	692
10.139.4.1	active	692
10.139.4.2	connections	692
10.139.4.3	lastMeasurementTime	692
10.139.4.4	lastUpdateTime	692
10.139.4.5	node	692
10.139.4.6	parentName	692
10.139.4.7	plugins	692
10.139.4.8	pose	692
10.139.4.9	poseSub	692
10.139.4.10	self	693
10.139.4.11	updatePeriod	693
10.139.4.12	world	693
10.140	SensorFactor Class Reference	693
10.140.1	Detailed Description	693
10.141	gazebo::sensors::SensorFactory Class Reference	693
10.141.1	Member Function Documentation	694

10.141.1.1	GetSensorTypes	694
10.141.1.2	NewSensor	694
10.141.1.3	RegisterAll	694
10.141.1.4	RegisterSensor	694
10.142	gazebo::sensors::SensorManager Class Reference	695
10.142.1	Detailed Description	696
10.142.2	Member Function Documentation	696
10.142.2.1	CreateSensor	696
10.142.2.2	Finis	696
10.142.2.3	GetSensor	696
10.142.2.4	GetSensors	697
10.142.2.5	GetSensorTypes	697
10.142.2.6	Init	697
10.142.2.7	RemoveSensor	697
10.142.2.8	RemoveSensors	697
10.142.2.9	ResetLastUpdateTimes	697
10.142.2.10	Run	697
10.142.2.11	RunThreads	698
10.142.2.12	SensorsInitialized	698
10.142.2.13	Stop	698
10.142.2.14	Update	698
10.143	gazebo::SensorPlugin Class Reference	698
10.143.1	Detailed Description	699
10.143.2	Constructor & Destructor Documentation	699
10.143.2.1	SensorPlugin	699
10.143.2.2	~SensorPlugin	699
10.143.3	Member Function Documentation	699
10.143.3.1	Init	699
10.143.3.2	Load	699
10.143.3.3	Reset	700
10.144	gazebo::Server Class Reference	700
10.144.1	Constructor & Destructor Documentation	700
10.144.1.1	Server	700
10.144.1.2	~Server	700
10.144.2	Member Function Documentation	700
10.144.2.1	Finis	700
10.144.2.2	GetInitialized	700

10.144.2.3	Init	700
10.144.2.4	LoadFile	701
10.144.2.5	LoadString	701
10.144.2.6	ParseArgs	701
10.144.2.7	PrintUsage	701
10.144.2.8	Run	701
10.144.2.9	SetParams	701
10.144.2.10	Stop	701
10.144.3	Member Data Documentation	701
10.144.3.1	systemPluginsArgc	701
10.144.3.2	systemPluginsArgv	701
10.145	Gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg Class Reference	701
10.145.1	Detailed Description	702
10.145.2	Member Function Documentation	703
10.145.2.1	defaultVpParams	703
10.145.2.2	generateFragmentProgram	703
10.145.2.3	generateVertexProgram	703
10.145.2.4	generateVertexProgramSource	703
10.145.2.5	generateVpDynamicShadows	703
10.145.2.6	generateVpDynamicShadowsParams	703
10.145.2.7	generateVpFooter	703
10.145.2.8	generateVpHeader	703
10.146	Gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL Class Reference	703
10.146.1	Detailed Description	705
10.146.2	Member Function Documentation	705
10.146.2.1	defaultVpParams	705
10.146.2.2	generateFpDynamicShadows	705
10.146.2.3	generateFpDynamicShadowsHelpers	705
10.146.2.4	generateFpDynamicShadowsParams	705
10.146.2.5	generateFpFooter	705
10.146.2.6	generateFpHeader	705
10.146.2.7	generateFpLayer	705
10.146.2.8	generateFragmentProgram	705
10.146.2.9	generateFragmentProgramSource	706
10.146.2.10	generateVertexProgram	706
10.146.2.11	generateVertexProgramSource	706
10.146.2.12	generateVpDynamicShadows	706

10.146.2.1	GenerateVpDynamicShadowsParams	706
10.146.2.1	GenerateVpFooter	706
10.146.2.1	GenerateVpHeader	706
10.146.2.1	UpdateParams	706
10.146.2.1	UpdateVpParams	706
10.147	gazebo::physics::Shape Class Reference	706
10.147.1	Detailed Description	707
10.147.2	Constructor & Destructor Documentation	707
10.147.2.1	Shape	708
10.147.2.2	~Shape	708
10.147.3	Member Function Documentation	708
10.147.3.1	FillMsg	708
10.147.3.2	Init	708
10.147.3.3	ProcessMsg	708
10.147.4	Member Data Documentation	709
10.147.4.1	collisionParent	709
10.148	gazebo::sensors::SimTimeEvent Class Reference	709
10.148.1	Detailed Description	709
10.148.2	Member Data Documentation	709
10.148.2.1	condition	709
10.148.2.2	time	709
10.149	gazebo::sensors::SimTimeEventHandler Class Reference	709
10.149.1	Detailed Description	710
10.149.2	Constructor & Destructor Documentation	710
10.149.2.1	SimTimeEventHandler	710
10.149.2.2	~SimTimeEventHandler	710
10.149.3	Member Function Documentation	710
10.149.3.1	AddRelativeEvent	711
10.150	SingletonT< T > Class Template Reference	711
10.150.1	Detailed Description	713
10.150.2	Constructor & Destructor Documentation	713
10.150.2.1	SingletonT	713
10.150.2.2	~SingletonT	713
10.150.3	Member Function Documentation	713
10.150.3.1	Instance	713
10.151	gazebo::common::Skeleton Class Reference	713
10.151.1	Detailed Description	715

10.151.2	Constructor & Destructor Documentation	715
10.151.2.1	Skeleton	715
10.151.2.2	Skeleton	715
10.151.2.3	~Skeleton	715
10.151.3	Member Function Documentation	715
10.151.3.1	AddAnimation	715
10.151.3.2	AddVertNodeWeight	716
10.151.3.3	BuildNodeMap	716
10.151.3.4	GetAnimation	716
10.151.3.5	GetBindShapeTransform	716
10.151.3.6	GetNodeByHandle	716
10.151.3.7	GetNodeById	716
10.151.3.8	GetNodeByName	717
10.151.3.9	GetNodes	717
10.151.3.10	GetNumAnimations	717
10.151.3.11	GetNumJoints	717
10.151.3.12	GetNumNodes	717
10.151.3.13	GetNumVertNodeWeights	718
10.151.3.14	GetRootNode	718
10.151.3.15	GetVertNodeWeight	718
10.151.3.16	PrintTransforms	718
10.151.3.17	Scale	718
10.151.3.18	SetBindShapeTransform	718
10.151.3.19	SetNumVertAttached	719
10.151.3.20	SetRootNode	719
10.151.4	Member Data Documentation	719
10.151.4.1	anims	719
10.151.4.2	bindShapeTransform	719
10.151.4.3	nodes	719
10.151.4.4	rawNW	719
10.151.4.5	root	719
10.152	gazebo::common::SkeletonAnimation Class Reference	720
10.152.1	Detailed Description	721
10.152.2	Constructor & Destructor Documentation	721
10.152.2.1	SkeletonAnimation	721
10.152.2.2	~SkeletonAnimation	721
10.152.3	Member Function Documentation	721

10.152.3.1	AddKeyFrame	721
10.152.3.2	AddKeyFrame	721
10.152.3.3	GetLength	721
10.152.3.4	GetName	722
10.152.3.5	GetNodeCount	722
10.152.3.6	GetNodePoseAt	722
10.152.3.7	GetPoseAt	722
10.152.3.8	GetPoseAtX	723
10.152.3.9	HasNode	723
10.152.3.10	Scale	723
10.152.3.11	SetName	723
10.152.4	Member Data Documentation	723
10.152.4.1	animations	723
10.152.4.2	length	724
10.152.4.3	name	724
10.153	Gazebo::common::SkeletonNode Class Reference	724
10.153.1	Detailed Description	726
10.153.2	Member Enumeration Documentation	726
10.153.2.1	SkeletonNodeType	726
10.153.3	Constructor & Destructor Documentation	726
10.153.3.1	SkeletonNode	726
10.153.3.2	SkeletonNode	727
10.153.3.3	~SkeletonNode	727
10.153.4	Member Function Documentation	727
10.153.4.1	AddChild	727
10.153.4.2	AddRawTransform	727
10.153.4.3	GetChild	727
10.153.4.4	GetChildById	727
10.153.4.5	GetChildByName	728
10.153.4.6	GetChildCount	728
10.153.4.7	GetHandle	728
10.153.4.8	GetId	728
10.153.4.9	GetInverseBindTransform	728
10.153.4.10	GetModelTransform	729
10.153.4.11	GetName	729
10.153.4.12	GetNumRawTrans	729
10.153.4.13	GetParent	729

10.153.4.10	GetRawTransform	729
10.153.4.11	GetRawTransforms	729
10.153.4.12	GetTransform	730
10.153.4.13	GetTransforms	730
10.153.4.14	Joint	730
10.153.4.15	RootNode	730
10.153.4.16	Reset	730
10.153.4.17	SetHandle	730
10.153.4.18	SetId	731
10.153.4.19	SetInitialTransform	731
10.153.4.20	SetInverseBindTransform	731
10.153.4.21	SetModelTransform	731
10.153.4.22	SetName	731
10.153.4.23	SetParent	731
10.153.4.24	SetTransform	732
10.153.4.25	SetType	732
10.153.4.26	UpdateChildrenTransforms	732
10.153.5	Member Data Documentation	732
10.153.5.1	children	732
10.153.5.2	handle	732
10.153.5.3	id	732
10.153.5.4	initialTransform	732
10.153.5.5	invBindTransform	732
10.153.5.6	modelTransform	732
10.153.5.7	name	733
10.153.5.8	parent	733
10.153.5.9	rawTransforms	733
10.153.5.10	transform	733
10.153.5.11	type	733
10.154	gazebo::physics::SliderJoint< T > Class Template Reference	733
10.154.1	Detailed Description	734
10.154.2	Constructor & Destructor Documentation	734
10.154.2.1	SliderJoint	734
10.154.2.2	~SliderJoint	734
10.154.3	Member Function Documentation	734
10.154.3.1	GetAnchor	734
10.154.3.2	GetAngleCount	735

10.154.3.3	Load	735
10.154.3.4	SetAnchor	735
10.154.4	Member Data Documentation	735
10.154.4.1	fakeAnchor	735
10.155	gazebo::rendering::GzTerrainMatGen::SM2Profile Class Reference	735
10.155.1	Detailed Description	736
10.155.2	Constructor & Destructor Documentation	737
10.155.2.1	SM2Profile	737
10.155.2.2	~SM2Profile	737
10.155.3	Member Function Documentation	737
10.155.3.1	addTechnique	737
10.155.3.2	generate	737
10.155.3.3	generateForCompositeMap	737
10.155.3.4	updateParams	737
10.155.3.5	updateParamsForCompositeMap	737
10.156	gazebo::physics::SphereShape Class Reference	737
10.156.1	Detailed Description	738
10.156.2	Constructor & Destructor Documentation	739
10.156.2.1	SphereShape	739
10.156.2.2	~SphereShape	739
10.156.3	Member Function Documentation	739
10.156.3.1	FillMsg	739
10.156.3.2	GetRadius	739
10.156.3.3	Init	739
10.156.3.4	ProcessMsg	739
10.156.3.5	SetRadius	740
10.157	gazebo::math::Spline Class Reference	740
10.157.1	Detailed Description	741
10.157.2	Constructor & Destructor Documentation	741
10.157.2.1	Spline	741
10.157.2.2	~Spline	741
10.157.3	Member Function Documentation	741
10.157.3.1	AddPoint	741
10.157.3.2	Clear	741
10.157.3.3	GetPoint	741
10.157.3.4	GetPointCount	742
10.157.3.5	GetTangent	742

10.157.3.6	GetTension	742
10.157.3.7	Interpolate	742
10.157.3.8	Interpolate	742
10.157.3.9	RecalcTangents	743
10.157.3.10	SetAutoCalculate	743
10.157.3.11	SetTension	743
10.157.3.12	UpdatePoint	743
10.157.4	Member Data Documentation	744
10.157.4.1	autoCalc	744
10.157.4.2	coeffs	744
10.157.4.3	points	744
10.157.4.4	tangents	744
10.157.4.5	tension	744
10.158	gazebo::physics::State Class Reference	744
10.158.1	Detailed Description	746
10.158.2	Constructor & Destructor Documentation	746
10.158.2.1	State	746
10.158.2.2	State	746
10.158.2.3	~State	746
10.158.3	Member Function Documentation	746
10.158.3.1	GetName	746
10.158.3.2	GetRealTime	747
10.158.3.3	GetSimTime	747
10.158.3.4	GetWallTime	747
10.158.3.5	Load	747
10.158.3.6	operator-	747
10.158.3.7	operator=	748
10.158.3.8	SetName	748
10.158.4	Member Data Documentation	748
10.158.4.1	name	748
10.158.4.2	realTime	748
10.158.4.3	simTime	748
10.158.4.4	wallTime	748
10.159	gazebo::common::STLLoader Class Reference	748
10.159.1	Detailed Description	749
10.159.2	Constructor & Destructor Documentation	749
10.159.2.1	STLLoader	749

10.159.2.2~STLLoader	749
10.159.3 Member Function Documentation	749
10.159.3.1Load	749
10.160 gazebo::common::SubMesh Class Reference	750
10.160.1 Detailed Description	752
10.160.2 Member Enumeration Documentation	752
10.160.2.1PrimitiveType	752
10.160.3 Constructor & Destructor Documentation	752
10.160.3.1SubMesh	752
10.160.3.2~SubMesh	752
10.160.4 Member Function Documentation	752
10.160.4.1AddIndex	752
10.160.4.2AddNodeAssignment	753
10.160.4.3AddNormal	753
10.160.4.4AddNormal	753
10.160.4.5AddTexCoord	753
10.160.4.6AddVertex	753
10.160.4.7AddVertex	754
10.160.4.8CopyNormals	754
10.160.4.9CopyVertices	754
10.160.4.10FillArrays	754
10.160.4.11GenSphericalTexCoord	754
10.160.4.12GetIndex	754
10.160.4.13GetIndexCount	755
10.160.4.14GetMaterialIndex	755
10.160.4.15GetMax	755
10.160.4.16GetMaxIndex	755
10.160.4.17GetMin	755
10.160.4.18GetNodeAssignment	755
10.160.4.19GetNodeAssignmentsCount	755
10.160.4.20GetNormal	755
10.160.4.21GetNormalCount	756
10.160.4.22GetPrimitiveType	756
10.160.4.23GetTexCoord	756
10.160.4.24GetTexCoordCount	756
10.160.4.25GetVertex	756
10.160.4.26GetVertexCount	757

10.160.4.27	SetVertexIndex	757
10.160.4.28	IsVertex	757
10.160.4.29	RecalculateNormals	757
10.160.4.30	Scale	757
10.160.4.31	SetIndexCount	757
10.160.4.32	SetMaterialIndex	757
10.160.4.33	SetNormal	758
10.160.4.34	SetNormalCount	758
10.160.4.35	SetPrimitiveType	758
10.160.4.36	SetScale	758
10.160.4.37	SetSubMeshCenter	758
10.160.4.38	SetTexCoord	758
10.160.4.39	SetTexCoordCount	759
10.160.4.40	SetVertex	759
10.160.4.41	SetVertexCount	759
10.161	gazebo::transport::SubscribeOptions Class Reference	759
10.161.1	Detailed Description	760
10.161.2	Constructor & Destructor Documentation	760
10.161.2.1	SubscribeOptions	760
10.161.3	Member Function Documentation	760
10.161.3.1	GetLatching	760
10.161.3.2	GetMsgType	760
10.161.3.3	GetNode	760
10.161.3.4	GetTopic	761
10.161.3.5	Init	761
10.161.3.6	Init	761
10.162	gazebo::transport::Subscriber Class Reference	761
10.162.1	Detailed Description	762
10.162.2	Constructor & Destructor Documentation	762
10.162.2.1	Subscriber	762
10.162.2.2	~Subscriber	762
10.162.3	Member Function Documentation	762
10.162.3.1	GetCallbackId	762
10.162.3.2	GetTopic	762
10.162.3.3	SetCallbackId	762
10.162.3.4	Unsubscribe	762
10.163	gazebo::transport::SubscriptionTransport Class Reference	763

10.163.1	Detailed Description	763
10.163.2	Constructor & Destructor Documentation	764
10.163.2.1	SubscriptionTransport	764
10.163.2.2	~SubscriptionTransport	764
10.163.3	Member Function Documentation	764
10.163.3.1	GetConnection	764
10.163.3.2	HandleData	764
10.163.3.3	init	764
10.163.3.4	isLocal	764
10.164	gazebo::physics::SurfaceParams Class Reference	765
10.164.1	Detailed Description	766
10.164.2	Constructor & Destructor Documentation	766
10.164.2.1	SurfaceParams	766
10.164.2.2	~SurfaceParams	766
10.164.3	Member Function Documentation	766
10.164.3.1	FillMsg	766
10.164.3.2	Load	766
10.164.3.3	ProcessMsg	766
10.164.4	Member Data Documentation	766
10.164.4.1	bounce	767
10.164.4.2	bounceThreshold	767
10.164.4.3	cfm	767
10.164.4.4	erp	767
10.164.4.5	dir1	767
10.164.4.6	kd	767
10.164.4.7	kp	768
10.164.4.8	maxVel	768
10.164.4.9	minDepth	768
10.164.4.10	u1	768
10.164.4.11	u2	768
10.164.4.12	sip1	769
10.164.4.13	sip2	769
10.165	gazebo::common::SystemPaths Class Reference	769
10.165.1	Detailed Description	771
10.165.2	Member Function Documentation	771
10.165.2.1	AddGazeboPaths	771
10.165.2.2	AddModelPaths	771

10.165.2.3	AddOgrePaths	771
10.165.2.4	AddPluginPaths	771
10.165.2.5	AddSearchPathSuffix	771
10.165.2.6	ClearGazeboPaths	772
10.165.2.7	ClearModelPaths	772
10.165.2.8	ClearOgrePaths	772
10.165.2.9	ClearPluginPaths	772
10.165.2.10	FindFile	772
10.165.2.11	FindFileURI	772
10.165.2.12	GetGazeboPaths	773
10.165.2.13	GetLogPath	773
10.165.2.14	GetModelPaths	773
10.165.2.15	GetOgrePaths	773
10.165.2.16	GetPluginPaths	773
10.165.2.17	GetWorldPathExtension	773
10.165.3	Member Data Documentation	774
10.165.3.1	gazeboPathsFromEnv	774
10.165.3.2	modelPathsFromEnv	774
10.165.3.3	ogrePathsFromEnv	774
10.165.3.4	pluginPathsFromEnv	774
10.166	gazebo::SystemPlugin Class Reference	774
10.166.1	Detailed Description	775
10.166.2	Constructor & Destructor Documentation	775
10.166.2.1	SystemPlugin	775
10.166.2.2	~SystemPlugin	775
10.166.3	Member Function Documentation	775
10.166.3.1	Init	775
10.166.3.2	Load	775
10.166.3.3	Reset	776
10.167	gazebo::common::Time Class Reference	776
10.167.1	Detailed Description	779
10.167.2	Constructor & Destructor Documentation	780
10.167.2.1	Time	780
10.167.2.2	Time	780
10.167.2.3	Time	780
10.167.2.4	Time	780
10.167.2.5	Time	780

10.167.2.6	Time	780
10.167.2.7	~Time	781
10.167.3	Member Function Documentation	781
10.167.3.1	Double	781
10.167.3.2	Float	781
10.167.3.3	GetWallTime	781
10.167.3.4	GetWallTimeAsISOString	781
10.167.3.5	MicToNano	781
10.167.3.6	MillToNano	782
10.167.3.7	MSleep	782
10.167.3.8	NSleep	782
10.167.3.9	NSleep	782
10.167.3.10	operator!=	783
10.167.3.11	operator!=	783
10.167.3.12	operator!=	783
10.167.3.13	operator!=	783
10.167.3.14	operator*	784
10.167.3.15	operator*	784
10.167.3.16	operator*	784
10.167.3.17	operator*==	784
10.167.3.18	operator*==	785
10.167.3.19	operator*==	785
10.167.3.20	operator+	785
10.167.3.21	operator+	785
10.167.3.22	operator+	786
10.167.3.23	operator+=	786
10.167.3.24	operator+=	786
10.167.3.25	operator+=	786
10.167.3.26	operator-	787
10.167.3.27	operator-	787
10.167.3.28	operator-	787
10.167.3.29	operator--	787
10.167.3.30	operator--	788
10.167.3.31	operator--	788
10.167.3.32	operator/	788
10.167.3.33	operator/	788
10.167.3.34	operator/	789

10.167.3.35	operator/=	789
10.167.3.36	operator/=	789
10.167.3.37	operator/=	789
10.167.3.38	operator<	790
10.167.3.39	operator<	790
10.167.3.40	operator<	790
10.167.3.41	operator<	790
10.167.3.42	operator<=	791
10.167.3.43	operator<=	791
10.167.3.44	operator<=	791
10.167.3.45	operator<=	791
10.167.3.46	operator=	792
10.167.3.47	operator=	792
10.167.3.48	operator=	792
10.167.3.49	operator==	792
10.167.3.50	operator==	793
10.167.3.51	operator==	793
10.167.3.52	operator==	793
10.167.3.53	operator>	793
10.167.3.54	operator>	794
10.167.3.55	operator>	794
10.167.3.56	operator>	794
10.167.3.57	operator>=	794
10.167.3.58	operator>=	795
10.167.3.59	operator>=	795
10.167.3.60	operator>=	795
10.167.3.61	SecToNano	795
10.167.3.62	Set	796
10.167.3.63	Set	796
10.167.3.64	SetToWallTime	796
10.167.3.65	Sleep	796
10.167.4	Friends And Related Function Documentation	796
10.167.4.1	operator<<	796
10.167.4.2	operator>>	797
10.167.5	Member Data Documentation	797
10.167.5.1	Insec	797
10.167.5.2	Sec	797

10.167.5.3Zero	797
10.168.0gazebo::common::Timer Class Reference	797
10.168.1Detailed Description	798
10.168.2Constructor & Destructor Documentation	798
10.168.2.1Timer	798
10.168.2.2~Timer	799
10.168.3Member Function Documentation	799
10.168.3.1GetElapsed	799
10.168.3.2GetRunning	799
10.168.3.3Start	799
10.168.3.4Stop	799
10.168.4Friends And Related Function Documentation	799
10.168.4.1operator<<	799
10.169.0gazebo::transport::TopicManager Class Reference	799
10.169.1Detailed Description	801
10.169.2Member Typedef Documentation	801
10.169.2.1SubNodeMap	801
10.169.3Member Function Documentation	801
10.169.3.1AddNode	801
10.169.3.2Advertise	802
10.169.3.3ClearBuffers	802
10.169.3.4ConnectPubToSub	802
10.169.3.5ConnectSubscribers	802
10.169.3.6ConnectSubToPub	802
10.169.3.7DisconnectPubFromSub	803
10.169.3.8DisconnectSubFromPub	803
10.169.3.9FindPublication	803
10.169.3.10Ini	803
10.169.3.11GetAdvertisedTopics	803
10.169.3.12GetTopicNamespaces	804
10.169.3.13Init	804
10.169.3.14Advertised	804
10.169.3.15PauseIncoming	804
10.169.3.16ProcessNodes	804
10.169.3.17Publish	804
10.169.3.18RegisterTopicNamespace	805
10.169.3.19RemoveNode	805

10.169.3.20	Subscribe	805
10.169.3.21	Inadvertise	805
10.169.3.22	Unsubscribe	805
10.169.3.23	UpdatePublications	806
10.170	gazebo::physics::TrajectoryInfo Struct Reference	806
10.170.1	Member Data Documentation	806
10.170.1.1	duration	806
10.170.1.2	endTime	806
10.170.1.3	d	806
10.170.1.4	startTime	806
10.170.1.5	translated	806
10.170.1.6	type	807
10.171	gazebo::physics::TrimeshShape Class Reference	807
10.171.1	Detailed Description	808
10.171.2	Constructor & Destructor Documentation	808
10.171.2.1	TrimeshShape	808
10.171.2.2	~TrimeshShape	808
10.171.3	Member Function Documentation	808
10.171.3.1	FillMsg	808
10.171.3.2	GetFilename	809
10.171.3.3	GetSize	809
10.171.3.4	Init	809
10.171.3.5	ProcessMsg	809
10.171.3.6	SetFilename	809
10.171.3.7	SetScale	809
10.171.3.8	Update	810
10.171.4	Member Data Documentation	810
10.171.4.1	mesh	810
10.172	gazebo::physics::UniversalJoint< T > Class Template Reference	810
10.172.1	Detailed Description	811
10.172.2	Constructor & Destructor Documentation	811
10.172.2.1	UniversalJoint	811
10.172.2.2	~UniversalJoint	811
10.172.3	Member Function Documentation	811
10.172.3.1	GetAngleCount	811
10.172.3.2	Load	811
10.173	gazebo::common::UpdateInfo Class Reference	811

10.173.1	Detailed Description	812
10.173.2	Member Data Documentation	812
10.173.2.1	realTime	812
10.173.2.2	simTime	812
10.173.2.3	worldName	812
10.174	urdf2gazebo::URDF2Gazebo Class Reference	812
10.174.1	Constructor & Destructor Documentation	813
10.174.1.1	URDF2Gazebo	813
10.174.1.2	~URDF2Gazebo	813
10.174.2	Member Function Documentation	813
10.174.2.1	InitModelDoc	813
10.174.2.2	InitModelFile	813
10.174.2.3	InitModelString	813
10.175	gazebo::rendering::UserCamera Class Reference	814
10.175.1	Detailed Description	816
10.175.2	Constructor & Destructor Documentation	816
10.175.2.1	UserCamera	816
10.175.2.2	~UserCamera	816
10.175.3	Member Function Documentation	816
10.175.3.1	AnimationComplete	816
10.175.3.2	AttachToVisualImpl	816
10.175.3.3	EnableViewController	817
10.175.3.4	Finis	817
10.175.3.5	GetAvgFPS	817
10.175.3.6	GetGUIOverlay	817
10.175.3.7	GetImageHeight	817
10.175.3.8	GetImageWidth	818
10.175.3.9	GetTriangleCount	818
10.175.3.10	GetViewControllerTypeString	818
10.175.3.11	GetVisual	818
10.175.3.12	GetVisual	818
10.175.3.13	HandleKeyPressEvent	819
10.175.3.14	HandleKeyReleaseEvent	819
10.175.3.15	HandleMouseEvent	819
10.175.3.16	Init	819
10.175.3.17	Load	819
10.175.3.18	Load	819

10.175.3.1	MoveToPosition	820
10.175.3.2	MoveToVisual	820
10.175.3.2	MoveToVisual	820
10.175.3.2	PostRender	820
10.175.3.2	Resize	820
10.175.3.2	SetFocalPoint	820
10.175.3.2	SetRenderTarget	821
10.175.3.2	SetViewController	821
10.175.3.2	SetViewController	821
10.175.3.2	SetViewportDimensions	821
10.175.3.2	SetWorldPose	821
10.175.3.3	TrackVisualImpl	822
10.175.3.3	Update	822
10.176	gazebo::math::Vector2d Class Reference	822
10.176.1	Detailed Description	824
10.176.2	Constructor & Destructor Documentation	824
10.176.2.1	Vector2d	824
10.176.2.2	Vector2d	824
10.176.2.3	Vector2d	824
10.176.2.4	~Vector2d	824
10.176.3	Member Function Documentation	824
10.176.3.1	Cross	824
10.176.3.2	Distance	825
10.176.3.3	IsFinite	825
10.176.3.4	Normalize	825
10.176.3.5	operator!=	825
10.176.3.6	operator*	825
10.176.3.7	operator*	826
10.176.3.8	operator*= operator*=	826
10.176.3.9	operator*= operator*=	826
10.176.3.10	operator+	826
10.176.3.11	operator+=	827
10.176.3.12	operator-	827
10.176.3.13	operator-- operator--	827
10.176.3.14	operator/	827
10.176.3.15	operator/ operator/	828
10.176.3.16	operator/=	828

10.176.3.17	operator/=	828
10.176.3.18	operator=	828
10.176.3.19	operator=	829
10.176.3.20	operator==	829
10.176.3.21	operator[]	829
10.176.3.22	set	829
10.176.4	Friends And Related Function Documentation	830
10.176.4.1	operator<<	830
10.176.4.2	operator>>	830
10.176.5	Member Data Documentation	830
10.176.5.1x		830
10.176.5.2y		830
10.177	Gazebo::math::Vector2i Class Reference	830
10.177.1	Detailed Description	832
10.177.2	Constructor & Destructor Documentation	832
10.177.2.1	Vector2i	832
10.177.2.2	Vector2i	832
10.177.2.3	Vector2i	832
10.177.2.4	~Vector2i	833
10.177.3	Member Function Documentation	833
10.177.3.1	Cross	833
10.177.3.2	Distance	833
10.177.3.3	isFinite	833
10.177.3.4	Normalize	833
10.177.3.5	operator!=	833
10.177.3.6	operator*	834
10.177.3.7	operator*	834
10.177.3.8	operator*= operator*=	834
10.177.3.9	operator*= operator*=	835
10.177.3.10	operator+	835
10.177.3.11	operator+=	835
10.177.3.12	operator-	835
10.177.3.13	operator-- operator--	836
10.177.3.14	operator/	836
10.177.3.15	operator/	836
10.177.3.16	operator/=	837
10.177.3.17	operator/=	837

10.177.3.1	operator=	837
10.177.3.1	operator=	837
10.177.3.2	operator==	838
10.177.3.2	operator[]	838
10.177.3.2	Set	838
10.177.4	Friends And Related Function Documentation	838
10.177.4.1	operator<<	838
10.177.4.2	operator>>	839
10.177.5	Member Data Documentation	839
10.177.5.1	x	839
10.177.5.2	y	839
10.178	Gazebo::math::Vector3 Class Reference	839
10.178.1	Detailed Description	842
10.178.2	Constructor & Destructor Documentation	842
10.178.2.1	Vector3	842
10.178.2.2	Vector3	842
10.178.2.3	Vector3	842
10.178.2.4	~Vector3	843
10.178.3	Member Function Documentation	843
10.178.3.1	Correct	843
10.178.3.2	Cross	843
10.178.3.3	Distance	843
10.178.3.4	Distance	843
10.178.3.5	Dot	844
10.178.3.6	Equal	844
10.178.3.7	GetAbs	844
10.178.3.8	GetDistToLine	844
10.178.3.9	GetLength	845
10.178.3.10	GetMax	845
10.178.3.11	GetMin	845
10.178.3.12	GetNormal	845
10.178.3.13	GetPerpendicular	845
10.178.3.14	GetRounded	845
10.178.3.15	GetSquaredLength	846
10.178.3.16	GetSum	846
10.178.3.17	Finite	846
10.178.3.18	Normalize	846

10.178.3.1	operator!=	846
10.178.3.2	operator*	846
10.178.3.2	operator*	847
10.178.3.2	operator*==	847
10.178.3.2	operator*==	847
10.178.3.2	operator+	847
10.178.3.2	operator+=	848
10.178.3.2	operator-	848
10.178.3.2	operator-	848
10.178.3.2	operator-=	848
10.178.3.2	operator/	849
10.178.3.3	operator/	849
10.178.3.3	operator/=	849
10.178.3.3	operator/=	849
10.178.3.3	operator=	850
10.178.3.3	operator=	850
10.178.3.3	operator==	850
10.178.3.3	operator[]	850
10.178.3.3	bound	850
10.178.3.3	bound	851
10.178.3.3	set	851
10.178.3.4	setToMax	851
10.178.3.4	setToMin	851
10.178.4	Friends And Related Function Documentation	851
10.178.4.1	operator*	851
10.178.4.2	operator<<	852
10.178.4.3	operator>>	852
10.178.5	Member Data Documentation	852
10.178.5.1	x	852
10.178.5.2	y	852
10.178.5.3	z	852
10.178.5.4	Zero	852
10.179	Gazebo::math::Vector4 Class Reference	853
10.179.1	Detailed Description	854
10.179.2	Constructor & Destructor Documentation	855
10.179.2.1	Vector4	855
10.179.2.2	Vector4	855

10.179.2.3	Vector4	855
10.179.2.4	~Vector4	855
10.179.3	Member Function Documentation	855
10.179.3.1	Distance	855
10.179.3.2	GetLength	855
10.179.3.3	GetSquaredLength	856
10.179.3.4	IsFinite	856
10.179.3.5	Normalize	856
10.179.3.6	operator!=	856
10.179.3.7	operator*	856
10.179.3.8	operator*	857
10.179.3.9	operator*	857
10.179.3.10	operator*=	857
10.179.3.11	operator*=	857
10.179.3.12	operator+	858
10.179.3.13	operator+=	858
10.179.3.14	operator-	858
10.179.3.15	operator-=	858
10.179.3.16	operator/	859
10.179.3.17	operator/	859
10.179.3.18	operator/=	859
10.179.3.19	operator/=	860
10.179.3.20	operator=	860
10.179.3.21	operator=	860
10.179.3.22	operator==	860
10.179.3.23	operator[]	861
10.179.3.24	set	861
10.179.4	Friends And Related Function Documentation	861
10.179.4.1	operator<<	861
10.179.4.2	operator>>	861
10.179.5	Member Data Documentation	861
10.179.5.1	w	862
10.179.5.2	x	862
10.179.5.3	y	862
10.179.5.4	z	862
10.180	Gazebo::common::Video Class Reference	862
10.180.1	Detailed Description	862

10.180.2	Constructor & Destructor Documentation	863
10.180.2.1	Video	863
10.180.2.2	~Video	863
10.180.3	Member Function Documentation	863
10.180.3.1	GetHeight	863
10.180.3.2	GetNextFrame	863
10.180.3.3	GetWidth	863
10.180.3.4	Load	863
10.180	gazebo::rendering::VideoVisual Class Reference	864
10.181.1	Detailed Description	864
10.181.2	Constructor & Destructor Documentation	865
10.181.2.1	VideoVisual	865
10.181.2.2	~VideoVisual	865
10.180	gazebo::rendering::ViewController Class Reference	865
10.182.1	Detailed Description	866
10.182.2	Constructor & Destructor Documentation	866
10.182.2.1	ViewController	866
10.182.2.2	~ViewController	866
10.182.3	Member Function Documentation	866
10.182.3.1	GetTypeString	867
10.182.3.2	HandleKeyPressEvent	867
10.182.3.3	HandleKeyReleaseEvent	867
10.182.3.4	HandleMouseEvent	867
10.182.3.5	Init	867
10.182.3.6	Init	868
10.182.3.7	setEnabled	868
10.182.3.8	Update	868
10.182.4	Member Data Documentation	868
10.182.4.1	camera	868
10.182.4.2	enabled	868
10.182.4.3	typeString	868
10.180	gazebo::rendering::Visual Class Reference	868
10.183.1	Detailed Description	873
10.183.2	Constructor & Destructor Documentation	873
10.183.2.1	Visual	873
10.183.2.2	Visual	874
10.183.2.3	~Visual	874

10.183.3 Member Function Documentation	874
10.183.3.1 AttachAxes	874
10.183.3.2 AttachLineVertex	874
10.183.3.3 AttachMesh	874
10.183.3.4 AttachObject	874
10.183.3.5 AttachVisual	875
10.183.3.6 ClearParent	875
10.183.3.7 Clone	875
10.183.3.8 CreateDynamicLine	875
10.183.3.9 DeleteDynamicLine	875
10.183.3.10 DetachObjects	875
10.183.3.11 DetachVisual	876
10.183.3.12 DetachVisual	876
10.183.3.13 DisableTrackVisual	876
10.183.3.14 EnableTrackVisual	876
10.183.3.15 End	876
10.183.3.16 GetAttachedObjectCount	876
10.183.3.17 GetBoundingBox	876
10.183.3.18 GetChild	877
10.183.3.19 GetChildCount	877
10.183.3.20 GetMaterialName	877
10.183.3.21 GetMeshName	877
10.183.3.22 GetName	877
10.183.3.23 GetNormalMap	877
10.183.3.24 GetParent	878
10.183.3.25 GetPose	878
10.183.3.26 GetPosition	878
10.183.3.27 GetRootVisual	878
10.183.3.28 GetRotation	878
10.183.3.29 GetScale	878
10.183.3.30 GetScene	879
10.183.3.31 GetSceneNode	879
10.183.3.32 GetShaderType	879
10.183.3.33 GetTransparency	879
10.183.3.34 GetVisibilityFlags	879
10.183.3.35 GetVisible	880
10.183.3.36 GetWorldPose	880

10.183.3.37	HasAttachedObject	880
10.183.3.38	Hit	880
10.183.3.39	InsertMesh	880
10.183.3.40	InsertMesh	880
10.183.3.41	IsPlane	880
10.183.3.42	IsStatic	881
10.183.3.43	Load	881
10.183.3.44	Load	881
10.183.3.45	LoadFromMsg	881
10.183.3.46	LoadPlugin	881
10.183.3.47	MakeStatic	881
10.183.3.48	MoveToPosition	882
10.183.3.49	MoveToPositions	882
10.183.3.50	RemovePlugin	882
10.183.3.51	SetAmbient	882
10.183.3.52	SetCastShadows	882
10.183.3.53	SetDiffuse	882
10.183.3.54	SetEmissive	883
10.183.3.55	SetHighlighted	883
10.183.3.56	SetMaterial	883
10.183.3.57	SetName	883
10.183.3.58	SetNormalMap	883
10.183.3.59	SetPose	884
10.183.3.60	SetPosition	884
10.183.3.61	SetRibbonTrail	884
10.183.3.62	SetRotation	884
10.183.3.63	SetScale	884
10.183.3.64	SetScene	884
10.183.3.65	SetShaderType	885
10.183.3.66	SetSkeletonPose	885
10.183.3.67	SetSpecular	885
10.183.3.68	SetTransparency	885
10.183.3.69	SetVisibilityFlags	885
10.183.3.70	SetVisible	886
10.183.3.71	SetWorldPose	886
10.183.3.72	SetWorldPosition	886
10.183.3.73	SetWorldRotation	886

10.183.3.78	ShowBoundingBox	886
10.183.3.79	ShowCollision	886
10.183.3.76	ShowCOM	887
10.183.3.75	ShowJoints	887
10.183.3.78	ShowSkeleton	887
10.183.3.70	ToggleVisible	887
10.183.3.80	Update	887
10.183.3.81	UpdateFromMsg	887
10.183.4	Member Data Documentation	887
10.183.4.1	parent	887
10.183.4.2	scene	888
10.183.4.3	sceneNode	888
10.184	gazebo::VisualPlugin Class Reference	888
10.184.1	Detailed Description	888
10.184.2	Constructor & Destructor Documentation	889
10.184.2.1	VisualPlugin	889
10.184.3	Member Function Documentation	889
10.184.3.1	Init	889
10.184.3.2	Load	889
10.184.3.3	Reset	889
10.185	gazebo::rendering::WindowManager Class Reference	889
10.185.1	Detailed Description	890
10.185.2	Member Function Documentation	890
10.185.2.1	CreateWindow	890
10.185.2.2	Finis	891
10.185.2.3	GetAvgFPS	891
10.185.2.4	GetTriangleCount	891
10.185.2.5	GetWindow	891
10.185.2.6	Moved	891
10.185.2.7	Resize	892
10.185.2.8	SetCamera	892
10.186	gazebo::rendering::WireBox Class Reference	892
10.186.1	Detailed Description	892
10.186.2	Constructor & Destructor Documentation	893
10.186.2.1	WireBox	893
10.186.2.2	~WireBox	893
10.186.3	Member Function Documentation	893

10.186.3.1Init	893
10.186.3.2SetVisible	893
10.187.1gazebo::physics::World Class Reference	893
10.187.1Detailed Description	896
10.187.2Constructor & Destructor Documentation	896
10.187.2.1World	896
10.187.2.2~World	897
10.187.3Member Function Documentation	897
10.187.3.1Clear	897
10.187.3.2DisableAllModels	897
10.187.3.3EnableAllModels	897
10.187.3.4EnablePhysicsEngine	897
10.187.3.5Fini	897
10.187.3.6GetByName	897
10.187.3.7GetEnablePhysicsEngine	898
10.187.3.8GetEntity	898
10.187.3.9GetEntityBelowPoint	898
10.187.3.10GetModel	898
10.187.3.11GetModel	899
10.187.3.12GetModelBelowPoint	899
10.187.3.13GetModelCount	899
10.187.3.14GetModels	899
10.187.3.15GetName	899
10.187.3.16GetPauseTime	900
10.187.3.17GetPhysicsEngine	900
10.187.3.18GetRealTime	900
10.187.3.19GetSelectedEntity	900
10.187.3.20GetSetWorldPoseMutex	900
10.187.3.21GetSimTime	900
10.187.3.22GetStartTime	901
10.187.3.23Init	901
10.187.3.24InsertModelFile	901
10.187.3.25InsertModelSDF	901
10.187.3.26InsertModelString	901
10.187.3.27IsLoaded	901
10.187.3.28IsPaused	902
10.187.3.29Load	902

10.187.3.30	LoadPlugin	902
10.187.3.31	PrintEntityTree	902
10.187.3.32	PublishModelPose	902
10.187.3.33	RemovePlugin	902
10.187.3.34	Reset	903
10.187.3.35	ResetEntities	903
10.187.3.36	ResetTime	903
10.187.3.37	Run	903
10.187.3.38	Save	903
10.187.3.39	SetPaused	903
10.187.3.40	SetSimTime	903
10.187.3.41	SetState	904
10.187.3.42	StepWorld	904
10.187.3.43	Stop	904
10.187.3.44	StripWorldName	904
10.187.3.45	UpdateStateSDF	904
10.187.4	Member Data Documentation	904
10.187.4.1	dirtyPoses	904
10.188	gazebo::WorldPlugin Class Reference	905
10.188.1	Detailed Description	905
10.188.2	Constructor & Destructor Documentation	905
10.188.2.1	WorldPlugin	905
10.188.2.2	~WorldPlugin	906
10.188.3	Member Function Documentation	906
10.188.3.1	Init	906
10.188.3.2	Load	906
10.188.3.3	Reset	906
10.189	gazebo::physics::WorldState Class Reference	906
10.189.1	Detailed Description	907
10.189.2	Constructor & Destructor Documentation	908
10.189.2.1	WorldState	908
10.189.2.2	WorldState	908
10.189.2.3	WorldState	908
10.189.2.4	~WorldState	908
10.189.3	Member Function Documentation	908
10.189.3.1	FillSDF	908
10.189.3.2	GetModelState	908

10.189.3.3	GetModelState	909
10.189.3.4	GetModelStateCount	909
10.189.3.5	GetModelStates	909
10.189.3.6	HasModelState	909
10.189.3.7	IsZero	910
10.189.3.8	Load	910
10.189.3.9	operator+	910
10.189.3.10	operator-	910
10.189.3.11	operator=	910
10.189.3.12	SetWorld	911
10.189.4	Friends And Related Function Documentation	911
10.189.4.1	operator<<	911
11	File Documentation	913
11.1	Actor.hh File Reference	913
11.2	Angle.hh File Reference	914
11.2.1	Macro Definition Documentation	916
11.2.1.1	GZ_DTOR	916
11.2.1.2	GZ_NORMALIZE	916
11.2.1.3	GZ_RTOD	916
11.3	Animation.hh File Reference	917
11.4	ArrowVisual.hh File Reference	918
11.5	Assert.hh File Reference	919
11.5.1	Macro Definition Documentation	920
11.5.1.1	GZ_ASSERT	920
11.6	AxisVisual.hh File Reference	920
11.7	BallJoint.hh File Reference	921
11.8	Base.hh File Reference	922
11.9	Box.hh File Reference	923
11.10	BoxShape.hh File Reference	923
11.11	BVHLoader.hh File Reference	925
11.11.1	Macro Definition Documentation	926
11.11.1.1	X_POSITION	926
11.11.1.2	X_ROTATION	926
11.11.1.3	Y_POSITION	926
11.11.1.4	Y_ROTATION	926
11.11.1.5	Z_POSITION	926

11.11.1.6 Z_ROTATION	926
11.12CallbackHelper.hh File Reference	926
11.13Camera.hh File Reference	928
11.14CameraSensor.hh File Reference	929
11.15CameraVisual.hh File Reference	930
11.16cegui.h File Reference	930
11.17ColladaLoader.hh File Reference	931
11.18Collision.hh File Reference	933
11.19CollisionState.hh File Reference	934
11.20Color.hh File Reference	935
11.21Common.hh File Reference	936
11.22CommonTypes.hh File Reference	938
11.22.1 Macro Definition Documentation	939
11.22.1.1 GAZEBO_DEPRECATED	939
11.22.1.2 GAZEBO_FORCEINLINE	939
11.22.1.3 NULL	939
11.23COMVisual.hh File Reference	939
11.24Connection.hh File Reference	940
11.24.1 Macro Definition Documentation	942
11.24.1.1 HEADER_LENGTH	942
11.25ConnectionManager.hh File Reference	942
11.26Console.hh File Reference	943
11.27Contact.hh File Reference	945
11.27.1 Macro Definition Documentation	946
11.27.1.1 MAX_COLLIDE_RETURNS	946
11.27.1.2 MAX_CONTACT_JOINTS	946
11.28ContactManager.hh File Reference	946
11.29ContactSensor.hh File Reference	947
11.30ContactVisual.hh File Reference	947
11.31Conversions.hh File Reference	948
11.32Converter.hh File Reference	949
11.33CylinderShape.hh File Reference	950
11.34DepthCamera.hh File Reference	951
11.35DepthCameraSensor.hh File Reference	951
11.36Diagnostics.hh File Reference	952
11.37DynamicLines.hh File Reference	953
11.38DynamicRenderable.hh File Reference	954

11.39Entity.hh File Reference	955
11.40Event.hh File Reference	956
11.41Events.hh File Reference	957
11.42Exception.hh File Reference	958
11.43FPSViewController.hh File Reference	959
11.44gazebo.hh File Reference	960
11.45gazebo_core.hh File Reference	961
11.46GazeboGenerator.hh File Reference	961
11.47GpuLaser.hh File Reference	962
11.48GpuRaySensor.hh File Reference	963
11.49Grid.hh File Reference	964
11.50Gripper.hh File Reference	964
11.51GUIOverlay.hh File Reference	966
11.52Heightmap.hh File Reference	966
11.53HeightmapShape.hh File Reference	967
11.54Helpers.hh File Reference	968
11.54.1 Macro Definition Documentation	970
11.54.1.1 GZ_DBL_MAX	970
11.54.1.2 GZ_DBL_MIN	970
11.54.1.3 GZ_FLT_MAX	970
11.54.1.4 GZ_FLT_MIN	970
11.55Hinge2Joint.hh File Reference	970
11.56HingeJoint.hh File Reference	972
11.57Image.hh File Reference	973
11.58ImuSensor.hh File Reference	974
11.59Inertial.hh File Reference	975
11.60IOManager.hh File Reference	976
11.61Joint.hh File Reference	977
11.61.1 Macro Definition Documentation	978
11.61.1.1 MAX_JOINT_AXIS	978
11.62JointController.hh File Reference	978
11.63JointState.hh File Reference	979
11.64JointVisual.hh File Reference	980
11.65JointWrench.hh File Reference	981
11.66KeyFrame.hh File Reference	982
11.67LaserVisual.hh File Reference	983
11.68Light.hh File Reference	984

11.69Link.hh File Reference	985
11.70LinkState.hh File Reference	986
11.71LogPlay.hh File Reference	988
11.72LogRecord.hh File Reference	989
11.72.1 Macro Definition Documentation	991
11.72.1.1 GZ_LOG_VERSION	991
11.73mainpage.html File Reference	991
11.74MapShape.hh File Reference	991
11.75Master.hh File Reference	992
11.76Material.hh File Reference	993
11.77Material.hh File Reference	995
11.78MathTypes.hh File Reference	995
11.78.1 Detailed Description	995
11.79Matrix3.hh File Reference	996
11.80Matrix4.hh File Reference	996
11.81Mesh.hh File Reference	997
11.82MeshCSG.hh File Reference	999
11.82.1 Typedef Documentation	999
11.82.1.1 GPtrArray	999
11.82.1.2 GtsSurface	1000
11.83MeshLoader.hh File Reference	1000
11.84MeshManager.hh File Reference	1001
11.85Model.hh File Reference	1002
11.86ModelDatabase.hh File Reference	1004
11.86.1 Macro Definition Documentation	1004
11.86.1.1 GZ_MODEL_DB_MANIFEST_FILENAME	1004
11.86.1.2 GZ_MODEL_MANIFEST_FILENAME	1005
11.87ModelState.hh File Reference	1005
11.88MouseEvent.hh File Reference	1007
11.89MovableText.hh File Reference	1008
11.90MsgFactory.hh File Reference	1008
11.91msgs.hh File Reference	1009
11.92MultiCameraSensor.hh File Reference	1012
11.93MultiRayShape.hh File Reference	1012
11.94Node.hh File Reference	1014
11.95ogre_gazebo.h File Reference	1015
11.96OrbitViewController.hh File Reference	1016

11.97Param.hh File Reference	1016
11.98parser.hh File Reference	1018
11.99parser_urdf.hh File Reference	1019
11.100Physics.hh File Reference	1020
11.101PhysicsEngine.hh File Reference	1021
11.102PhysicsFactory.hh File Reference	1023
11.103PhysicsTypes.hh File Reference	1024
11.103.1Detailed Description	1026
11.103.2Macro Definition Documentation	1026
11.103.2.1GZ_ALL_COLLIDE	1026
11.103.2.2GZ_FIXED_COLLIDE	1026
11.103.2.3GZ_GHOST_COLLIDE	1026
11.103.2.4GZ_NONE_COLLIDE	1026
11.103.2.5GZ_SENSOR_COLLIDE	1026
11.104PID.hh File Reference	1026
11.105Plane.hh File Reference	1027
11.106PlaneShape.hh File Reference	1028
11.107Plugin.hh File Reference	1030
11.107.1Macro Definition Documentation	1032
11.107.1.1GZ_REGISTER_MODEL_PLUGIN	1032
11.107.1.2GZ_REGISTER_SENSOR_PLUGIN	1032
11.107.1.3GZ_REGISTER_SYSTEM_PLUGIN	1033
11.107.1.4GZ_REGISTER_VISUAL_PLUGIN	1033
11.107.1.5GZ_REGISTER_WORLD_PLUGIN	1033
11.108Plugin.hh File Reference	1034
11.109Pose.hh File Reference	1034
11.110Projector.hh File Reference	1035
11.111Publication.hh File Reference	1036
11.112PublicationTransport.hh File Reference	1037
11.113Publisher.hh File Reference	1039
11.114Quaternion.hh File Reference	1041
11.115Sand.hh File Reference	1042
11.116RaySensor.hh File Reference	1043
11.117RayShape.hh File Reference	1044
11.118RenderEngine.hh File Reference	1045
11.119RenderEvents.hh File Reference	1046
11.120Rendering.hh File Reference	1047

11.12	RenderTypes.hh File Reference	1048
11.121	Macro Definition Documentation	1049
11.121.1	1.GZ_VISIBILITY_ALL	1049
11.121.1	2.GZ_VISIBILITY_GUI	1049
11.121.1	3.GZ_VISIBILITY_NOT_SELECTABLE	1049
11.121.1	4.GZ_VISIBILITY_SELECTION	1050
11.122	RFIDSensor.hh File Reference	1050
11.123	RFIDTag.hh File Reference	1050
11.124	RFIDTagVisual.hh File Reference	1051
11.125	RFIDVisual.hh File Reference	1052
11.126	Road.hh File Reference	1052
11.127	Road2d.hh File Reference	1054
11.128	RotationSpline.hh File Reference	1054
11.129	RTShaderSystem.hh File Reference	1055
11.130	Scene.hh File Reference	1056
11.131	ScrewJoint.hh File Reference	1057
11.132	df.hh File Reference	1058
11.133	SDF.hh File Reference	1059
11.133	Macro Definition Documentation	1060
11.133.1	1.SDF_VERSION	1060
11.134	SelectionObj.hh File Reference	1060
11.135	Sensor.hh File Reference	1061
11.136	SensorFactory.hh File Reference	1062
11.137	SensorManager.hh File Reference	1063
11.138	Sensors.hh File Reference	1064
11.139	SensorTypes.hh File Reference	1066
11.139	Detailed Description	1067
11.140	Server.hh File Reference	1067
11.141	Shape.hh File Reference	1068
11.142	SingletonT.hh File Reference	1070
11.143	Skeleton.hh File Reference	1070
11.144	SkeletonAnimation.hh File Reference	1072
11.145	SliderJoint.hh File Reference	1073
11.146	SphereShape.hh File Reference	1075
11.147	Spline.hh File Reference	1076
11.148	State.hh File Reference	1077
11.149	STLLoader.hh File Reference	1078

11.149.1	Macro Definition Documentation	1079
11.149.1.1	COR3_MAX	1079
11.149.1.2	FACE_MAX	1080
11.149.1.3	LINE_MAX_LEN	1080
11.149.1.4	ORDER_MAX	1080
11.150	SubscribeOptions.hh File Reference	1080
11.151	Subscriber.hh File Reference	1081
11.152	SubscriptionTransport.hh File Reference	1083
11.153	SurfaceParams.hh File Reference	1084
11.154	SystemPaths.hh File Reference	1086
11.154.1	Macro Definition Documentation	1087
11.154.1.1	GetCurrentDir	1087
11.154.1.2	LINUX	1087
11.155	Time.hh File Reference	1087
11.156	Timer.hh File Reference	1088
11.157	TopicManager.hh File Reference	1089
11.158	Transport.hh File Reference	1091
11.159	TransportTypes.hh File Reference	1093
11.159.1	Detailed Description	1094
11.160	TrimeshShape.hh File Reference	1095
11.161	UniversalJoint.hh File Reference	1096
11.162	UpdateInfo.hh File Reference	1097
11.163	UserCamera.hh File Reference	1098
11.164	UtilTypes.hh File Reference	1098
11.165	Vector2d.hh File Reference	1099
11.166	Vector2i.hh File Reference	1100
11.167	Vector3.hh File Reference	1101
11.168	Vector4.hh File Reference	1102
11.169	Video.hh File Reference	1104
11.170	VideoVisual.hh File Reference	1105
11.171	ViewController.hh File Reference	1106
11.172	Visual.hh File Reference	1107
11.173	WindowManager.hh File Reference	1108
11.174	WireBox.hh File Reference	1108
11.175	World.hh File Reference	1109
11.176	WorldState.hh File Reference	1111

Index

1112

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 (p. 109) set of classes that encapsulate math related properties and functions.

Messages: All messages and helper functions.

Physics: Classes for physics and dynamics

Rendering: A (p. 109) set of rendering related class, functions, and definitions.

Sensors: A (p. 109) 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 (p. 109) 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. 383) (`int _index`) `GAZEBO_DEPRECATED`

: not yet implemented. Get the forces applied at this Joint. Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Member `gazebo::physics::Joint::GetForce` (p. 383) (`unsigned int _index`)

: not yet implemented. Get the forces applied at this Joint. Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Member `gazebo::sensors::CameraSensor::GetTopic` (p. 187) () `const`

to be implemented

Class `gazebo::SystemPlugin` (p. 774)

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	27
Events	38
Classes for physics and dynamics	41
Math	48
Messages	54
Rendering	64
Gazebo_parser	68
Sensors	69
Transport	74
Utility	79

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

boost	81
gazebo	
Forward declarations for the common classes	81
gazebo::common	
Common namespace	83
gazebo::event	
Event (p. 288) namespace	86
gazebo::math	
Math namespace	87
gazebo::msgs	
Messages namespace	89
gazebo::physics	
Namespace for physics	91
gazebo::rendering	
Rendering namespace	96
gazebo::sensors	
Sensors namespace	100
gazebo::transport	103
gazebo::util	105
google	105
google::protobuf	105
google::protobuf::compiler	105
google::protobuf::compiler::cpp	105
Ogre	106
ogre	106
sdf	
Namespace for Simulation Description Format parser	106
SkyX	108
urdf2gazebo	
Namespace for URDF to SDF parser	108

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

A	109
gazebo::math::Angle	116
gazebo::common::Animation	123
gazebo::common::NumericAnimation	542
gazebo::common::PoseAnimation	593
gazebo::math::Box	146
gazebo::common::BVHLoader	154
gazebo::transport::CallbackHelper	155
gazebo::transport::CallbackHelperT< M >	157
gazebo::transport::RawCallbackHelper	625
gazebo::transport::SubscriptionTransport	763
CodeGenerator	
google::protobuf::compiler::cpp::GazeboGenerator	319
gazebo::common::Color	205
gazebo::event::Connection	218
gazebo::physics::Contact	232
gazebo::physics::ContactManager	235
gazebo::rendering::Conversions	243
sdf::Converter	245
enable_shared_from_this	
gazebo::physics::Base	135
gazebo::physics::Entity	277
gazebo::physics::Collision	192
gazebo::physics::Link	412
gazebo::physics::Model	480
gazebo::physics::Actor	109
gazebo::physics::Joint	375
gazebo::physics::Road	651
gazebo::physics::Shape	706
gazebo::physics::BoxShape	151
gazebo::physics::CylinderShape	246
gazebo::physics::HeightmapShape	348
gazebo::physics::MultiRayShape	518

gazebo::physics::PlaneShape	578
gazebo::physics::RayShape	634
gazebo::physics::SphereShape	737
gazebo::physics::TrimeshShape	807
gazebo::physics::World	893
gazebo::rendering::Camera	159
gazebo::rendering::DepthCamera	249
gazebo::rendering::GpuLaser	320
gazebo::rendering::UserCamera	814
gazebo::rendering::Scene	662
gazebo::rendering::Visual	868
gazebo::rendering::ArrowVisual	127
gazebo::rendering::AxisVisual	131
gazebo::rendering::CameraVisual	189
gazebo::rendering::COMVisual	216
gazebo::rendering::ContactVisual	242
gazebo::rendering::JointVisual	399
gazebo::rendering::LaserVisual	404
gazebo::rendering::RFIDTagVisual	648
gazebo::rendering::RFIDVisual	650
gazebo::rendering::VideoVisual	864
gazebo::sensors::Sensor	683
gazebo::sensors::CameraSensor	185
gazebo::sensors::ContactSensor	237
gazebo::sensors::DepthCameraSensor	254
gazebo::sensors::GpuRaySensor	324
gazebo::sensors::ImuSensor	361
gazebo::sensors::MultiCameraSensor	514
gazebo::sensors::RaySensor	627
gazebo::sensors::RFIDSensor	644
gazebo::sensors::RFIDTag	646
gazebo::transport::Connection	219
gazebo::transport::Node	526
sdf::Element	269
gazebo::event::Event	288
gazebo::event::EventT< void()>	305
gazebo::event::EventT< void(bool)>	305
gazebo::event::EventT< void(const common::UpdateInfo &)>	305
gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>	305
gazebo::event::EventT< void(const std::string &)>	305
gazebo::event::EventT< void(const std::string &, const Contact &)>	305
gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>	305
gazebo::event::EventT< void(std::string)>	305
gazebo::event::EventT< void(std::string, std::string)>	305
gazebo::event::EventT< T >	305
gazebo::rendering::Events	291
gazebo::event::Events	293
gazebo::common::Exception	314
gazebo::common::InternalError	372
gazebo::common::AssertionInternalError	129
urdf2gazebo::GazeboExtension	319
gazebo::rendering::Grid	336

gazebo::physics::Gripper	340
gazebo::rendering::GUIOverlay	341
gazebo::rendering::Heightmap	346
gazebo::common::Image	356
gazebo::physics::Inertial	365
gazebo::transport::IOManager	374
gazebo::physics::JointController	393
gazebo::physics::JointWrench	401
gazebo::common::KeyFrame	402
gazebo::common::NumericKeyFrame	544
gazebo::common::PoseKeyFrame	595
gazebo::rendering::Light	405
Logplay	439
gazebo::Master	445
gazebo::common::Material	446
gazebo::math::Matrix3	456
gazebo::math::Matrix4	459
gazebo::common::Mesh	466
gazebo::common::MeshCSG	472
gazebo::common::MeshLoader	474
gazebo::common::ColladaLoader	190
gazebo::common::STLLoader	748
gazebo::common::MouseEvent	503
MovableObject	
gazebo::rendering::MovableText	506
gazebo::msgs::MsgFactory	512
gazebo::common::NodeAnimation	532
gazebo::common::NodeAssignment	536
gazebo::common::NodeTransform	537
sdf::Param	549
sdf::ParamT< std::string >	555
sdf::ParamT< T >	555
ParamT< T >	555
gazebo::physics::PhysicsEngine	558
gazebo::physics::PhysicsFactory	570
gazebo::common::PID	571
gazebo::math::Plane	575
gazebo::PluginT< T >	582
gazebo::PluginT< ModelPlugin >	582
gazebo::ModelPlugin	495
gazebo::PluginT< SensorPlugin >	582
gazebo::SensorPlugin	698
gazebo::PluginT< SystemPlugin >	582
gazebo::SystemPlugin	774
gazebo::PluginT< VisualPlugin >	582
gazebo::VisualPlugin	888
gazebo::PluginT< WorldPlugin >	582
gazebo::WorldPlugin	905
gazebo::math::Pose	584
gazebo::rendering::Projector	597
gazebo::transport::Publication	600
gazebo::transport::PublicationTransport	604

gazebo::transport::Publisher	606
gazebo::math::Quaternion	609
gazebo::math::Rand	623
Renderable	
gazebo::rendering::MovableText	506
RenderObjectListener	
gazebo::rendering::GpuLaser	320
Road	651
gazebo::rendering::Road2d	653
gazebo::math::RotationSpline	654
sdf::SDF	680
SDFBase	
sdf::Plugin	581
gazebo::rendering::SelectionObj	681
SensorFactor	693
gazebo::sensors::SensorFactory	693
gazebo::Server	700
ShaderHelperCg	
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg	701
ShaderHelperGLSL	
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL	703
SimpleRenderable	
gazebo::rendering::DynamicRenderable	265
gazebo::rendering::DynamicLines	262
gazebo::sensors::SimTimeEvent	709
SingletonT< T >	711
gazebo::common::Console	231
gazebo::common::LogPlay	437
gazebo::common::LogRecord	440
gazebo::common::MeshManager	475
gazebo::common::ModelDatabase	494
gazebo::common::SystemPaths	769
gazebo::rendering::RenderEngine	639
gazebo::rendering::RTShaderSystem	658
gazebo::sensors::SensorManager	695
gazebo::sensors::SimTimeEventHandler	709
gazebo::transport::ConnectionManager	225
gazebo::transport::TopicManager	799
gazebo::util::DiagnosticManager	257
SingletonT< ConnectionManager >	711
SingletonT< Console >	711
SingletonT< DiagnosticManager >	711
SingletonT< LogPlay >	711
SingletonT< LogRecord >	711
SingletonT< MeshManager >	711
SingletonT< ModelDatabase >	711
SingletonT< RenderEngine >	711
SingletonT< RTShaderSystem >	711
SingletonT< SensorManager >	711
SingletonT< SimTimeEventHandler >	711
SingletonT< SystemPaths >	711
SingletonT< TopicManager >	711
SingletonT< WindowManager >	711
gazebo::rendering::WindowManager	889

gazebo::common::Skeleton	713
gazebo::common::SkeletonAnimation	720
gazebo::common::SkeletonNode	724
SM2Profile	
gazebo::rendering::GzTerrainMatGen::SM2Profile	735
gazebo::math::Spline	740
gazebo::physics::State	744
gazebo::physics::CollisionState	201
gazebo::physics::JointState	395
gazebo::physics::LinkState	431
gazebo::physics::ModelState	497
gazebo::physics::WorldState	906
gazebo::common::SubMesh	750
gazebo::transport::SubscribeOptions	759
gazebo::transport::Subscriber	761
gazebo::physics::SurfaceParams	765
T	
gazebo::physics::BallJoint< T >	133
gazebo::physics::Hinge2Joint< T >	352
gazebo::physics::HingeJoint< T >	354
gazebo::physics::ScrewJoint< T >	677
gazebo::physics::SliderJoint< T >	733
gazebo::physics::UniversalJoint< T >	810
task	
gazebo::transport::ConnectionReadTask	230
TerrainMaterialGeneratorA	
gazebo::rendering::GzTerrainMatGen	345
gazebo::common::Time	776
gazebo::common::Timer	797
gazebo::util::DiagnosticTimer	260
gazebo::physics::TrajectoryInfo	806
gazebo::common::UpdateInfo	811
urdf2gazebo::URDF2Gazebo	812
gazebo::math::Vector2d	822
gazebo::math::Vector2i	830
gazebo::math::Vector3	839
gazebo::math::Vector4	853
gazebo::common::Video	862
gazebo::rendering::ViewController	865
gazebo::rendering::FPSViewController	316
gazebo::rendering::OrbitViewController	545
gazebo::rendering::WireBox	892

Chapter 6

Class Index

6.1 Class List

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

A

Holding gazebo extension elements in urdf	109
gazebo::physics::Actor	
Actor (p. 109) class enables GPU based mesh model / skeleton scriptable animation	109
gazebo::math::Angle	
An angle and related functions	116
gazebo::common::Animation	
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes	123
gazebo::rendering::ArrowVisual	
Basic arrow visualization	127
gazebo::common::AssertionInternalError	
Class for generating Exceptions which come from gazebo assertions	129
gazebo::rendering::AxisVisual	
Basic axis visualization	131
gazebo::physics::BallJoint< T >	
Base (p. 135) class for a ball joint	133
gazebo::physics::Base	
Base (p. 135) class for most physics classes	135
gazebo::math::Box	
Mathematical representation of a box and related functions	146
gazebo::physics::BoxShape	
Box geometry primitive	151
gazebo::common::BVHLoader	
Handles loading BVH animation files	154
gazebo::transport::CallbackHelper	
A (p. 109) helper class to handle callbacks when messages arrive	155
gazebo::transport::CallbackHelperT< M >	
Callback helper Template	157
gazebo::rendering::Camera	
Basic camera sensor	159
gazebo::sensors::CameraSensor	
Basic camera sensor	185

gazebo::rendering::CameraVisual	
Basic camera visualization	189
gazebo::common::ColladaLoader	
Class used to load Collada mesh files	190
gazebo::physics::Collision	
Base (p. 135) class for all collision entities	192
gazebo::physics::CollisionState	
Store state information of a physics::Collision (p. 192) object	201
gazebo::common::Color	
Defines a color	205
gazebo::rendering::COMVisual	
Basic Center of Mass visualization	216
gazebo::event::Connection	
A (p. 109) class that encapsulates a connection	218
gazebo::transport::Connection	
Single TCP/IP connection manager	219
gazebo::transport::ConnectionManager	
Manager of connections	225
gazebo::transport::ConnectionReadTask	
230	
gazebo::common::Console	
Message, error, warning functionality	231
gazebo::physics::Contact	
A (p. 109) contact between two collisions	232
gazebo::physics::ContactManager	
Aggregates all the contact information generated by the collision detection engine	235
gazebo::sensors::ContactSensor	
Contact sensor	237
gazebo::rendering::ContactVisual	
Contact visualization	242
gazebo::rendering::Conversions	
Conversions (p. 243) Conversions.hh (p. 948) rendering/Conversions.hh (p. 948)	243
sdf::Converter	
Convert from one version of SDF (p. 680) to another	245
gazebo::physics::CylinderShape	
Cylinder collision	246
gazebo::rendering::DepthCamera	
Depth camera used to render depth data into an image buffer	249
gazebo::sensors::DepthCameraSensor	254
gazebo::util::DiagnosticManager	
A (p. 109) diagnostic manager class	257
gazebo::util::DiagnosticTimer	
A (p. 109) timer designed for diagnostics	260
gazebo::rendering::DynamicLines	
Class for drawing lines that can change	262
gazebo::rendering::DynamicRenderable	
Abstract base class providing mechanisms for dynamically growing hardware buffers	265
sdf::Element	
SDF (p. 680) Element (p. 269) class	269
gazebo::physics::Entity	
Base (p. 135) class for all physics objects in Gazebo	277
gazebo::event::Event	
Base class for all events	288

gazebo::rendering::Events	
Base class for rendering events	291
gazebo::event::Events	
An Event (p. 288) class to get notifications for simulator events	293
gazebo::event::EventT< T >	
A (p. 109) class for event processing	305
gazebo::common::Exception	
Class for generating exceptions	314
gazebo::rendering::FPSViewController	
First Person Shooter style view controller	316
urdf2gazebo::GazeboExtension	319
google::protobuf::compiler::cpp::GazeboGenerator	
Google protobuf message generator for gazebo::msgs (p. 89)	319
gazebo::rendering::GpuLaser	
GPU based laser distance sensor	320
gazebo::sensors::GpuRaySensor	324
gazebo::rendering::Grid	
Displays a grid of cells, drawn with lines	336
gazebo::physics::Gripper	
A (p. 109) gripper abstraction	340
gazebo::rendering::GUIOverlay	
A (p. 109) class that creates a CEGUI overlay on a render window	341
gazebo::rendering::GzTerrainMatGen	345
gazebo::rendering::Heightmap	
Rendering a terrain using heightmap information	346
gazebo::physics::HeightmapShape	
HeightmapShape (p. 348) collision shape builds a heightmap from an image	348
gazebo::physics::Hinge2Joint< T >	
A (p. 109) two axis hinge joint	352
gazebo::physics::HingeJoint< T >	
A (p. 109) single axis hinge joint	354
gazebo::common::Image	
Encapsulates an image	356
gazebo::sensors::ImuSensor	
An IMU sensor	361
gazebo::physics::Inertial	
A (p. 109) class for inertial information about a link	365
gazebo::common::InternalError	
Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs	372
gazebo::transport::IOManager	
Manages boost::asio IO	374
gazebo::physics::Joint	
Base (p. 135) class for all joints	375
gazebo::physics::JointController	
A (p. 109) class for manipulating physics::Joint (p. 375)	393
gazebo::physics::JointState	
Keeps track of state of a physics::Joint (p. 375)	395
gazebo::rendering::JointVisual	
Visualization for joints	399
gazebo::physics::JointWrench	
Wrench information from a joint	401
gazebo::common::KeyFrame	
A (p. 109) key frame in an animation	402

gazebo::rendering::LaserVisual	
Visualization for laser data	404
gazebo::rendering::Light	
A (p. 109) light source	405
gazebo::physics::Link	
Link (p. 412) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body	412
gazebo::physics::LinkState	
Store state information of a physics::Link (p. 412) object	431
gazebo::common::LogPlay	437
Logplay	
Open and playback log files that were recorded using LogRecord	439
gazebo::common::LogRecord	
Addtogroup gazebo_common	440
gazebo::Master	
A (p. 109) ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication	445
gazebo::common::Material	
Encapsulates description of a material	446
gazebo::math::Matrix3	
A (p. 109) 3x3 matrix class	456
gazebo::math::Matrix4	
A (p. 109) 3x3 matrix class	459
gazebo::common::Mesh	
A (p. 109) 3D mesh	466
gazebo::common::MeshCSG	
Creates CSG meshes	472
gazebo::common::MeshLoader	
Base class for loading meshes	474
gazebo::common::MeshManager	
Maintains and manages all meshes	475
gazebo::physics::Model	
A (p. 109) model is a collection of links, joints, and plugins	480
gazebo::common::ModelDatabase	
Connects to model database, and has utility functions to find models	494
gazebo::ModelPlugin	
A (p. 109) plugin with access to physics::Model (p. 480)	495
gazebo::physics::ModelState	
Store state information of a physics::Model (p. 480) object	497
gazebo::common::MouseEvent	
Generic description of a mouse event	503
gazebo::rendering::MovableText	
Movable text	506
gazebo::msgs::MsgFactory	
A (p. 109) factory that generates protobuf message based on a string type	512
gazebo::sensors::MultiCameraSensor	
Multiple camera sensor	514
gazebo::physics::MultiRayShape	
Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner	518
gazebo::transport::Node	
A (p. 109) node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics	526
gazebo::common::NodeAnimation	
Node animation	532

gazebo::common::NodeAssignment	
Vertex to node weighted assignement for skeleton animation visualization	536
gazebo::common::NodeTransform	
NodeTransform (p. 537) Skeleton.hh (p. 1070) common/common.hh	537
gazebo::common::NumericAnimation	
A (p. 109) numeric animation	542
gazebo::common::NumericKeyFrame	
A (p. 109) keyframe for a NumericAnimation (p. 542)	544
gazebo::rendering::OrbitViewController	
Orbit view controller	545
sdf::Param	
A (p. 109) parameter class	549
ParamT < T >	555
sdf::ParamT < T >	
Templatized parameter class	555
gazebo::physics::PhysicsEngine	
Base (p. 135) class for a physics engine	558
gazebo::physics::PhysicsFactory	
The physics factory instantiates different physics engines	570
gazebo::common::PID	
Generic PID (p. 571) controller class	571
gazebo::math::Plane	
A (p. 109) plane and related functions	575
gazebo::physics::PlaneShape	
Collision (p. 192) for an infinite plane	578
sdf::Plugin	581
gazebo::PluginT < T >	
A (p. 109) class which all plugins must inherit from	582
gazebo::math::Pose	
Encapsulates a position and rotation in three space	584
gazebo::common::PoseAnimation	
A (p. 109) pose animation	593
gazebo::common::PoseKeyFrame	
A (p. 109) keyframe for a PoseAnimation (p. 593)	595
gazebo::rendering::Projector	
Projects a material onto surface, light a light projector	597
gazebo::transport::Publication	
A (p. 109) publication for a topic	600
gazebo::transport::PublicationTransport	
Transport/transport.hh	604
gazebo::transport::Publisher	
A (p. 109) publisher of messages on a topic	606
gazebo::math::Quaternion	
A (p. 109) quaternion class	609
gazebo::math::Rand	
Random number generator class	623
gazebo::transport::RawCallbackHelper	
Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher	625
gazebo::sensors::RaySensor	
Sensor (p. 683) with one or more rays	627
gazebo::physics::RayShape	
Base (p. 135) class for Ray collision geometry	634

gazebo::rendering::RenderEngine	
Adaptor to Ogre3d	639
gazebo::sensors::RFIDSensor	
Sensor (p. 683) class for RFID type of sensor	644
gazebo::sensors::RFIDTag	
RFIDTag (p. 646) to interact with RFIDTagSensors	646
gazebo::rendering::RFIDTagVisual	
Visualization for RFID tags sensor	648
gazebo::rendering::RFIDVisual	
Visualization for RFID sensor	650
Road	
Used to render a strip of road	651
gazebo::physics::Road	
For building a Road (p. 651) from SDF	651
gazebo::rendering::Road2d	653
gazebo::math::RotationSpline	
Spline (p. 740) for rotations	654
gazebo::rendering::RTShaderSystem	
Implements Ogre (p. 106)'s Run-Time Shader system	658
gazebo::rendering::Scene	
Representation of an entire scene graph	662
gazebo::physics::ScrewJoint< T >	
A (p. 109) screw joint, which has both prismatic and rotational DOFs	677
sdf::SDF	
Base SDF (p. 680) class	680
gazebo::rendering::SelectionObj	
A (p. 109) graphical selection object	681
gazebo::sensors::Sensor	
Base class for sensors	683
SensorFactor	
The sensor factory; the class is just for namespacing purposes	693
gazebo::sensors::SensorFactory	693
gazebo::sensors::SensorManager	
Class to manage and update all sensors	695
gazebo::SensorPlugin	
A (p. 109) plugin with access to physics::Sensor	698
gazebo::Server	700
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg	
Keeping the CG shader for reference	701
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL	
Utility class to help with generating shaders for GLSL	703
gazebo::physics::Shape	
Base (p. 135) class for all shapes	706
gazebo::sensors::SimTimeEvent	
709	
gazebo::sensors::SimTimeEventHandler	
Monitors simulation time, and notifies conditions when a specified time has been reached	709
SingletonT< T >	
Singleton template class	711
gazebo::common::Skeleton	
A (p. 109) skeleton	713
gazebo::common::SkeletonAnimation	
Skeleton (p. 713) animation	720

gazebo::common::SkeletonNode	
A (p. 109) skeleton node	724
gazebo::physics::SliderJoint< T >	
A (p. 109) slider joint	733
gazebo::rendering::GzTerrainMatGen::SM2Profile	
Shader model 2 profile target	735
gazebo::physics::SphereShape	
Sphere collision shape	737
gazebo::math::Spline	
Splines	740
gazebo::physics::State	
State (p. 744) of an entity	744
gazebo::common::STLLoader	
Class used to load STL mesh files	748
gazebo::common::SubMesh	
A (p. 109) child mesh	750
gazebo::transport::SubscribeOptions	
Options for a subscription	759
gazebo::transport::Subscriber	
A (p. 109) subscriber to a topic	761
gazebo::transport::SubscriptionTransport	
Transport/transport.hh	763
gazebo::physics::SurfaceParams	
SurfaceParams (p. 765) defines various Surface contact parameters	765
gazebo::common::SystemPaths	
Functions to handle getting system paths, keeps track of:	769
gazebo::SystemPlugin	
A (p. 109) plugin loaded within the gzserver on startup	774
gazebo::common::Time	
A (p. 109) Time (p. 776) class, can be used to hold wall- or sim-time	776
gazebo::common::Timer	
A (p. 109) timer class, used to time things in real world walltime	797
gazebo::transport::TopicManager	
Manages topics and their subscriptions	799
gazebo::physics::TrajectoryInfo	806
gazebo::physics::TrimeshShape	
Triangle mesh collision shape	807
gazebo::physics::UniversalJoint< T >	
A (p. 109) universal joint	810
gazebo::common::UpdateInfo	
Information for use in an update event	811
urdf2gazebo::URDF2Gazebo	812
gazebo::rendering::UserCamera	
A (p. 109) camera used for user visualization of a scene	814
gazebo::math::Vector2d	
Generic double x, y vector	822
gazebo::math::Vector2i	
Generic integer x, y vector	830
gazebo::math::Vector3	
Generic vector containing 3 elements	839
gazebo::math::Vector4	
Double Generic x, y, z, w vector	853
gazebo::common::Video	
Handle video encoding and decoding using libavcodec	862

gazebo::rendering::VideoVisual	
A (p. 109) visual element that displays a video as a texture	864
gazebo::rendering::ViewController	
Base class for view controllers	865
gazebo::rendering::Visual	
A (p. 109) renderable object	868
gazebo::VisualPlugin	
A (p. 109) plugin loaded within the gzserver on startup	888
gazebo::rendering::WindowManager	
Class to manage render windows	889
gazebo::rendering::WireBox	
Draws a wireframe box	892
gazebo::physics::World	
The world provides access to all other object within a simulated environment	893
gazebo::WorldPlugin	
A (p. 109) plugin with access to physics::World (p. 893)	905
gazebo::physics::WorldState	
Store state information of a physics::World (p. 893) object	906

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Actor.hh	913
Angle.hh	914
Animation.hh	917
ArrowVisual.hh	918
Assert.hh	919
AxisVisual.hh	920
BallJoint.hh	921
Base.hh	922
Box.hh	923
BoxShape.hh	923
BVHLoader.hh	925
CallbackHelper.hh	926
Camera.hh	928
CameraSensor.hh	929
CameraVisual.hh	930
cegui.h	930
ColladaLoader.hh	931
Collision.hh	933
CollisionState.hh	934
Color.hh	935
Common.hh	936
CommonTypes.hh	938
COMVisual.hh	939
Connection.hh	940
ConnectionManager.hh	942
Console.hh	943
Contact.hh	945
ContactManager.hh	946
ContactSensor.hh	947
ContactVisual.hh	947
Conversions.hh	948
Converter.hh	949
CylinderShape.hh	950
DepthCamera.hh	951

DepthCameraSensor.hh	951
Diagnostics.hh	952
DynamicLines.hh	953
DynamicRenderable.hh	954
Entity.hh	955
Event.hh	956
Events.hh	957
Exception.hh	958
FPSViewController.hh	959
gazebo.hh	960
gazebo_core.hh	961
GazeboGenerator.hh	961
GpuLaser.hh	962
GpuRaySensor.hh	963
Grid.hh	964
Gripper.hh	964
GUIOverlay.hh	966
Heightmap.hh	966
HeightmapShape.hh	967
Helpers.hh	968
Hinge2Joint.hh	970
HingeJoint.hh	972
Image.hh	973
ImuSensor.hh	974
Inertial.hh	975
IOManager.hh	976
Joint.hh	977
JointController.hh	978
JointState.hh	979
JointVisual.hh	980
JointWrench.hh	981
KeyFrame.hh	982
LaserVisual.hh	983
Light.hh	984
Link.hh	985
LinkState.hh	986
LogPlay.hh	988
LogRecord.hh	989
mainpage.html	991
MapShape.hh	991
Master.hh	992
common/Material.hh	993
rendering/Material.hh	995
MathTypes.hh	
Forward declarations for the math classes	995
Matrix3.hh	996
Matrix4.hh	996
Mesh.hh	997
MeshCSG.hh	999
MeshLoader.hh	1000
MeshManager.hh	1001
Model.hh	1002
ModelDatabase.hh	1004
ModelState.hh	1005

MouseEvent.hh	1007
MovableText.hh	1008
MsgFactory.hh	1008
msgs.hh	1009
MultiCameraSensor.hh	1012
MultiRayShape.hh	1012
Node.hh	1014
ogre_gazebo.h	1015
OrbitViewController.hh	1016
Param.hh	1016
parser.hh	1018
parser_urdf.hh	1019
Physics.hh	1020
PhysicsEngine.hh	1021
PhysicsFactory.hh	1023
PhysicsTypes.hh	
Default namespace for gazebo	1024
PID.hh	1026
Plane.hh	1027
PlaneShape.hh	1028
common/Plugin.hh	1030
sdf/interface/Plugin.hh	1034
Pose.hh	1034
Projector.hh	1035
Publication.hh	1036
PublicationTransport.hh	1037
Publisher.hh	1039
Quaternion.hh	1041
Rand.hh	1042
RaySensor.hh	1043
RayShape.hh	1044
RenderEngine.hh	1045
RenderEvents.hh	1046
Rendering.hh	1047
RenderTypes.hh	1048
RFIDSensor.hh	1050
RFIDTag.hh	1050
RFIDTagVisual.hh	1051
RFIDVisual.hh	1052
Road.hh	1052
Road2d.hh	1054
RotationSpline.hh	1054
RTShaderSystem.hh	1055
Scene.hh	1056
ScrewJoint.hh	1057
sdf.hh	1058
SDF.hh	1059
SelectionObj.hh	1060
Sensor.hh	1061
SensorFactory.hh	1062
SensorManager.hh	1063
Sensors.hh	1064
SensorTypes.hh	
Forward declarations and typedefs for sensors	1066

Server.hh	1067
Shape.hh	1068
SingletonT.hh	1070
Skeleton.hh	1070
SkeletonAnimation.hh	1072
SliderJoint.hh	1073
SphereShape.hh	1075
Spline.hh	1076
State.hh	1077
STLLoader.hh	1078
SubscribeOptions.hh	1080
Subscriber.hh	1081
SubscriptionTransport.hh	1083
SurfaceParams.hh	1084
SystemPaths.hh	1086
Time.hh	1087
Timer.hh	1088
TopicManager.hh	1089
Transport.hh	1091
TransportTypes.hh	
Forward declarations for transport	1093
TrimeshShape.hh	1095
UniversalJoint.hh	1096
UpdateInfo.hh	1097
UserCamera.hh	1098
UtilTypes.hh	1098
Vector2d.hh	1099
Vector2i.hh	1100
Vector3.hh	1101
Vector4.hh	1102
Video.hh	1104
VideoVisual.hh	1105
ViewController.hh	1106
Visual.hh	1107
WindowManager.hh	1108
WireBox.hh	1108
World.hh	1109
WorldState.hh	1111

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::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 happend and represent programming bugs.
- class **gazebo::common::KeyFrame**

- A* (p. 109) key frame in an animation.
- class **gazebo::common::Material**
 - Encapsulates description of a material.*
- class **gazebo::common::Mesh**
 - A* (p. 109) 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* (p. 109) plugin with access to **physics::Model** (p. 480).
- 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. 537) *Skeleton.hh* (p. 1070) *common/common.hh*
- class **gazebo::common::NumericAnimation**
 - A* (p. 109) numeric animation.
- class **gazebo::common::NumericKeyFrame**
 - A* (p. 109) keyframe for a **NumericAnimation** (p. 542).
- class **gazebo::common::PID**
 - Generic PID* (p. 571) controller class.
- class **gazebo::PluginT < T >**
 - A* (p. 109) class which all plugins must inherit from.
- class **gazebo::common::PoseAnimation**
 - A* (p. 109) pose animation.
- class **gazebo::common::PoseKeyFrame**
 - A* (p. 109) keyframe for a **PoseAnimation** (p. 593).
- class **gazebo::SensorPlugin**
 - A* (p. 109) plugin with access to **physics::Sensor**.
- class **SingletonT < T >**
 - Singleton template class.*
- class **gazebo::common::Skeleton**
 - A* (p. 109) skeleton.
- class **gazebo::common::SkeletonAnimation**
 - Skeleton* (p. 713) animation.
- class **gazebo::common::SkeletonNode**
 - A* (p. 109) skeleton node.
- class **gazebo::common::STLLoader**
 - Class used to load STL mesh files.*

- class **gazebo::common::SubMesh**
A (p. 109) child mesh.
- class **gazebo::common::SystemPaths**
Functions to handle getting system paths, keeps track of:
- class **gazebo::SystemPlugin**
A (p. 109) plugin loaded within the gzserver on startup.
- class **gazebo::common::Time**
A (p. 109) **Time** (p. 776) class, can be used to hold wall- or sim-time.
- class **gazebo::common::Timer**
A (p. 109) 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 (p. 109) plugin loaded within the gzserver on startup.
- class **gazebo::WorldPlugin**
A (p. 109) plugin with access to **physics::World** (p. 893).

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. 769)*
- 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, bool _searchLocalPath=true)
*search for file in **common::SystemPaths** (p. 769)*
- std::string **gazebo::common::find_file_path** (const std::string &_file)
*search for a file in **common::SystemPaths** (p. 769)*
- std::string **gazebo::common::ModelDatabase::GetDBConfig** (const std::string &_uri)
Return the database.config file as a string.
- std::string **gazebo::common::ModelDatabase::GetManifest** (const std::string &_uri) **GAZEBO_DEPRECATED**
Deprecated.
- 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.
- 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.
- 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.

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.

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::event::Events::ConnectWorldUpdateStart(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::physics::ScrewJoint< T >::Load(), sdf::ParamT< std::string >::Set(), and sdf::ParamT< std::string >::Update().

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.

Referenced by sdf::ParamT< std::string >::Set().

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::Joint::GetLowerLimit()`, and `gazebo::physics::Joint::GetUpperLimit()`.

8.1.3 Enumeration Type Documentation

8.1.3.1 enum gazebo::PluginType

Used to specify the type of plugin.

Enumerator:

WORLD_PLUGIN A (p. 109) World plugin.

MODEL_PLUGIN A (p. 109) Model plugin.

SENSOR_PLUGIN A (p. 109) Sensor plugin.

SYSTEM_PLUGIN A (p. 109) System plugin.

VISUAL_PLUGIN A (p. 109) 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. 769)

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

in	<i>_lbl</i>	Text label
in	<i>_file</i>	File containing the error
in	<i>_line</i>	Line containing the error
in	<i>_color</i>	Color (p. 205) 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>_lbl</code>	Text label
<code>in</code>	<code>_color</code>	Color (p. 205) 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>_path</code>	Path to a model.
-----------------	--------------------	------------------

8.1.4.6 `std::string gazebo::common::find_file (const std::string & _file, bool _searchLocalPath = true)`

search for file in **common::SystemPaths** (p. 769)

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.

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

search for a file in **common::SystemPaths** (p. 769)

Parameters

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

Returns

The path containing the file

8.1.4.8 `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.

8.1.4.9 `std::string gazebo::common::ModelDatabase::GetManifest (const std::string & _uri)`

Deprecated.

See Also

ModelDatabase::GetModelConfig (p. 34)

ModelDatabase::GetDBConfig (p. 34)

8.1.4.10 `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.

8.1.4.11 `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

in	_uri	The URI of the model
----	------	----------------------

Returns

The full path and filename to the SDF file

8.1.4.12 `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

in	_uri	the model uri
----	------	---------------

Returns

the model's name.

8.1.4.13 `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

8.1.4.14 `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. 494) to download the model list.

Returns

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

8.1.4.15 `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. 35)

Parameters

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

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

Returns the the global model database URI.

Returns

the URI.

8.1.4.17 `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.

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

Load the message parameters.

8.1.4.19 `std::ofstream& gazebo::common::Console::Log ()`

Use this to output a colored message to the terminal.

Parameters

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

Returns

Reference to an output stream

8.1.4.20 `void gazebo::common::Console::SetQuiet (bool _q)`

Set quiet output.

Parameters

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

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. 357).

8.2 Events

Namespaces

- namespace **gazebo::event**
Event (p. 288) namespace.

Classes

- class **gazebo::event::Connection**
A (p. 109) class that encapsulates a connection.
- class **gazebo::event::Event**
Base class for all events.
- class **gazebo::event::Events**
*An **Event** (p. 288) class to get notifications for simulator events.*
- class **gazebo::event::EventT< T >**
A (p. 109) 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	_subscriber	Pointer to a callback function
----	-------------	--------------------------------

Returns

A (p. 109) **Connection** (p. 218) object, which will automatically call Disconnect when it goes out of scope

Parameters

in	_subscriber	the subscriber to connect
----	-------------	---------------------------

Referenced by gazebo::event::Events::ConnectAddEntity(), gazebo::physics::Collision::ConnectContact(), 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::ConnectStep(), gazebo::event::Events::ConnectStop(), gazebo::transport::Connection::ConnectToShutdown(), gazebo::sensors::Sensor::ConnectUpdated(), gazebo::event::Events::ConnectWorldCreated(), gazebo::event::Events::ConnectWorldUpdateBegin(), gazebo::event::Events::ConnectWorldUpdateEnd(), and gazebo::event::Events::ConnectWorldUpdateStart().

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. 288).

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	_c	The connection to disconnect
in	_c	the connection

Implements **gazebo::event::Event** (p. 290).

References gazebo::event::Connection::GetId(), and NULL.

Referenced by gazebo::event::Events::DisconnectAddEntity(), gazebo::physics::Collision::DisconnectContact(), 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::Disconnect

NewImageFrame(), 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::DisconnectStep(), gazebo::event::Events::DisconnectStop(), 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

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

Implements **gazebo::event::Event** (p. 290).

8.3 Classes for physics and dynamics

Files

- file **PhysicsTypes.hh**
default namespace for gazebo

Namespaces

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

Classes

- class **gazebo::physics::Actor**
Actor (p. 109) class enables GPU based mesh model / skeleton scriptable animation.
- class **gazebo::physics::BallJoint**< T >
Base (p. 135) class for a ball joint.
- class **gazebo::physics::Base**
Base (p. 135) class for most physics classes.
- class **gazebo::physics::BoxShape**
Box geometry primitive.
- class **gazebo::physics::Collision**
Base (p. 135) class for all collision entities.
- class **gazebo::physics::CollisionState**
*Store state information of a **physics::Collision** (p. 192) object.*
- class **gazebo::physics::Contact**
A (p. 109) 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. 135) class for all physics objects in Gazebo.
- class **gazebo::physics::Gripper**
A (p. 109) gripper abstraction.
- class **gazebo::physics::HeightmapShape**
HeightmapShape (p. 348) collision shape builds a heightmap from an image.
- class **gazebo::physics::Hinge2Joint**< T >
A (p. 109) two axis hinge joint.
- class **gazebo::physics::HingeJoint**< T >
A (p. 109) single axis hinge joint.
- class **gazebo::physics::Inertial**
A (p. 109) class for inertial information about a link.
- class **gazebo::physics::Joint**
Base (p. 135) class for all joints.
- class **gazebo::physics::JointController**

- A* (p. 109) class for manipulating *physics::Joint* (p. 375).
- class **gazebo::physics::JointState**
 - keeps track of state of a physics::Joint* (p. 375)
- class **gazebo::physics::JointWrench**
 - Wrench information from a joint.*
- class **gazebo::physics::Link**
 - Link* (p. 412) 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. 412) object.
- class **Logplay**
 - Open and playback log files that were recorded using LogRecord.*
- class **gazebo::common::LogPlay**
- class **gazebo::physics::Model**
 - A* (p. 109) model is a collection of links, joints, and plugins.
- class **gazebo::physics::ModelState**
 - Store state information of a physics::Model* (p. 480) 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. 135) class for a physics engine.
- class **gazebo::physics::PhysicsFactory**
 - The physics factory instantiates different physics engines.*
- class **gazebo::physics::PlaneShape**
 - Collision* (p. 192) for an infinite plane.
- class **gazebo::physics::RayShape**
 - Base* (p. 135) class for Ray collision geometry.
- class **gazebo::physics::Road**
 - for building a Road* (p. 651) from SDF
- class **gazebo::physics::ScrewJoint** < T >
 - A* (p. 109) screw joint, which has both prismatic and rotational DOFs.
- class **gazebo::physics::Shape**
 - Base* (p. 135) class for all shapes.
- class **gazebo::physics::SliderJoint** < T >
 - A* (p. 109) slider joint.
- class **gazebo::physics::SphereShape**
 - Sphere collision shape.*
- class **gazebo::physics::State**
 - State* (p. 744) of an entity.
- class **gazebo::physics::SurfaceParams**
 - SurfaceParams* (p. 765) defines various Surface contact parameters.
- class **gazebo::physics::TrimeshShape**
 - Triangle mesh collision shape.*
- class **gazebo::physics::UniversalJoint** < T >
 - A* (p. 109) 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. 893) 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. 76).*
- WorldPtr **gazebo::physics::get_world** (const std::string &_name="")
Returns a pointer to a world by name.
- 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. 774)'s and call **gazebo::transport::init** (p. 77).*
- void **gazebo::physics::load_world** (WorldPtr _world, sdf::ElementPtr _sdf)
*Load world from **sdf::Element** (p. 269) pointer.*
- void **gazebo::physics::load_worlds** (sdf::ElementPtr _sdf)
*load multiple worlds from single **sdf::Element** (p. 269) pointer*
- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)
*Pause world by calling **World::SetPaused** (p. 903).*
- 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)
*Run world by calling **World::Run()** (p. 903) given a pointer to it.*
- void **gazebo::physics::run_worlds** ()
run multiple worlds stored in static variable gazebo::g_worlds
- void **gazebo::physics::stop_world** (WorldPtr _world)
*Stop world by calling **World::Stop()** (p. 904) given a pointer to it.*
- void **gazebo::physics::stop_worlds** ()
stop multiple worlds stored in static variable gazebo::g_worlds

Variables

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

8.3.1 Detailed Description

8.3.2 Macro Definition Documentation

8.3.2.1 #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

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

8.3.3 Typedef Documentation

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

8.3.4 Function Documentation

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

Create a world given a name.

Parameters

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

Returns

Pointer to the new world.

8.3.4.2 bool gazebo::physics::fini ()

Finalize transport by calling **gazebo::transport::fini** (p. 76).

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

Returns a pointer to a world by name.

Parameters

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

Returns

Pointer to the world.

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

Init world given a pointer to it.

Parameters

in	<code>_world</code>	World (p. 893) to initialize.
----	---------------------	--------------------------------------

8.3.4.5 void gazebo::physics::init_worlds ()

initialize multiple worlds stored in static variable gazebo::g_worlds

8.3.4.6 bool gazebo::physics::load ()

Setup **gazebo::SystemPlugin** (p. 774)'s and call **gazebo::transport::init** (p. 77).

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

Load world from **sdf::Element** (p. 269) pointer.

Parameters

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

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

load multiple worlds from single **sdf::Element** (p. 269) pointer

Parameters

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

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

Pause world by calling **World::SetPaused** (p. 903).

Parameters

in	<code>_world</code>	World (p. 893) to pause or unpause.
in	<code>_pause</code>	True to pause, False to unpause.

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

pause multiple worlds stored in static variable gazebo::g_worlds

Parameters

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

8.3.4.11 void gazebo::physics::remove_worlds ()

remove multiple worlds stored in static variable gazebo::g_worlds

8.3.4.12 void gazebo::physics::run_world (WorldPtr *_world*)

Run world by calling **World::Run()** (p. 903) given a pointer to it.

Parameters

in	<i>_world</i>	World (p. 893) to run.
----	---------------	-------------------------------

8.3.4.13 void gazebo::physics::run_worlds ()

run multiple worlds stored in static variable gazebo::g_worlds

8.3.4.14 void gazebo::physics::stop_world (WorldPtr *_world*)

Stop world by calling **World::Stop()** (p. 904) given a pointer to it.

Parameters

in	<i>_world</i>	World (p. 893) to stop.
----	---------------	--------------------------------

8.3.4.15 void gazebo::physics::stop_worlds ()

stop multiple worlds stored in static variable gazebo::g_worlds

8.3.5 Variable Documentation**8.3.5.1 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.

8.4 Math

A (p. 109) 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** (p. 109) 3x3 matrix class.*
- class **gazebo::math::Matrix4**
***A** (p. 109) 3x3 matrix class.*
- class **gazebo::math::Plane**
***A** (p. 109) plane and related functions.*
- class **gazebo::math::Pose**
Encapsulates a position and rotation in three space.
- class **gazebo::math::Quaternion**
***A** (p. 109) quaternion class.*
- class **gazebo::math::Rand**
Random number generator class.
- class **gazebo::math::RotationSpline**
***Spline** (p. 740) 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. 839) 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)

8.4.1 Detailed Description

A (p. 109) set of classes that encapsulate math related properties and functions.

8.4.2 Function Documentation

8.4.2.1 `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()`.

8.4.2.2 `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()`, and `gazebo::math::Quaternion::GetInverse()`.

8.4.2.3 `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()`.

8.4.2.4 `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()`.

8.4.2.5 `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

8.4.2.6 `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()`.

8.4.2.7 `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

8.4.2.8 `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().

8.4.2.9 `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.

8.4.2.10 `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.

8.4.2.11 `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

8.4.2.12 `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

8.4.3 Variable Documentation

8.4.3.1 `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()`.

8.4.3.2 `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()`.

8.5 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. 89).*
- class **gazebo::msgs::MsgFactory**
A (p. 109) 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. 839) to a **msgs::Vector3d**.*
- **msgs::Quaternion gazebo::msgs::Convert** (const math::Quaternion &_q)
*Convert a **math::Quaternion** (p. 609) to a **msgs::Quaternion**.*
- **msgs::Pose gazebo::msgs::Convert** (const math::Pose &_p)
*Convert a **math::Pose** (p. 584) to a **msgs::Pose**.*
- **msgs::Color gazebo::msgs::Convert** (const common::Color &_c)
*Convert a **common::Color** (p. 205) to a **msgs::Color**.*
- **msgs::Time gazebo::msgs::Convert** (const common::Time &_t)
*Convert a **common::Time** (p. 776) to a **msgs::Time**.*
- **msgs::PlaneGeom gazebo::msgs::Convert** (const math::Plane &_p)
*Convert a **math::Plane** (p. 575) 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. 609).*
- **math::Pose gazebo::msgs::Convert** (const msgs::Pose &_p)
*Convert a **msgs::Pose** to a **math::Pose** (p. 584).*
- **common::Color gazebo::msgs::Convert** (const msgs::Color &_c)
*Convert a **msgs::Color** to a **common::Color** (p. 205).*
- **common::Time gazebo::msgs::Convert** (const msgs::Time &_t)
*Convert a **msgs::Time** to a **common::Time** (p. 776).*
- **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::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::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. 356).
- `void gazebo::msgs::Set` (`msgs::Image *_msg, const common::Image &_i`)
Set a `msgs::Image` from a `common::Image` (p. 356).
- `void gazebo::msgs::Set` (`msgs::Vector3d *_pt, const math::Vector3 &_v`)
Set a `msgs::Vector3d` from a `math::Vector3` (p. 839).
- `void gazebo::msgs::Set` (`msgs::Vector2d *_pt, const math::Vector2d &_v`)
Set a `msgs::Vector2d` from a `math::Vector3` (p. 839).
- `void gazebo::msgs::Set` (`msgs::Quaternion *_q, const math::Quaternion &_v`)
Set a `msgs::Quaternion` from a `math::Quaternion` (p. 609).
- `void gazebo::msgs::Set` (`msgs::Pose *_p, const math::Pose &_v`)
Set a `msgs::Pose` from a `math::Pose` (p. 584).
- `void gazebo::msgs::Set` (`msgs::Color *_c, const common::Color &_v`)
Set a `msgs::Color` from a `common::Color` (p. 205).
- `void gazebo::msgs::Set` (`msgs::Time *_t, const common::Time &_v`)
Set a `msgs::Time` from a `common::Time` (p. 776).
- `void gazebo::msgs::Set` (`msgs::PlaneGeom *_p, const math::Plane &_v`)
Set a `msgs::Plane` from a `math::Plane` (p. 575).
- `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.

8.5.1 Detailed Description

All messages and helper functions.

8.5.2 Macro Definition Documentation

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

Value:

```
google::protobuf::Message* New##_classname() \
{ \
    return 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.

8.5.3 Function Documentation

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

Convert a **math::Vector3** (p. 839) to a msgs::Vector3d.

Parameters

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

Returns

A (p. 109) msgs::Vector3d object

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

Convert a **math::Quaternion** (p. 609) to a msgs::Quaternion.

Parameters

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

Returns

A (p. 109) msgs::Quaternion object

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

Convert a **math::Pose** (p. 584) to a `msgs::Pose`.

Parameters

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

Returns

A (p. 109) `msgs::Pose` object

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

Convert a **common::Color** (p. 205) to a `msgs::Color`.

Parameters

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

Returns

A (p. 109) `msgs::Color` object

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

Convert a **common::Time** (p. 776) to a `msgs::Time`.

Parameters

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

Returns

A (p. 109) `msgs::Time` object

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

Convert a **math::Plane** (p. 575) to a `msgs::PlaneGeom`.

Parameters

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

Returns

A (p. 109) `msgs::PlaneGeom` object

8.5.3.7 `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 (p. 109) **math::Vector3** (p. 839) object

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

Convert a `msgs::Quaternion` to a **math::Quaternion** (p. 609).

Parameters

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

Returns

A (p. 109) **math::Quaternion** (p. 609) object

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

Convert a `msgs::Pose` to a **math::Pose** (p. 584).

Parameters

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

Returns

A (p. 109) **math::Pose** (p. 584) object

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

Convert a `msgs::Color` to a **common::Color** (p. 205).

Parameters

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

Returns

A (p. 109) **common::Color** (p. 205) object

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

Convert a `msgs::Time` to a **common::Time** (p. 776).

Parameters

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

Returns

A (p. 109) **common::Time** (p. 776) object

8.5.3.12 `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 (p. 109) `common::Plane` object

8.5.3.13 `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 (p. 109) Request message

8.5.3.14 `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

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

Get the header from a protobuf message.

Parameters

in	<i>_message</i>	A (p. 109) google protobuf message
----	-----------------	------------------------------------

Returns

A (p. 109) pointer to the message's header

8.5.3.16 msgs::GUI gazebo::msgs::GUIFromSDF (sdf::ElementPtr _sdf)

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

Parameters

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

Returns

The new msgs::GUI object

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

Initialize a message.

Parameters

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

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

8.5.3.18 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

8.5.3.19 msgs::Scene gazebo::msgs::SceneFromSDF (sdf::ElementPtr _sdf)

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

Parameters

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

Returns

The new `msgs::Scene` object

8.5.3.20 `void gazebo::msgs::Set (common::Image & _img, const msgs::Image & _msg)`

Convert a `msgs::Image` to a **`common::Image`** (p. 356).

Parameters

out	<code>_img</code>	The <code>common::Image</code> (p. 356) container
in	<code>_msg</code>	The Image message to convert

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

Set a `msgs::Image` from a **`common::Image`** (p. 356).

Parameters

out	<code>_msg</code>	A (p. 109) <code>msgs::Image</code> pointer
in	<code>_i</code>	A (p. 109) <code>common::Image</code> (p. 356) reference

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

Set a `msgs::Vector3d` from a **`math::Vector3`** (p. 839).

Parameters

out	<code>_pt</code>	A (p. 109) <code>msgs::Vector3d</code> pointer
in	<code>_v</code>	A (p. 109) <code>math::Vector3</code> (p. 839) reference

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

Set a `msgs::Vector2d` from a **`math::Vector3`** (p. 839).

Parameters

out	<code>_pt</code>	A (p. 109) <code>msgs::Vector2d</code> pointer
in	<code>_v</code>	A (p. 109) <code>math::Vector2d</code> (p. 822) reference

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

Set a `msgs::Quaternion` from a **`math::Quaternion`** (p. 609).

Parameters

out	<code>_q</code>	A (p. 109) <code>msgs::Quaternion</code> pointer
in	<code>_v</code>	A (p. 109) <code>math::Quaternion</code> (p. 609) reference

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

Set a `msgs::Pose` from a `math::Pose` (p. 584).

Parameters

out	<code>_p</code>	A (p. 109) <code>msgs::Pose</code> pointer
in	<code>_v</code>	A (p. 109) <code>math::Pose</code> (p. 584) reference

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

Set a `msgs::Color` from a `common::Color` (p. 205).

Parameters

out	<code>_p</code>	A (p. 109) <code>msgs::Color</code> pointer
in	<code>_v</code>	A (p. 109) <code>common::Color</code> (p. 205) reference

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

Set a `msgs::Time` from a `common::Time` (p. 776).

Parameters

out	<code>_p</code>	A (p. 109) <code>msgs::Time</code> pointer
in	<code>_v</code>	A (p. 109) <code>common::Time</code> (p. 776) reference

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

Set a `msgs::Plane` from a `math::Plane` (p. 575).

Parameters

out	<code>_p</code>	A (p. 109) <code>msgs::Plane</code> pointer
in	<code>_v</code>	A (p. 109) <code>math::Plane</code> (p. 575) reference

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

Time stamp a header.

Parameters

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

8.5.3.30 void gazebo::msgs::Stamp (msgs::Time * *_time*)

Set the time in a time message.

Parameters

in	<i>_time</i>	A (p. 109) Time message
----	--------------	-------------------------

8.5.3.31 msgs::TrackVisual gazebo::msgs::TrackVisualFromSDF (sdf::ElementPtr *_sdf*)

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

Parameters

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

Returns

The new msgs::TrackVisual object

8.5.3.32 msgs::Visual gazebo::msgs::VisualFromSDF (sdf::ElementPtr *_sdf*)

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

Parameters

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

Returns

The new msgs::Visual object

8.6 Rendering

A (p. 109) 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. 243) Conversions.hh (p. 948) rendering/Conversions.hh (p. 948).
- 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 (p. 109) 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 (p. 109) 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. 106)'s Run-Time Shader system.*
- class **gazebo::rendering::Scene**
Representation of an entire scene graph.
- class **gazebo::rendering::SelectionObj**
A (p. 109) graphical selection object.
- class **gazebo::rendering::UserCamera**
A (p. 109) camera used for user visualization of a scene.
- class **gazebo::rendering::VideoVisual**
A (p. 109) visual element that displays a video as a texture.
- class **gazebo::rendering::ViewController**
Base class for view controllers.
- class **gazebo::rendering::Visual**
A (p. 109) renderable object.
- class **gazebo::rendering::WindowManager**
Class to manage render windows.
- class **gazebo::rendering::WireBox**
Draws a wireframe box.

Functions

- rendering::ScenePtr **gazebo::rendering::create_scene** (const std::string &_name, bool _enableVisualizations)

*create **rendering::Scene** (p. 662) 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. 662) 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. 662) by name*

8.6.1 Detailed Description

A (p. 109) set of rendering related class, functions, and definitions.

8.6.2 Function Documentation

8.6.2.1 rendering::ScenePtr gazebo::rendering::create_scene (const std::string &_name, bool _enableVisualizations)

create **rendering::Scene** (p. 662) by name.

Parameters

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

8.6.2.2 bool gazebo::rendering::fini ()

teardown rendering engine.

8.6.2.3 rendering::ScenePtr gazebo::rendering::get_scene (const std::string &_name)

get pointer to **rendering::Scene** (p. 662) by name.

Parameters

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

8.6.2.4 bool gazebo::rendering::init ()

init rendering engine.

8.6.2.5 bool gazebo::rendering::load ()

load rendering engine.

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

remove a **rendering::Scene** (p. 662) by name

Parameters

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

8.7 Gazebo_parser

Namespaces

- namespace **sdf**
namespace for Simulation Description Format parser
- namespace **urdf2gazebo**
namespace for URDF to SDF parser

Classes

- class **A**
holding gazebo extension elements in urdf
- class **sdf::Element**
SDF (p. 680) Element (p. 269) class.
- class **urdf2gazebo::GazeboExtension**
- class **sdf::SDF**
Base SDF (p. 680) class.
- class **urdf2gazebo::URDF2Gazebo**

Typedefs

- typedef const urdf::Link * **urdf2gazebo::ConstUrdfLinkPtr**
- typedef urdf::Collision * **urdf2gazebo::UrdfCollisionPtr**
- typedef urdf::Link * **urdf2gazebo::UrdfLinkPtr**
- typedef urdf::Visual * **urdf2gazebo::UrdfVisualPtr**

8.7.1 Detailed Description

8.7.2 Typedef Documentation

8.7.2.1 typedef const urdf::Link* urdf2gazebo::ConstUrdfLinkPtr

8.7.2.2 typedef urdf::Collision* urdf2gazebo::UrdfCollisionPtr

8.7.2.3 typedef urdf::Link* urdf2gazebo::UrdfLinkPtr

8.7.2.4 typedef urdf::Visual* urdf2gazebo::UrdfVisualPtr

8.8 Sensors

A (p. 109) 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::GpuRaySensor**
- class **gazebo::sensors::ImuSensor**
An IMU sensor.
- class **gazebo::sensors::MultiCameraSensor**
Multiple camera sensor.
- class **gazebo::sensors::RaySensor**
Sensor (p. 683) with one or more rays.
- class **gazebo::sensors::RFIDSensor**
Sensor (p. 683) class for RFID type of sensor.
- class **gazebo::sensors::RFIDTag**
RFIDTag (p. 646) 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.

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)`
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 ()` **GAZEBO_DEPRECATED**
Deprecated.
- `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 (p. 109) 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)`

Create a sensor using SDF.

Parameters

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

Returns

The name of the new sensor.

8.8.3.2 `bool gazebo::sensors::fini ()`

shutdown the sensor generation loop.

Returns

True if successfully finalized, false if not

8.8.3.3 `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 <code>_name = world_name::model_name::link_name::sensor_name</code> . 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.

8.8.3.4 `bool gazebo::sensors::init ()`

initialize the sensor generation loop.

Returns

True if successfully initialized, false if not

8.8.3.5 bool gazebo::sensors::load ()

Load the sensor library.

Returns

True if successfully loaded, false if not.

8.8.3.6 void gazebo::sensors::remove_sensor (const std::string & _sensorName)

Remove a sensor by name.

Parameters

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

8.8.3.7 bool gazebo::sensors::remove_sensors ()

Remove all sensors.

Returns

True if all successfully removed, false if not

8.8.3.8 void gazebo::sensors::run ()

Deprecated.

See Also

run_threads (p. 72)

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

Run the sensor generation one step.

Parameters

<code>_force,:</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 (p. 109) 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 (p. 109) node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.
- class **gazebo::transport::Publication**
A (p. 109) publication for a topic.
- class **gazebo::transport::PublicationTransport**
transport/transport.hh
- class **gazebo::transport::Publisher**
A (p. 109) 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 (p. 109) subscriber to a topic.
- class **gazebo::transport::SubscriptionTransport**
transport/transport.hh
- class **gazebo::transport::TopicManager**
Manages topics and their subscriptions.

Typedefs

- typedef CallbackHelper * **gazebo::transport::CallbackHelperPtr**
*boost shared pointer to **transport::CallbackHelper** (p. 155)*

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.
- 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.
- 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::stop** ()
Stop the transport component from running.

8.9.1 Detailed Description

Handles transportation of messages.

8.9.2 Typedef Documentation

8.9.2.1 typedef CallbackHelper* gazebo::transport::CallbackHelperPtr

boost shared pointer to **transport::CallbackHelper** (p. 155)

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

Returns

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

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

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

Parameters

out	<i>_namespaces</i>	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 (p. 109) 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	<i>_msgType</i>	Type of message to filter the result on. If empty, then a list of all the topics is returned.
----	-----------------	---

Returns

A (p. 109) list of the advertised topics that publish messages of the type specified by `_msgType`.

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

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

Parameters

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

Returns

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

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

Initialize the transport system.

Parameters

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

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

Is the transport system stopped?

Returns

true if the transport system is stopped; false otherwise

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

Pause or unpaue incoming messages.

When paused, messages are queued for later delivery

Parameters

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

8.9.3.11 `msgs::Response* gazebo::transport::request (const std::string & _worldName, const std::string & _request, const std::string & _data = " ")`

Send a request and receive a response.

This call will block until a response is received.

Parameters

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

Returns

The response to the request. Can be empty.

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

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

8.9.3.14 `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

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

Stop the transport component from running.

8.10 Utility

Files

- file **UtilTypes.hh**

Classes

- class **gazebo::util::DiagnosticManager**
A (p. 109) diagnostic manager class.
- class **gazebo::util::DiagnosticTimer**
A (p. 109) timer designed for diagnostics.

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. 288) 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 (p. 109) 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 (p. 109) plugin with access to **physics::Model** (p. 480).*

- class **PluginT**
A (p. 109) class which all plugins must inherit from.
- class **SensorPlugin**
A (p. 109) plugin with access to `physics::Sensor`.
- class **Server**
- class **SystemPlugin**
A (p. 109) plugin loaded within the gzserver on startup.
- class **VisualPlugin**
A (p. 109) plugin loaded within the gzserver on startup.
- class **WorldPlugin**
A (p. 109) plugin with access to `physics::World` (p. 893).

Typedefs

- typedef GUIPlugin * **GUIPluginPtr**
- typedef ModelPlugin * **ModelPluginPtr**
- typedef SensorPlugin * **SensorPluginPtr**
- typedef SystemPlugin * **SystemPluginPtr**
- typedef VisualPlugin * **VisualPluginPtr**
- typedef 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 GUIPlugin* gazebo::GUIPluginPtr

9.2.2.2 typedef ModelPlugin* gazebo::ModelPluginPtr

9.2.2.3 typedef SensorPlugin* gazebo::SensorPluginPtr

9.2.2.4 typedef SystemPlugin* gazebo::SystemPluginPtr

9.2.2.5 typedef VisualPlugin* gazebo::VisualPluginPtr

9.2.2.6 typedef 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 **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 **KeyFrame**
 - A (p. 109) key frame in an animation.*
- class **LogPlay**
- class **LogRecord**
 - addtogroup gazebo_common*
- class **Material**
 - Encapsulates description of a material.*
- class **Mesh**
 - A (p. 109) 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. 537) Skeleton.hh (p. 1070) common/common.hh*
- class **NumericAnimation**
 - A (p. 109) numeric animation.*
- class **NumericKeyFrame**
 - A (p. 109) keyframe for a NumericAnimation (p. 542).*
- class **PID**
 - Generic PID (p. 571) controller class.*
- class **PoseAnimation**
 - A (p. 109) pose animation.*
- class **PoseKeyFrame**
 - A (p. 109) keyframe for a PoseAnimation (p. 593).*
- class **Skeleton**
 - A (p. 109) skeleton.*
- class **SkeletonAnimation**

- *Skeleton* (p. 713) animation.
- class **SkeletonNode**
 - *A* (p. 109) skeleton node.
- class **STLLoader**
 - *Class used to load STL mesh files.*
- class **SubMesh**
 - *A* (p. 109) child mesh.
- class **SystemPaths**
 - *Functions to handle getting system paths, keeps track of:*
- class **Time**
 - *A* (p. 109) **Time** (p. 776) class, can be used to hold wall- or sim-time.
- class **Timer**
 - *A* (p. 109) 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 **Animation** * **AnimationPtr**
- typedef DiagnosticTimer * **DiagnosticTimerPtr**
- typedef std::map< unsigned int, **SkeletonNode** * > **NodeMap**
- typedef std::map< unsigned int, **SkeletonNode** * >::iterator **NodeMapIter**
- typedef **NumericAnimation** * **NumericAnimationPtr**
- typedef std::vector< common::Param * > **Param_V**
- typedef **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 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. 769)*
- std::string **find_file** (const std::string &_file, bool _searchLocalPath=true)
 - *search for file in **common::SystemPaths** (p. 769)*
- std::string **find_file_path** (const std::string &_file)
 - *search for a file in **common::SystemPaths** (p. 769)*

Variables

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

9.3.1 Detailed Description

Common namespace.

9.3.2 Typedef Documentation

9.3.2.1 typedef Animation* gazebo::common::AnimationPtr

9.3.2.2 typedef 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 NumericAnimation* gazebo::common::NumericAnimationPtr

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

9.3.2.7 typedef 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 std::map<std::string, std::string> gazebo::common::StrStr_M

9.4 gazebo::event Namespace Reference

Event (p. 288) namespace.

Classes

- class **Connection**
A (p. 109) class that encapsulates a connection.
- class **Event**
Base class for all events.
- class **Events**
*An **Event** (p. 288) class to get notifications for simulator events.*
- class **EventT**
A (p. 109) class for event processing.

Typedefs

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

9.4.1 Detailed Description

Event (p. 288) namespace.

9.4.2 Typedef Documentation

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

9.4.2.2 typedef 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 (p. 109) 3x3 matrix class.
- class **Matrix4**
A (p. 109) 3x3 matrix class.
- class **Plane**
A (p. 109) plane and related functions.
- class **Pose**
Encapsulates a position and rotation in three space.
- class **Quaternion**
A (p. 109) quaternion class.
- class **Rand**
Random number generator class.
- class **RotationSpline**
Spline (p. 740) for rotations.
- class **Spline**
Splines.
- class **Vector2d**
Generic double x, y vector.
- class **Vector2i**
Generic integer x, y vector.
- class **Vector3**

The **Vector3** (p. 839) 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 (p. 109) factory that generates protobuf message based on a string type.

Typedefs

- typedef
google::protobuf::Message **MsgFactoryFn** (*)()

Functions

- msgs::Vector3d **Convert** (const **math::Vector3** &_v)
*Convert a **math::Vector3** (p. 839) to a msgs::Vector3d.*
- msgs::Quaternion **Convert** (const **math::Quaternion** &_q)
*Convert a **math::Quaternion** (p. 609) to a msgs::Quaternion.*
- msgs::Pose **Convert** (const **math::Pose** &_p)
*Convert a **math::Pose** (p. 584) to a msgs::Pose.*
- msgs::Color **Convert** (const **common::Color** &_c)
*Convert a **common::Color** (p. 205) to a msgs::Color.*
- msgs::Time **Convert** (const **common::Time** &_t)
*Convert a **common::Time** (p. 776) to a msgs::Time.*
- msgs::PlaneGeom **Convert** (const **math::Plane** &_p)
*Convert a **math::Plane** (p. 575) 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. 609).*
- **math::Pose Convert** (const msgs::Pose &_p)
*Convert a msgs::Pose to a **math::Pose** (p. 584).*
- **common::Color Convert** (const msgs::Color &_c)
*Convert a msgs::Color to a **common::Color** (p. 205).*
- **common::Time Convert** (const msgs::Time &_t)
*Convert a msgs::Time to a **common::Time** (p. 776).*
- **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::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::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. 356).*

- void **Set** (msgs::Image *_msg, const **common::Image** &_i)
*Set a msgs::Image from a **common::Image** (p. 356).*
- void **Set** (msgs::Vector3d *_pt, const **math::Vector3** &_v)
*Set a msgs::Vector3d from a **math::Vector3** (p. 839).*
- void **Set** (msgs::Vector2d *_pt, const **math::Vector2d** &_v)
*Set a msgs::Vector2d from a **math::Vector3** (p. 839).*
- void **Set** (msgs::Quaternion *_q, const **math::Quaternion** &_v)
*Set a msgs::Quaternion from a **math::Quaternion** (p. 609).*
- void **Set** (msgs::Pose *_p, const **math::Pose** &_v)
*Set a msgs::Pose from a **math::Pose** (p. 584).*
- void **Set** (msgs::Color *_c, const **common::Color** &_v)
*Set a msgs::Color from a **common::Color** (p. 205).*
- void **Set** (msgs::Time *_t, const **common::Time** &_v)
*Set a msgs::Time from a **common::Time** (p. 776).*
- void **Set** (msgs::PlaneGeom *_p, const **math::Plane** &_v)
*Set a msgs::Plane from a **math::Plane** (p. 575).*
- 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 google::protobuf::Message*(* gazebo::msgs::MsgFactoryFn)()

9.7 gazebo::physics Namespace Reference

namespace for physics

Classes

- class **Actor**
***Actor** (p. 109) class enables GPU based mesh model / skeleton scriptable animation.*
- class **BallJoint**
***Base** (p. 135) class for a ball joint.*
- class **Base**
***Base** (p. 135) class for most physics classes.*
- class **BoxShape**

- Box geometry primitive.*
- class **Collision**
 - Base (p. 135) class for all collision entities.*
- class **CollisionState**
 - Store state information of a **physics::Collision** (p. 192) object.*
- class **Contact**
 - A (p. 109) contact between two collisions.*
- class **ContactManager**
 - Aggregates all the contact information generated by the collision detection engine.*
- class **CylinderShape**
 - Cylinder collision.*
- class **Entity**
 - Base (p. 135) class for all physics objects in Gazebo.*
- class **Gripper**
 - A (p. 109) gripper abstraction.*
- class **HeightmapShape**
 - HeightmapShape (p. 348) collision shape builds a heightmap from an image.*
- class **Hinge2Joint**
 - A (p. 109) two axis hinge joint.*
- class **HingeJoint**
 - A (p. 109) single axis hinge joint.*
- class **Inertial**
 - A (p. 109) class for inertial information about a link.*
- class **Joint**
 - Base (p. 135) class for all joints.*
- class **JointController**
 - A (p. 109) class for manipulating **physics::Joint** (p. 375).*
- class **JointState**
 - keeps track of state of a **physics::Joint** (p. 375)*
- class **JointWrench**
 - Wrench information from a joint.*
- class **Link**
 - Link (p. 412) 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. 412) object.*
- class **Model**
 - A (p. 109) model is a collection of links, joints, and plugins.*
- class **ModelState**
 - Store state information of a **physics::Model** (p. 480) object.*
- class **MultiRayShape**
 - Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.*
- class **PhysicsEngine**
 - Base (p. 135) class for a physics engine.*
- class **PhysicsFactory**
 - The physics factory instantiates different physics engines.*
- class **PlaneShape**

- Collision* (p. 192) for an infinite plane.
- class **RayShape**
 - Base* (p. 135) class for Ray collision geometry.
- class **Road**
 - for building a **Road** (p. 651) from SDF
- class **ScrewJoint**
 - A* (p. 109) screw joint, which has both prismatic and rotational DOFs.
- class **Shape**
 - Base* (p. 135) class for all shapes.
- class **SliderJoint**
 - A* (p. 109) slider joint.
- class **SphereShape**
 - Sphere collision shape.*
- class **State**
 - State* (p. 744) of an entity.
- class **SurfaceParams**
 - SurfaceParams* (p. 765) defines various Surface contact parameters.
- struct **TrajectoryInfo**
- class **TrimeshShape**
 - Triangle mesh collision shape.*
- class **UniversalJoint**
 - A* (p. 109) 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. 893) object.*

Typedefs

- typedef std::vector< **ActorPtr** > **Actor_V**
- typedef **Actor** * **ActorPtr**
- typedef std::vector< **BasePtr** > **Base_V**
- typedef **Base** * **BasePtr**
- typedef **BoxShape** * **BoxShapePtr**
- typedef std::vector< **CollisionPtr** > **Collision_V**
- typedef **Collision** * **CollisionPtr**
- typedef **Contact** * **ContactPtr**
- typedef **CylinderShape** * **CylinderShapePtr**
- typedef **Entity** * **EntityPtr**
- typedef **HeightmapShape** * **HeightmapShapePtr**
- typedef **Inertial** * **InertialPtr**
- typedef std::vector< **JointPtr** > **Joint_V**
- typedef std::vector
< **JointControllerPtr** > **JointController_V**
- typedef **JointController** * **JointControllerPtr**
- typedef **Joint** * **JointPtr**
- typedef std::vector< **LinkPtr** > **Link_V**
- typedef **Link** * **LinkPtr**

- typedef MeshShape * **MeshShapePtr**
- typedef std::vector< **ModelPtr** > **Model_V**
- typedef **Model** * **ModelPtr**
- typedef **MultiRayShape** * **MultiRayShapePtr**
- typedef **PhysicsEngine** * **PhysicsEnginePtr**
- typedef **PhysicsEnginePtr**(* **PhysicsFactoryFn**)(WorldPtr world)
- typedef **RayShape** * **RayShapePtr**
- typedef **Road** * **RoadPtr**
- typedef **Shape** * **ShapePtr**
- typedef **SphereShape** * **SphereShapePtr**
- typedef **SurfaceParams** * **SurfaceParamsPtr**
- typedef **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. 76).
- **WorldPtr get_world** (const std::string &_name="")
Returns a pointer to a world by name.
- 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. 774)'s and call `gazebo::transport::init` (p. 77).
- void **load_world** (**WorldPtr** _world, **sdf::ElementPtr** _sdf)
Load world from `sdf::Element` (p. 269) pointer.
- void **load_worlds** (**sdf::ElementPtr** _sdf)
load multiple worlds from single `sdf::Element` (p. 269) pointer
- void **pause_world** (**WorldPtr** _world, bool _pause)
Pause world by calling `World::SetPaused` (p. 903).
- 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)
Run world by calling `World::Run()` (p. 903) given a pointer to it.
- void **run_worlds** ()
run multiple worlds stored in static variable `gazebo::g_worlds`
- void **stop_world** (**WorldPtr** _world)
Stop world by calling `World::Stop()` (p. 904) given a pointer to it.
- void **stop_worlds** ()
stop multiple worlds stored in static variable `gazebo::g_worlds`

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 Actor* gazebo::physics::ActorPtr

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

9.7.2.4 typedef Base* gazebo::physics::BasePtr

9.7.2.5 typedef BoxShape* gazebo::physics::BoxShapePtr

9.7.2.6 typedef std::vector<CollisionPtr> gazebo::physics::Collision_V

9.7.2.7 typedef Collision* gazebo::physics::CollisionPtr

9.7.2.8 typedef Contact* gazebo::physics::ContactPtr

9.7.2.9 typedef CylinderShape* gazebo::physics::CylinderShapePtr

9.7.2.10 typedef Entity* gazebo::physics::EntityPtr

9.7.2.11 typedef HeightmapShape* gazebo::physics::HeightmapShapePtr

9.7.2.12 typedef Inertial* gazebo::physics::InertialPtr

9.7.2.13 typedef std::vector<JointPtr> gazebo::physics::Joint_V

9.7.2.14 typedef std::vector<JointControllerPtr> gazebo::physics::JointController_V

9.7.2.15 typedef JointController* gazebo::physics::JointControllerPtr

9.7.2.16 typedef Joint* gazebo::physics::JointPtr

9.7.2.17 typedef std::vector<LinkPtr> gazebo::physics::Link_V

9.7.2.18 typedef Link* gazebo::physics::LinkPtr

9.7.2.19 typedef MeshShape* gazebo::physics::MeshShapePtr

- 9.7.2.20 `typedef std::vector<ModelPtr> gazebo::physics::Model_V`
- 9.7.2.21 `typedef Model* gazebo::physics::ModelPtr`
- 9.7.2.22 `typedef MultiRayShape* gazebo::physics::MultiRayShapePtr`
- 9.7.2.23 `typedef PhysicsEngine* gazebo::physics::PhysicsEnginePtr`
- 9.7.2.24 `typedef RayShape* gazebo::physics::RayShapePtr`
- 9.7.2.25 `typedef Road* gazebo::physics::RoadPtr`
- 9.7.2.26 `typedef Shape* gazebo::physics::ShapePtr`
- 9.7.2.27 `typedef SphereShape* gazebo::physics::SphereShapePtr`
- 9.7.2.28 `typedef SurfaceParams* gazebo::physics::SurfaceParamsPtr`
- 9.7.2.29 `typedef 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. 243) [Conversions.hh](#) (p. 948) [rendering/Conversions.hh](#) (p. 948).
- class **DepthCamera**
Depth camera used to render depth data into an image buffer.
- 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 (p. 109) 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 (p. 109) 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. 106)'s Run-Time Shader system.*
- class **Scene**
Representation of an entire scene graph.
- class **SelectionObj**
A (p. 109) graphical selection object.
- class **UserCamera**
A (p. 109) camera used for user visualization of a scene.
- class **VideoVisual**
A (p. 109) visual element that displays a video as a texture.
- class **ViewController**
Base class for view controllers.
- class **Visual**
A (p. 109) renderable object.
- class **WindowManager**
Class to manage render windows.
- class **WireBox**
Draws a wireframe box.

Typedefs

- typedef **ArrowVisual** * **ArrowVisualPtr**
- typedef **AxisVisual** * **AxisVisualPtr**
- typedef **Camera** * **CameraPtr**
- typedef **CameraVisual** * **CameraVisualPtr**
- typedef **COMVisual** * **COMVisualPtr**
- typedef **ContactVisual** * **ContactVisualPtr**
- typedef **DepthCamera** * **DepthCameraPtr**
- typedef **DynamicLines** * **DynamicLinesPtr**
- typedef **GpuLaser** * **GpuLaserPtr**
- typedef **JointVisual** * **JointVisualPtr**
- typedef **LaserVisual** * **LaserVisualPtr**
- typedef **Light** * **LightPtr**
- typedef **RFIDTagVisual** * **RFIDTagVisualPtr**
- typedef **RFIDVisual** * **RFIDVisualPtr**
- typedef **Scene** * **ScenePtr**
- typedef **UserCamera** * **UserCameraPtr**
- typedef **Visual** * **VisualPtr**

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)
*create **rendering::Scene** (p. 662) by name.*
- bool **fini** ()
teardown rendering engine.
- **rendering::ScenePtr get_scene** (const std::string &_name)
*get pointer to **rendering::Scene** (p. 662) by name.*
- bool **init** ()
init rendering engine.
- bool **load** ()
load rendering engine.
- void **remove_scene** (const std::string &_name)
*remove a **rendering::Scene** (p. 662) by name*

9.8.1 Detailed Description

Rendering namespace.

9.8.2 Typedef Documentation

9.8.2.1 typedef ArrowVisual* gazebo::rendering::ArrowVisualPtr

9.8.2.2 typedef AxisVisual* gazebo::rendering::AxisVisualPtr

9.8.2.3 typedef Camera* gazebo::rendering::CameraPtr

9.8.2.4 typedef CameraVisual* gazebo::rendering::CameraVisualPtr

9.8.2.5 typedef COMVisual* gazebo::rendering::COMVisualPtr

9.8.2.6 typedef ContactVisual* gazebo::rendering::ContactVisualPtr

9.8.2.7 typedef DepthCamera* gazebo::rendering::DepthCameraPtr

9.8.2.8 typedef DynamicLines* gazebo::rendering::DynamicLinesPtr

9.8.2.9 typedef GpuLaser* gazebo::rendering::GpuLaserPtr

9.8.2.10 typedef JointVisual* gazebo::rendering::JointVisualPtr

9.8.2.11 typedef LaserVisual* gazebo::rendering::LaserVisualPtr

9.8.2.12 typedef Light* gazebo::rendering::LightPtr

9.8.2.13 typedef RFIDTagVisual* gazebo::rendering::RFIDTagVisualPtr

9.8.2.14 typedef RFIDVisual* gazebo::rendering::RFIDVisualPtr

9.8.2.15 typedef Scene* gazebo::rendering::ScenePtr

9.8.2.16 typedef UserCamera* gazebo::rendering::UserCameraPtr

9.8.2.17 typedef Visual* gazebo::rendering::VisualPtr

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 (p. 109) list of points, 1 vertex per point.

RENDERING_LINE_LIST A (p. 109) list of lines, 2 vertices per line.

RENDERING_LINE_STRIP A (p. 109) strip of connected lines, 1 vertex per line plus 1 start vertex.

RENDERING_TRIANGLE_LIST A (p. 109) list of triangles, 3 vertices per triangle.

RENDERING_TRIANGLE_STRIP A (p. 109) strip of triangles, 3 vertices for the first triangle, and 1 per triangle after that.

RENDERING_TRIANGLE_FAN A (p. 109) 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 **GpuRaySensor**
- class **ImuSensor**
An IMU sensor.
- class **MultiCameraSensor**
Multiple camera sensor.
- class **RaySensor**
Sensor (p. 683) with one or more rays.
- class **RFIDSensor**
Sensor (p. 683) class for RFID type of sensor.
- class **RFIDTag**
RFIDTag (p. 646) to interact with RFIDTagSensors.
- class **Sensor**
Base class for sensors.
- class **SensorFactory**
- class **SensorManager**
Class to manage and update all sensors.
- class **SimTimeEvent**

- class **SimTimeEventHandler**
Monitors simulation time, and notifies conditions when a specified time has been reached.

Typedefs

- typedef std::vector
 < **CameraSensorPtr** > **CameraSensor_V**
- typedef **CameraSensor** * **CameraSensorPtr**
- typedef std::vector
 < **ContactSensorPtr** > **ContactSensor_V**
- typedef **ContactSensor** * **ContactSensorPtr**
- typedef std::vector
 < **DepthCameraSensorPtr** > **DepthCameraSensor_V**
- typedef **DepthCameraSensor** * **DepthCameraSensorPtr**
- typedef std::vector
 < **GpuRaySensorPtr** > **GpuRaySensor_V**

- typedef **GpuRaySensor** * **GpuRaySensorPtr**
- typedef std::vector< **ImuSensorPtr** > **ImuSensor_V**
- typedef **ImuSensor** * **ImuSensorPtr**
- typedef std::vector< **RaySensorPtr** > **RaySensor_V**
- typedef **RaySensor** * **RaySensorPtr**
- typedef std::vector< **RFIDSensor** > **RFIDSensor_V**
- typedef **RFIDSensor** * **RFIDSensorPtr**
- typedef std::vector< **RFIDTag** > **RFIDTag_V**
- typedef **RFIDTag** * **RFIDTagPtr**
- typedef std::vector< **SensorPtr** > **Sensor_V**
- typedef **Sensor** *(* **SensorFactoryFn**)()
- typedef **Sensor** * **SensorPtr**

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)

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** () **GAZEBO_DEPRECATED**

Deprecated.
- 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 CameraSensor* gazebo::sensors::CameraSensorPtr
- 9.9.2.3 typedef std::vector<ContactSensorPtr> gazebo::sensors::ContactSensor_V
- 9.9.2.4 typedef ContactSensor* gazebo::sensors::ContactSensorPtr
- 9.9.2.5 typedef std::vector<DepthCameraSensorPtr> gazebo::sensors::DepthCameraSensor_V
- 9.9.2.6 typedef DepthCameraSensor* gazebo::sensors::DepthCameraSensorPtr
- 9.9.2.7 typedef std::vector<GpuRaySensorPtr> gazebo::sensors::GpuRaySensor_V
- 9.9.2.8 typedef GpuRaySensor* gazebo::sensors::GpuRaySensorPtr
- 9.9.2.9 typedef std::vector<ImuSensorPtr> gazebo::sensors::ImuSensor_V
- 9.9.2.10 typedef ImuSensor* gazebo::sensors::ImuSensorPtr
- 9.9.2.11 typedef std::vector<RaySensorPtr> gazebo::sensors::RaySensor_V
- 9.9.2.12 typedef RaySensor* gazebo::sensors::RaySensorPtr
- 9.9.2.13 typedef std::vector<RFIDSensor> gazebo::sensors::RFIDSensor_V
- 9.9.2.14 typedef RFIDSensor* gazebo::sensors::RFIDSensorPtr
- 9.9.2.15 typedef std::vector<RFIDTag> gazebo::sensors::RFIDTag_V
- 9.9.2.16 typedef RFIDTag* gazebo::sensors::RFIDTagPtr
- 9.9.2.17 typedef std::vector<SensorPtr> gazebo::sensors::Sensor_V
- 9.9.2.18 typedef Sensor*(* gazebo::sensors::SensorFactoryFn)()
- 9.9.2.19 typedef Sensor* gazebo::sensors::SensorPtr

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 (p. 109) type of sensor is not a RAY or IMAGE sensor.

CATEGORY_COUNT Number of **Sensor** (p. 683) Categories.

9.10 gazebo::transport Namespace Reference

Classes

- class **CallbackHelper**
A (p. 109) helper class to handle callbacks when messages arrive.
- class **CallbackHelperT**
Callback helper Template.
- class **Connection**
Single TCP/IP connection manager.
- class **ConnectionManager**
Manager of connections.
- class **ConnectionReadTask**

- class **IOManager**
Manages boost::asio IO.
- class **Node**
A (p. 109) node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.
- class **Publication**
A (p. 109) publication for a topic.
- class **PublicationTransport**
transport/transport.hh
- class **Publisher**
A (p. 109) 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 (p. 109) subscriber to a topic.
- class **SubscriptionTransport**
transport/transport.hh
- class **TopicManager**
Manages topics and their subscriptions.

Typedefs

- typedef **CallbackHelper * CallbackHelperPtr**
boost shared pointer to **transport::CallbackHelper** (p. 155)
- typedef **Connection * ConnectionPtr**
- typedef **Node * NodePtr**
- typedef **Publication * PublicationPtr**
- typedef **PublicationTransport * PublicationTransportPtr**
- typedef **Publisher * PublisherPtr**
- typedef **Subscriber * SubscriberPtr**
- typedef **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.
- 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 unpaue incoming messages.
- 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 **stop** ()
Stop the transport component from running.

9.10.1 Typedef Documentation

9.10.1.1 typedef Connection* gazebo::transport::ConnectionPtr

9.10.1.2 typedef Node* gazebo::transport::NodePtr

9.10.1.3 typedef Publication* gazebo::transport::PublicationPtr

9.10.1.4 typedef PublicationTransport* gazebo::transport::PublicationTransportPtr

9.10.1.5 typedef Publisher* gazebo::transport::PublisherPtr

9.10.1.6 typedef Subscriber* gazebo::transport::SubscriberPtr

9.10.1.7 typedef SubscriptionTransport* gazebo::transport::SubscriptionTransportPtr

9.11 gazebo::util Namespace Reference

Classes

- class **DiagnosticManager**
A (p. 109) diagnostic manager class.
- class **DiagnosticTimer**
A (p. 109) timer designed for diagnostics.

Typedefs

- typedef **DiagnosticTimer * DiagnosticTimerPtr**

9.11.1 Typedef Documentation

9.11.1.1 typedef DiagnosticTimer* gazebo::util::DiagnosticTimerPtr

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. 89).*

9.16 Ogre Namespace Reference

9.17 ogre Namespace Reference

9.18 sdf Namespace Reference

namespace for Simulation Description Format parser

Classes

- class **Converter**
*Convert from one version of **SDF** (p. 680) to another.*
- class **Element**
***SDF** (p. 680) **Element** (p. 269) class.*
- class **Param**
***A** (p. 109) parameter class.*
- class **ParamT**
Templatized parameter class.
- class **Plugin**
- class **SDF**
*Base **SDF** (p. 680) class.*

Typedefs

- typedef **Element** * **ElementPtr**
- typedef std::vector< **ElementPtr** > **ElementPtr_V**
- typedef std::vector< **ParamPtr** > **Param_V**
- typedef **Param** * **ParamPtr**
- typedef **SDF** * **SDFPtr**

Functions

- void **addNestedModel** (**ElementPtr** _sdf, **ElementPtr** _includeSDF)
- void **copyChildren** (**ElementPtr** _sdf, TiXmlElement * _xml)
- bool **init** (**SDFPtr** _sdf)
Init based on the installed sdf_format.xml file.
- bool **initDoc** (TiXmlDocument * _xmlDoc, **SDFPtr** _sdf)
- bool **initDoc** (TiXmlDocument * _xmlDoc, **ElementPtr** _sdf)
- bool **initFile** (const std::string & _filename, **SDFPtr** _sdf)
- bool **initFile** (const std::string & _filename, **ElementPtr** _sdf)
- bool **initString** (const std::string & _xmlString, **SDFPtr** _sdf)
- bool **initXml** (TiXmlElement * _xml, **ElementPtr** _sdf)
- bool **readDoc** (TiXmlDocument * _xmlDoc, **SDFPtr** _sdf, const std::string & _source)
*Populate the **SDF** (p. 680) values from a TinyXML document.*
- bool **readDoc** (TiXmlDocument * _xmlDoc, **ElementPtr** _sdf, const std::string & _source)
- bool **readFile** (const std::string & _filename, **SDFPtr** _sdf)

Populate the **SDF** (p. 680) values from a file.

- bool **readString** (const std::string &_xmlString, **SDFPtr** _sdf)

Populate the **SDF** (p. 680) values from a string.

- bool **readString** (const std::string &_xmlString, **ElementPtr** _sdf)
- bool **readXml** (TiXmlElement *_xml, **ElementPtr** _sdf)

9.18.1 Detailed Description

namespace for Simulation Description Format parser

9.18.2 Typedef Documentation

9.18.2.1 typedef **Element*** **sdf::ElementPtr**

9.18.2.2 typedef std::vector< **ElementPtr** > **sdf::ElementPtr_V**

9.18.2.3 typedef std::vector< **ParamPtr** > **sdf::Param_V**

9.18.2.4 typedef **Param*** **sdf::ParamPtr**

9.18.2.5 typedef **SDF*** **sdf::SDFPtr**

9.18.3 Function Documentation

9.18.3.1 void **sdf::addNestedModel** (**ElementPtr** _sdf, **ElementPtr** _includeSDF)

9.18.3.2 void **sdf::copyChildren** (**ElementPtr** _sdf, TiXmlElement *_xml)

9.18.3.3 bool **sdf::init** (**SDFPtr** _sdf)

Init based on the installed sdf_format.xml file.

9.18.3.4 bool **sdf::initDoc** (TiXmlDocument *_xmlDoc, **SDFPtr** _sdf)

9.18.3.5 bool **sdf::initDoc** (TiXmlDocument *_xmlDoc, **ElementPtr** _sdf)

9.18.3.6 bool **sdf::initFile** (const std::string & _filename, **SDFPtr** _sdf)

9.18.3.7 bool **sdf::initFile** (const std::string & _filename, **ElementPtr** _sdf)

9.18.3.8 bool **sdf::initString** (const std::string & _xmlString, **SDFPtr** _sdf)

9.18.3.9 bool **sdf::initXml** (TiXmlElement *_xml, **ElementPtr** _sdf)

9.18.3.10 bool **sdf::readDoc** (TiXmlDocument *_xmlDoc, **SDFPtr** _sdf, const std::string & _source)

Populate the **SDF** (p. 680) values from a TinyXML document.

9.18.3.11 `bool sdf::readDoc (TiXmlDocument * _xmlDoc, ElementPtr _sdf, const std::string & _source)`

9.18.3.12 `bool sdf::readFile (const std::string & _filename, SDFPtr _sdf)`

Populate the **SDF** (p. 680) values from a file.

9.18.3.13 `bool sdf::readString (const std::string & _xmlString, SDFPtr _sdf)`

Populate the **SDF** (p. 680) values from a string.

9.18.3.14 `bool sdf::readString (const std::string & _xmlString, ElementPtr _sdf)`

9.18.3.15 `bool sdf::readXml (TiXmlElement * _xml, ElementPtr _sdf)`

9.19 SkyX Namespace Reference

9.20 urdf2gazebo Namespace Reference

namespace for URDF to SDF parser

Classes

- class **GazeboExtension**
- class **URDF2Gazebo**

Typedefs

- typedef const urdf::Link * **ConstUrdfLinkPtr**
- typedef urdf::Collision * **UrdfCollisionPtr**
- typedef urdf::Link * **UrdfLinkPtr**
- typedef urdf::Visual * **UrdfVisualPtr**

9.20.1 Detailed Description

namespace for URDF to SDF parser

Chapter 10

Class Documentation

10.1 A Class Reference

holding gazebo extension elements in urdf

10.1.1 Detailed Description

holding gazebo extension elements in urdf

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

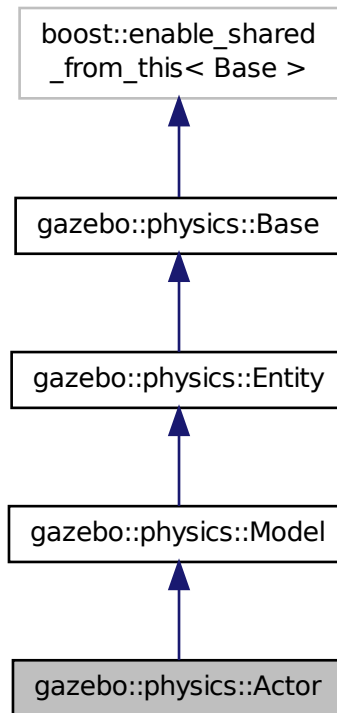
- `parser_urdf.hh`

10.2 gazebo::physics::Actor Class Reference

Actor (p. 109) 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. 135) 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.
- std::string **visualName**
Name of the visual.

Additional Inherited Members

10.2.1 Detailed Description

Actor (p. 109) class enables GPU based mesh model / skeleton scriptable animation.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 gazebo::physics::Actor::Actor (BasePtr _parent) [explicit]

Constructor.

Parameters

in	<code>_parent</code>	Parent object
----	----------------------	---------------

10.2.2.2 virtual gazebo::physics::Actor::~~Actor () [virtual]

Destructor.

10.2.3 Member Function Documentation

10.2.3.1 virtual void gazebo::physics::Actor::Fini () [virtual]

Finalize the actor.

Reimplemented from **gazebo::physics::Model** (p. 485).

10.2.3.2 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. 488).

10.2.3.3 virtual void gazebo::physics::Actor::Init () [virtual]

Initialize the actor.

Reimplemented from **gazebo::physics::Model** (p. 489).

10.2.3.4 virtual bool gazebo::physics::Actor::IsActive () [virtual]

Returns true when actor is playing animation.

10.2.3.5 void gazebo::physics::Actor::Load (sdf::ElementPtr _sdf) [virtual]

Load the actor.

Parameters

in	<i>_sdf</i>	SDF parameters
----	-------------	----------------

Reimplemented from **gazebo::physics::Entity** (p. 284).

10.2.3.6 virtual void gazebo::physics::Actor::Play () [virtual]

Start playing the script.

10.2.3.7 virtual void gazebo::physics::Actor::Stop () [virtual]

Stop playing the script.

10.2.3.8 void gazebo::physics::Actor::Update () [virtual]

Update the actor.

Reimplemented from **gazebo::physics::Base** (p. 146).

10.2.3.9 virtual void gazebo::physics::Actor::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::Model** (p. 493).

10.2.4 Member Data Documentation

10.2.4.1 `bool gazebo::physics::Actor::active` [protected]

True if the actor is being updated.

10.2.4.2 `bool gazebo::physics::Actor::autoStart` [protected]

True if the actor should start running automatically.

10.2.4.3 `transport::PublisherPtr gazebo::physics::Actor::bonePosePub` [protected]

Where to send bone info.

10.2.4.4 `std::map<std::string, bool> gazebo::physics::Actor::interpolateX` [protected]

True to interpolate along x direction.

10.2.4.5 `math::Vector3 gazebo::physics::Actor::lastPos` [protected]

Last position of the actor.

10.2.4.6 `double gazebo::physics::Actor::lastScriptTime` [protected]

Time the script was last updated.

10.2.4.7 `unsigned int gazebo::physics::Actor::lastTraj` [protected]

The last trajectory.

10.2.4.8 `bool gazebo::physics::Actor::loop` [protected]

True if the animation should loop.

10.2.4.9 `LinkPtr gazebo::physics::Actor::mainLink` [protected]

Base (p. 135) link.

10.2.4.10 `const common::Mesh* gazebo::physics::Actor::mesh` [protected]

Pointer to the actor's mesh.

10.2.4.11 `std::string gazebo::physics::Actor::oldAction` [protected]

The old action.

10.2.4.12 `double gazebo::physics::Actor::pathLength` [protected]

Length of the actor's path.

10.2.4.13 `common::Time gazebo::physics::Actor::playStartTime` [protected]

Time when the animation was started.

10.2.4.14 `common::Time gazebo::physics::Actor::prevFrameTime` [protected]

Time of the previous frame.

10.2.4.15 `double gazebo::physics::Actor::scriptLength` [protected]

Time length of a script.

10.2.4.16 `std::map<std::string, common::SkeletonAnimation*> gazebo::physics::Actor::skelAnimation` [protected]

Skeleton animations.

10.2.4.17 `common::Skeleton* gazebo::physics::Actor::skeleton` [protected]

The actor's skeleton.

10.2.4.18 `std::map<std::string, std::map<std::string, std::string>> gazebo::physics::Actor::skelNodesMap` [protected]

Skeleton to node map.

10.2.4.19 `std::string gazebo::physics::Actor::skinFile` [protected]

Filename for the skin.

10.2.4.20 `double gazebo::physics::Actor::skinScale` [protected]

Scaling factor to apply to the skin.

10.2.4.21 `double gazebo::physics::Actor::startDelay` [protected]

Amount of time to delay start by.

10.2.4.22 `std::map<unsigned int, common::PoseAnimation*> gazebo::physics::Actor::trajectories` [protected]

All the trajectories.

10.2.4.23 `std::vector<TrajectoryInfo> gazebo::physics::Actor::trajInfo` [protected]

Trajectory information.

10.2.4.24 `std::string gazebo::physics::Actor::visualName` [protected]

Name of the visual.

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

- **Actor.hh**

10.3 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.

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.

10.3.1 Detailed Description

An angle and related functions.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 gazebo::math::Angle::Angle ()

Constructor.

10.3.2.2 gazebo::math::Angle::Angle (double _radian)

Copy Constructor.

Parameters

<i>in</i>	<i>_radian</i>	Radians
-----------	----------------	---------

10.3.2.3 gazebo::math::Angle::Angle (const Angle & *_angle*)

Copy constructor.

Parameters

<i>in</i>	<i>_angle</i>	Angle (p. 116) to copy
-----------	---------------	-------------------------------

10.3.2.4 virtual gazebo::math::Angle::~~Angle () [virtual]

Destructor.

10.3.3 Member Function Documentation

10.3.3.1 double gazebo::math::Angle::Degree () const

Get the angle in degrees.

Returns

double containing the angle's degree value

10.3.3.2 void gazebo::math::Angle::Normalize ()

Normalize the angle in the range -Pi to Pi.

10.3.3.3 bool gazebo::math::Angle::operator!=(const Angle & *_angle*) const

Inequality.

Parameters

<i>in</i>	<i>_angle</i>	Angle (p. 116) to check for inequality
-----------	---------------	---

Returns

true if this != *_angle*

10.3.3.4 double gazebo::math::Angle::operator*() const [inline]

Dereference operator.

Returns

Double containing the angle's radian value

10.3.3.5 Angle gazebo::math::Angle::operator*(const Angle & *_angle*) const

Multiplication operator, result = this * *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 116) for multiplication
----	---------------	--

Returns

the new angle

10.3.3.6 **Angle** gazebo::math::Angle::operator*=(const **Angle** & *_angle*)

Multiplication set, this = this * *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 116) for multiplication
----	---------------	--

Returns

angle

10.3.3.7 **Angle** gazebo::math::Angle::operator+(const **Angle** & *_angle*) const

Addition operator, result = this + *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 116) for addition
----	---------------	------------------------------------

Returns

the new angle

10.3.3.8 **Angle** gazebo::math::Angle::operator+=(const **Angle** & *_angle*)

Addition set, this = this + *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 116) for addition
----	---------------	------------------------------------

Returns

angle

10.3.3.9 **Angle** gazebo::math::Angle::operator-(const **Angle** & *_angle*) const

Substraction, result = this - *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 116) for subtraction
----	---------------	---------------------------------------

Returns

the new angle

10.3.3.10 **Angle** gazebo::math::Angle::operator-= (const **Angle** & *_angle*)

Subtraction set, this = this - *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 116) for subtraction
----	---------------	---------------------------------------

Returns

angle

10.3.3.11 **Angle** gazebo::math::Angle::operator/ (const **Angle** & *_angle*) const

Division, result = this / *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 116) for division
----	---------------	------------------------------------

Returns

the new angle

10.3.3.12 **Angle** gazebo::math::Angle::operator/= (const **Angle** & *_angle*)

Division set, this = this / *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 116) for division
----	---------------	------------------------------------

Returns

angle

10.3.3.13 **bool** gazebo::math::Angle::operator< (const **Angle** & *_angle*) const

Less than operator.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 116) to check
-----------------	---------------------	--------------------------------

Returns

true if this < `_angle`

10.3.3.14 `bool gazebo::math::Angle::operator<= (const Angle & _angle) const`

Less or equal operator.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 116) to check
-----------------	---------------------	--------------------------------

Returns

true if this <= `_angle`

10.3.3.15 `bool gazebo::math::Angle::operator==(const Angle & _angle) const`

Equality operator, result = this == `_angle`.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 116) to check for equality
-----------------	---------------------	---

Returns

true if this == `_angle`

10.3.3.16 `bool gazebo::math::Angle::operator> (const Angle & _angle) const`

Greater than operator.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 116) to check
-----------------	---------------------	--------------------------------

Returns

true if this > `_angle`

10.3.3.17 `bool gazebo::math::Angle::operator>= (const Angle & _angle) const`

Greater or equal operator.

Parameters

in	<i>_angle</i>	Angle (p. 116) to check
----	---------------	--------------------------------

Returns

true if this \geq *_angle*

10.3.3.18 `double gazebo::math::Angle::Radian () const`

Get the angle in radians.

Returns

double containing the angle's radian value

10.3.3.19 `void gazebo::math::Angle::SetFromDegree (double _degree)`

Set the value from an angle in degrees.

Parameters

in	<i>_degree</i>	Degree value
----	----------------	--------------

10.3.3.20 `void gazebo::math::Angle::SetFromRadian (double _radian)`

Set the value from an angle in radians.

Parameters

in	<i>_radian</i>	Radian value
----	----------------	--------------

10.3.4 Friends And Related Function Documentation

10.3.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Angle & _a) [friend]`

Stream insertion operator.

Outputs in degrees

Parameters

in	<i>_out</i>	output stream
in	<i>_a</i>	angle to output

Returns

The output stream

10.3.4.2 `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

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

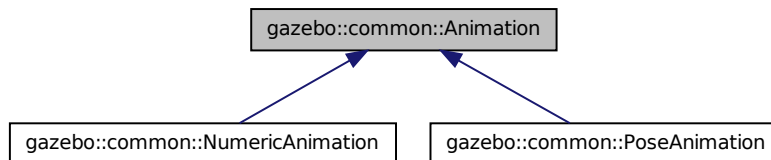
- **Angle.hh**

10.4 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 &_firstKey-Index) 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

10.4.1 Detailed Description

Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.

10.4.2 Member Typedef Documentation

10.4.2.1 typedef std::vector<KeyFrame*> gazebo::common::Animation::KeyFrame_V [protected]

array of keyframe type alias

10.4.3 Constructor & Destructor Documentation

10.4.3.1 `gazebo::common::Animation::Animation (const std::string & _name, double _length, bool _loop)`

Constructor.

Parameters

in	<i>_name</i>	Name of the animation, should be unique
in	<i>_length</i>	Duration of the animation in seconds
in	<i>_loop</i>	Set to true if the animation should repeat

10.4.3.2 `virtual gazebo::common::Animation::~~Animation () [virtual]`

Destructor.

10.4.4 Member Function Documentation

10.4.4.1 `void gazebo::common::Animation::AddTime (double _time)`

Add time to the animation.

Parameters

in	<i>_time</i>	The amount of time to add in seconds
----	--------------	--------------------------------------

10.4.4.2 `KeyFrame* gazebo::common::Animation::GetKeyFrame (unsigned int _index) const`

Get a key frame using an index value.

Parameters

in	<i>_index</i>	The index of the key frame
----	---------------	----------------------------

Returns

A (p. 109) pointer the keyframe, NULL if the *_index* is invalid

10.4.4.3 `unsigned int gazebo::common::Animation::GetKeyFrameCount () const`

Return the number of key frames in the animation.

Returns

The number of keyframes

10.4.4.4 `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	<code>_time</code>	The time in seconds
out	<code>_kf1</code>	Lower bound keyframe that is returned
out	<code>_kf2</code>	Upper bound keyframe that is returned
out	<code>_firstKeyIndex</code>	Index of the lower bound key frame

Returns

The time between the two keyframe

10.4.4.5 double gazebo::common::Animation::GetLength () const

Return the duration of the animation.

Returns

Duration of the animation in seconds

10.4.4.6 double gazebo::common::Animation::GetTime () const

Return the current time position.

Returns

The time position in seconds

10.4.4.7 void gazebo::common::Animation::SetLength (double *len*)

Set the duration of the animation.

Parameters

in	<code>_len</code>	The length of the animation in seconds
----	-------------------	--

10.4.4.8 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
----	--------------------	------------------------------

10.4.5 Member Data Documentation**10.4.5.1 bool gazebo::common::Animation::build [mutable], [protected]**

determines if the interpolation splines need building

10.4.5.2 KeyFrame_V gazebo::common::Animation::keyFrames [protected]

array of key frames

10.4.5.3 double gazebo::common::Animation::length [protected]

animation duration

10.4.5.4 bool gazebo::common::Animation::loop [protected]

true if animation repeats

10.4.5.5 std::string gazebo::common::Animation::name [protected]

animation name

10.4.5.6 double gazebo::common::Animation::timePos [protected]

current time position

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

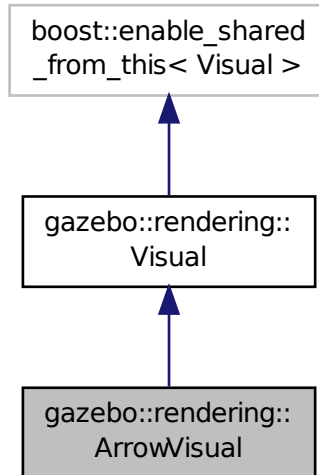
- **Animation.hh**

10.5 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

10.5.1 Detailed Description

Basic arrow visualization.

10.5.2 Constructor & Destructor Documentation

10.5.2.1 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

10.5.2.2 virtual gazebo::rendering::ArrowVisual::~~ArrowVisual () [virtual]

Destructor.

10.5.3 Member Function Documentation

10.5.3.1 virtual void gazebo::rendering::ArrowVisual::Load () [virtual]

Load the visual with default parameters.

Reimplemented from **gazebo::rendering::Visual** (p. 881).

10.5.3.2 void gazebo::rendering::ArrowVisual::ShowRotation ()

Show the rotation of the visual.

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

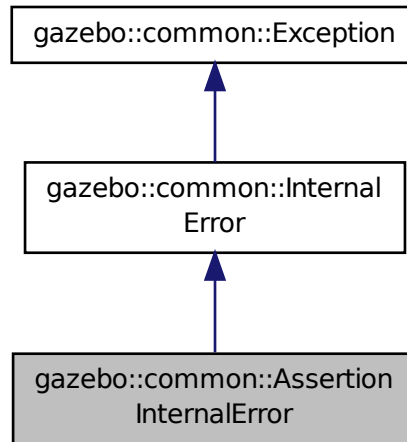
- **ArrowVisual.hh**

10.6 gazebo::common::AssertionInternalError Class Reference

Class for generating Exceptions which come from gazebo assertions.

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

Inheritance diagram for gazebo::common::AssertionInternalError:



Public Member Functions

- **AssertionInternalError** (const char * _file, int _line, const std::string _expr, const std::string _function, const std::string _msg="")
Constructor for assertions.
- virtual ~**AssertionInternalError** ()
Destructor.

10.6.1 Detailed Description

Class for generating Exceptions which come from gazebo assertions.

They include information about the assertion expression violated, function where problem appeared and assertion debug message.

10.6.2 Constructor & Destructor Documentation

- 10.6.2.1 gazebo::common::AssertionInternalError::AssertionInternalError (const char * *_file*, int *_line*, const std::string *_expr*, const std::string *_function*, const std::string *_msg* = " ")

Constructor for assertions.

Parameters

in	<i>_file</i>	File name
in	<i>_line</i>	Line number where the error occurred
in	<i>_expr</i>	Assertion expression failed resulting in an internal error

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

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

Destructor.

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

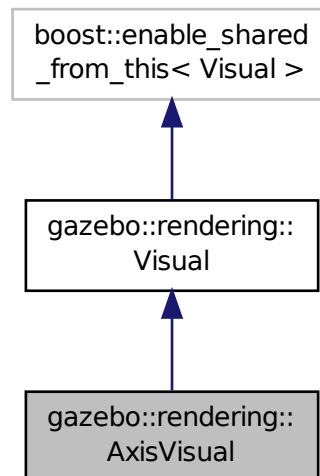
- **Exception.hh**

10.7 gazebo::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	<code>_name</code>	Name of the AxisVisual (p. 131)
in	<code>_vis</code>	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. 881).

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

Scale the X axis.

Parameters

in	<code>_scale</code>	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:

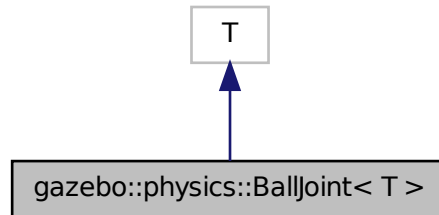
- **AxisVisual.hh**

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

Base (p. 135) 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. 133).*
- 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. 135) 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	<i>_parent</i>	Pointer to the parent link.
----	----------------	-----------------------------

References gazebo::physics::Base::BALL_JOINT.

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. 133).

Parameters

in	_sdf	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:

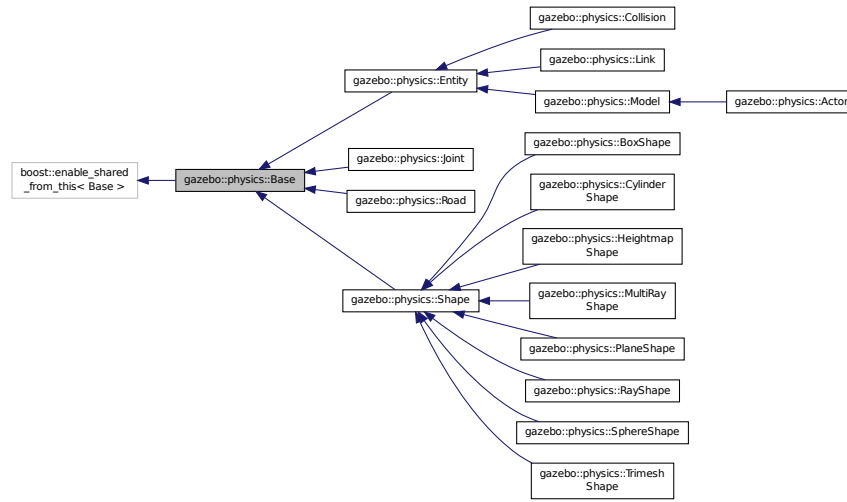
- **BallJoint.hh**

10.9 gazebo::physics::Base Class Reference

Base (p. 135) 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,
TRIMESH_SHAPE = 0x00400000 }

Unique identifiers for all entity types.

Public Member Functions

- **Base** (**BasePtr** _parent)
Constructor.
- virtual **~Base** ()
Destructor.
- void **AddChild** (**BasePtr** _child)
Add a child to this entity.
- void **AddType** (**EntityType** _type)
Add a type specifier.
- virtual void **Fini** ()
Finalize the object.
- **BasePtr** **GetById** (unsigned int _id) const

This is an internal function.

- **BasePtr GetByName** (const std::string &_name)
Get by name.
- **BasePtr GetChild** (unsigned int _i) const
Get a child by index.
- **BasePtr GetChild** (const std::string &_name)
Get a child by name.
- unsigned int **GetChildCount** () const
Get the number of children.
- unsigned int **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. 893) 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. 138)'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 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. 135) 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. 135) type.

ENTITY **Entity** (p. 277) type.

MODEL **Model** (p. 480) type.

LINK **Link** (p. 412) type.

COLLISION **Collision** (p. 192) type.

ACTOR Actor (p. 109) type.

LIGHT Light type.

VISUAL Visual type.

JOINT Joint (p. 375) type.

BALL_JOINT BallJoint (p. 133) type.

HINGE2_JOINT Hing2Joint type.

HINGE_JOINT HingeJoint (p. 354) type.

SLIDER_JOINT SliderJoint (p. 733) type.

SCREW_JOINT ScrewJoint (p. 677) type.

UNIVERSAL_JOINT UniversalJoint (p. 810) type.

SHAPE Shape (p. 706) type.

BOX_SHAPE BoxShape (p. 151) type.

CYLINDER_SHAPE CylinderShape (p. 246) type.

HEIGHTMAP_SHAPE HeightmapShape (p. 348) type.

MAP_SHAPE MapShape type.

MULTIRAY_SHAPE MultiRayShape (p. 518) type.

RAY_SHAPE RayShape (p. 634) type.

PLANE_SHAPE PlaneShape (p. 578) type.

SPHERE_SHAPE SphereShape (p. 737) type.

TRIMESH_SHAPE TrimeshShape (p. 807) type.

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

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

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

Finalize the object.

Reimplemented in **gazebo::physics::Actor** (p. 112), **gazebo::physics::Model** (p. 485), **gazebo::physics::Entity** (p. 280), **gazebo::physics::Link** (p. 419), and **gazebo::physics::Collision** (p. 195).

10.9.4.4 BasePtr gazebo::physics::Base::GetById (unsigned int *_id*) const

This is an internal function.

Get a child or self by id.

Parameters

<i>in</i>	<i>_id</i>	ID of the object to retrieve.
-----------	------------	-------------------------------

Returns

A (p. 109) pointer to the object, NULL if not found

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 (p. 109) 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 (p. 109) 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 (p. 109) 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 unsigned int 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. 112), and **`gazebo::physics::Model`** (p. 488).

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. 893) this object is in.

Returns

The **World** (p. 893) 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

<code>in</code>	<code>_t</code>	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. 388), **`gazebo::physics::RayShape`** (p. 637), **`gazebo::physics::Actor`** (p. 113), **`gazebo::physics::Model`** (p. 489), **`gazebo::physics::Link`** (p. 425), **`gazebo::physics::Collision`** (p. 198), **`gazebo::physics::HeightmapShape`** (p. 351), **`gazebo::physics::TrimeshShape`** (p. 809), **`gazebo::physics::Multi-RayShape`** (p. 525), **`gazebo::physics::PlaneShape`** (p. 580), **`gazebo::physics::Road`** (p. 653), **`gazebo::physics::Shape`** (p. 708), **`gazebo::physics::SphereShape`** (p. 739), **`gazebo::physics::BoxShape`** (p. 153), and **`gazebo::physics::CylinderShape`** (p. 249).

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. 388), **`gazebo::physics::Actor`** (p. 113), **`gazebo::physics::Entity`** (p. 284), **`gazebo::physics::Model`** (p. 489), **`gazebo::physics::Link`** (p. 426), **`gazebo::physics::Collision`** (p. 199), **`gazebo::physics::HeightmapShape`** (p. 351), and **`gazebo::physics::Road`** (p. 653).

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

in	_ent	Base (p. 135) 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	_prefix	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	_id	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	_name	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. 389), **gazebo::physics::Model** (p. 490), **gazebo::physics::Entity** (p. 285), and **gazebo::physics::Link** (p. 426).

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

Calls recursive Reset on one of the **Base::EntityType** (p. 138)'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. 286).

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

Set the parent.

Parameters

in	<code>_parent</code>	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	<code>_v</code>	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	<code>_show</code>	True to set this entity as selected.
----	--------------------	--------------------------------------

Reimplemented in `gazebo::physics::Link` (p. 429).

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	<code>_newWorld</code>	The new World (p. 893) 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. 391), `gazebo::physics::MultiRayShape` (p. 525), `gazebo::physics::Actor` (p. 113), `gazebo::physics::RayShape` (p. 639), `gazebo::physics::Link` (p. 430), `gazebo::physics::Model` (p. 493), and `gazebo::physics::TrimeshShape` (p. 810).

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. 391), `gazebo::physics::Actor` (p. 113), `gazebo::physics::Model` (p. 493), `gazebo::physics::Link` (p. 430), `gazebo::physics::Entity` (p. 287), and `gazebo::physics::Collision` (p. 200).

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 ~**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** &_b)
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	<i>_min</i>	Minimum corner of the box
in	<i>_max</i>	Maximum corner of the box

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

Copy Constructor.

Parameters

in	<i>_b</i>	Box (p. 146) 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. 146) 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. 146) 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

in	_b	Box (p. 146) 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. 146) 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. 146) 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

<code>in</code>	<code>_out</code>	Output stream
<code>in</code>	<code>_b</code>	Box (p. 146) 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:

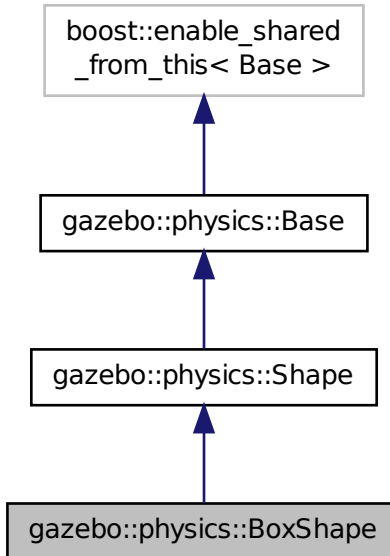
- **Box.hh**

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 **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. 192).
----	----------------	-----------------------------------

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. 708).

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. 708).

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

Process a geometry message.

Parameters

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

Implements **gazebo::physics::Shape** (p. 708).

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

Set the size of the box.

Parameters

in	_size	Size of each side of the box.
----	-------	-------------------------------

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 ()

Destructor.

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

Returns

A (p. 109) pointer to a new **Skeleton** (p. 713)

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

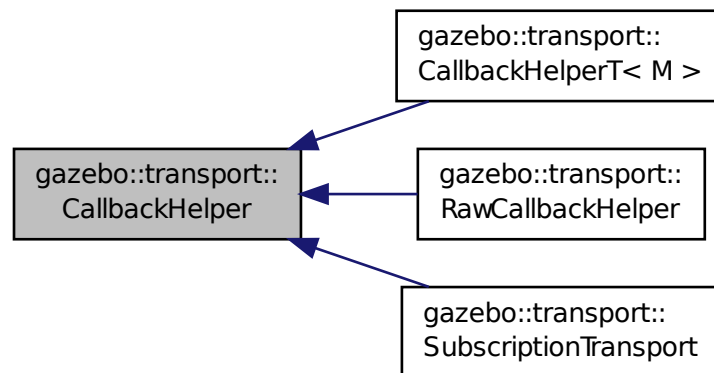
- **BVHLoader.hh**

10.13 gazebo::transport::CallbackHelper Class Reference

A (p. 109) 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)=0
Process new incoming data.
- 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 (p. 109) 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

<i>in</i>	<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. 626), and **gazebo::transport::CallbackHelperT< M >** (p. 159).

10.13.3.4 virtual bool gazebo::transport::CallbackHelper::HandleData (const std::string & *_newdata*) [pure virtual]

Process new incoming data.

Parameters

in	<i>_newdata</i>	Incoming data to be processed
----	-----------------	-------------------------------

Returns

true if successfully processed; false otherwise

Implemented in **gazebo::transport::RawCallbackHelper** (p. 626), **gazebo::transport::CallbackHelperT< M >** (p. 159), and **gazebo::transport::SubscriptionTransport** (p. 764).

10.13.3.5 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. 627), **gazebo::transport::CallbackHelperT< M >** (p. 159), and **gazebo::transport::SubscriptionTransport** (p. 764).

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:

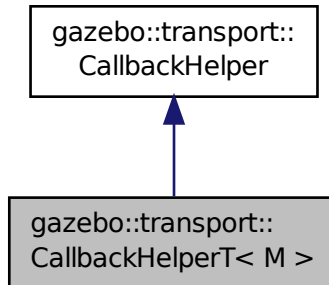
- **CallbackHelper.hh**

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 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)
Process new incoming data.
- 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 M const > &) [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. 157).

References `gzthrow`, and `NULL`.

10.14.3.2 `template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::HandleData (const std::string & _newdata)` `[inline], [virtual]`

Process new incoming data.

Parameters

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

Returns

true if successfully processed; false otherwise

Implements `gazebo::transport::CallbackHelper` (p. 157).

10.14.3.3 `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. 157).

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

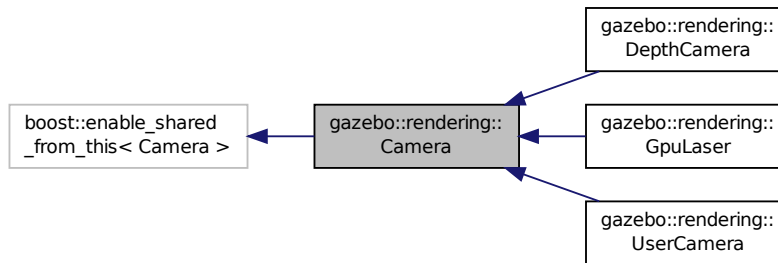
- `CallbackHelper.hh`

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.*
- template<typename T >
 - event::ConnectionPtr ConnectNewImageFrame** (T _subscriber)
 - Connect a to the new image signal.*
- void **CreateRenderTexture** (const std::string &_textureName)
 - Set the render target.*
- void **DisconnectNewImageFrame** (**event::ConnectionPtr** &_c)
 - Disconnect from an image frame.*
- void **EnableSaveFrame** (bool _enable)
 - Enable or disable saving.*
- virtual void **Fini** ()
 - Finalize the camera.*
- float **GetAspectRatio** () const
 - Get the aspect ratio.*
- virtual float **GetAvgFPS** ()
 - 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.*
- **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.
- `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 **IsInitialized** () const **GAZEBO_DEPRECATED**
- Deprecated.*

 - 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 **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** (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.
- virtual void **RenderImpl** ()
Implementation of the render call.
- bool **TrackVisualImpl** (const std::string &_visualName)
*Implementation of the **Camera::TrackVisual** (p. 181) 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.
- 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. 662) *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. 662) *node that controls camera position.*
- sdf::ElementPtr **sdf**
Camera (p. 159)'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

in	<i>_namePrefix</i>	Unique prefix name for the camera.
in	<i>_scene</i>	Scene (p. 662) that will contain the camera
in	<i>_autoRender</i>	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. 816).

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

10.15.3.3 virtual bool gazebo::rendering::Camera::AttachToVisualImpl (const std::string & *_name*, bool *_inheritOrientation*, double *_minDist* = 0, double *_maxDist* = 0) [protected],[virtual]

Attach the camera to a scene node.

Parameters

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

Returns

True on success

10.15.3.4 `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	<code>_visual</code>	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

Reimplemented in `gazebo::rendering::UserCamera` (p. 816).

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

Connect a to the new image signal.

Parameters

in	<code>_subscriber</code>	Callback that is called when a new image is generated
----	--------------------------	---

Returns

A (p. 109) pointer to the connection. This must be kept in scope.

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

10.15.3.6 `void gazebo::rendering::Camera::CreateRenderTexture (const std::string & _textureName)`

Set the render target.

Parameters

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

10.15.3.7 `void gazebo::rendering::Camera::DisconnectNewImageFrame (event::ConnectionPtr & _c) [inline]`

Disconnect from an image frame.

Parameters

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

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

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

Enable or disable saving.

Parameters

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

10.15.3.9 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. 323), `gazebo::rendering::DepthCamera` (p. 253), and `gazebo::rendering::UserCamera` (p. 817).

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

Get the aspect ratio.

Returns

The aspect ratio (width / height) in pixels

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

Get the average FPS.

Returns

The average frames per second

10.15.3.12 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	<code>_screenx</code>	X coordinate in the camera's viewport, in pixels.
in	<code>_screeny</code>	Y coordinate in the camera's viewport, in pixels.
out	<code>_origin</code>	Origin in the world coordinate frame of the resulting ray
out	<code>_dir</code>	Direction of the resulting ray

10.15.3.13 `math::Vector3 gazebo::rendering::Camera::GetDirection () const`

Get the camera's direction vector.

Returns

Direction the camera is facing

10.15.3.14 `double gazebo::rendering::Camera::GetFarClip ()`

Get the far clip distance.

Returns

Far clip distance

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

Get the next frame filename based on SDF parameters.

Returns

The frame's filename

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

Get the camera FOV (horizontal)

Returns

The horizontal field of view

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

Get the image size in bytes.

Returns

Size in bytes

10.15.3.18 `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

10.15.3.19 `virtual const unsigned char* gazebo::rendering::Camera::GetImageData (unsigned int i = 0)` [virtual]

Get a pointer to the image data.

Get the raw image data from a camera's buffer.

Parameters

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

Returns

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

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

Get the depth of the image.

Returns

Depth of the image

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

Get the string representation of the image format.

Returns

String representation of the image format.

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

Get the height of the image.

Returns

Image height

Reimplemented in `gazebo::rendering::UserCamera` (p. 817).

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

Get the width of the image.

Returns

Image width

Reimplemented in `gazebo::rendering::UserCamera` (p. 818).

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

Return true if the camera has been initialized.

Returns

True if initialized was successful

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

Get the last time the camera was rendered.

Returns

Time the camera was last rendered

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

Get the camera's name.

Returns

The name of the camera

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

Get the near clip distance.

Returns

Near clip distance

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

Get a pointer to the ogre camera.

Returns

Pointer to the OGRE camera

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

Get the camera's pitch scene node.

Returns

The pitch node the camera is attached to

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

Get the render Hz rate.

Returns

The Hz rate

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

Get the render texture.

Returns

Pointer to the render texture

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

Get the viewport right vector.

Returns

The viewport right vector

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

Get the scene this camera is in.

Returns

Pointer to scene containing this camera

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

Get the camera's scene node.

Returns

The scene node the camera is attached to

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

Get the height of the off-screen render texture.

Returns

Render texture height

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

Get the width of the off-screen render texture.

Returns

Render texture width

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

Get the triangle count.

Returns

The current triangle count

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

Get the viewport up vector.

Returns

The viewport up vector

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

Get the camera FOV (vertical)

Returns

The vertical field of view

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

Get a pointer to the Ogre::Viewport.

Returns

Pointer to the Ogre::Viewport

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

Get the viewport height in pixels.

Returns

The viewport height

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

Get the viewport width in pixels.

Returns

The viewport width

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

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

Returns

The ID of the window.

10.15.3.44 `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

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

Get the global pose of the camera.

Returns

Pose of the camera in the world coordinate frame

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

Get the camera position in the world.

Returns

The world position of the camera

10.15.3.47 `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. 609)

10.15.3.48 `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.

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

Initialize the camera.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 323), **gazebo::rendering::DepthCamera** (p. 253), and **gazebo::rendering::UserCamera** (p. 819).

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

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

10.15.3.51 `bool gazebo::rendering::Camera::IsInitialized () const [inline]`

Deprecated.

See Also

GetInitialized (p. 171)

10.15.3.52 `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

10.15.3.53 `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

10.15.3.54 `virtual void gazebo::rendering::Camera::Load (sdf::ElementPtr _sdf) [virtual]`

Load the camera with a set of parameters.

Parameters

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

Reimplemented in **`gazebo::rendering::UserCamera`** (p. 819).

10.15.3.55 `virtual void gazebo::rendering::Camera::Load () [virtual]`

Load the camera with default parameters.

Reimplemented in **`gazebo::rendering::GpuLaser`** (p. 323), **`gazebo::rendering::DepthCamera`** (p. 253), and **`gazebo::rendering::UserCamera`** (p. 819).

10.15.3.56 `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. 177)

Parameters

<code>in</code>	<code>_pose</code>	End position of the camera
<code>in</code>	<code>_time</code>	Duration of the camera's movement

Reimplemented in **`gazebo::rendering::UserCamera`** (p. 820).

10.15.3.57 `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. 176)

Parameters

in	<code>_pts</code>	Vector of poses to move to
in	<code>_time</code>	Duration of the entire move
in	<code>_onComplete</code>	Callback that is called when the move is complete

10.15.3.58 `virtual void gazebo::rendering::Camera::PostRender () [virtual]`

Post render.

Called after the render signal.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 323), **gazebo::rendering::DepthCamera** (p. 253), and **gazebo::rendering::UserCamera** (p. 820).

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

Render the camera.

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

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

Implementation of the render call.

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

Rotate the camera around the pitch axis.

Parameters

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

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

Rotate the camera around the yaw axis.

Parameters

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

10.15.3.63 `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

10.15.3.64 `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

<code>in</code>	<code>_image</code>	The raw image buffer
<code>in</code>	<code>_width</code>	Width of the image
<code>in</code>	<code>_height</code>	Height of the image
<code>in</code>	<code>_depth</code>	Depth of the image data
<code>in</code>	<code>_format</code>	Format the image data is in
<code>in</code>	<code>_filename</code>	Name of the file in which to write the frame

Returns

True if saving was successful

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

Set the aspect ratio.

Parameters

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

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

Set whether to capture data.

Parameters

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

10.15.3.67 `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

10.15.3.68 void gazebo::rendering::Camera::SetHFOV (math::Angle *_angle*)

Set the camera FOV (horizontal)

Parameters

in	<code>_radians</code>	Horizontal field of view
----	-----------------------	--------------------------

10.15.3.69 void gazebo::rendering::Camera::SetImageHeight (unsigned int *_h*)

Set the image height.

Parameters

in	<code>_h</code>	Image height
----	-----------------	--------------

10.15.3.70 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

10.15.3.71 void gazebo::rendering::Camera::SetImageWidth (unsigned int *_w*)

Set the image height.

Parameters

in	<code>_w</code>	Image width
----	-----------------	-------------

10.15.3.72 void gazebo::rendering::Camera::SetName (const std::string & *_name*)

Set the camera's name.

Parameters

in	<code>_name</code>	New name for the camera
----	--------------------	-------------------------

10.15.3.73 void gazebo::rendering::Camera::SetRenderRate (double *_hz*)

Set the render Hz rate.

Parameters

in	<i>_hz</i>	The Hz rate
----	------------	-------------

10.15.3.74 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. 821).

10.15.3.75 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
----	------------------	--

10.15.3.76 void gazebo::rendering::Camera::SetScene (ScenePtr *_scene*)

Set the scene this camera is viewing.

Parameters

in	<i>_scene</i>	Pointer to the scene
----	---------------	----------------------

10.15.3.77 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
----	--------------	---

10.15.3.78 void gazebo::rendering::Camera::SetWindowId (unsigned int *_windowId*)

10.15.3.79 virtual void gazebo::rendering::Camera::SetWorldPose (const math::Pose & *_pose*) [virtual]

Set the global pose of the camera.

Parameters

in	_pose	The new math::Pose (p. 584) of the camera
----	-------	--

Reimplemented in **gazebo::rendering::UserCamera** (p. 821).

10.15.3.80 void gazebo::rendering::Camera::SetWorldPosition (const math::Vector3 & _pos)

Set the world position.

Parameters

in	_pos	The new position of the camera
----	------	--------------------------------

10.15.3.81 void gazebo::rendering::Camera::SetWorldRotation (const math::Quaternion & _quat)

Set the world orientation.

Parameters

in	_quat	The new orientation of the camera
----	-------	-----------------------------------

10.15.3.82 void gazebo::rendering::Camera::ShowWireframe (bool _s)

Set whether to view the world in wireframe.

Parameters

in	_s	Set to True to render objects as wireframe
----	----	--

10.15.3.83 void gazebo::rendering::Camera::ToggleShowWireframe ()

Toggle whether to view the world in wireframe.

10.15.3.84 void gazebo::rendering::Camera::TrackVisual (const std::string & _visualName)

Set the camera to track a scene node.

Parameters

in	_visualName	Name of the visual to track
----	-------------	-----------------------------

10.15.3.85 bool gazebo::rendering::Camera::TrackVisualImpl (const std::string & _visualName) [protected]

Implementation of the **Camera::TrackVisual** (p. 181) call.

Parameters

in	_visualName	Name of the visual to track
----	-------------	-----------------------------

Returns

True if able to track the visual

10.15.3.86 `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. 822).

10.15.3.87 `void gazebo::rendering::Camera::Translate (const math::Vector3 & _direction)`

Translate the camera.

Parameters

in	<code>_direction</code>	The translation vector
----	-------------------------	------------------------

10.15.3.88 `virtual void gazebo::rendering::Camera::Update ()` [virtual]

Reimplemented in `gazebo::rendering::UserCamera` (p. 822).

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 `std::vector<event::ConnectionPtr> gazebo::rendering::Camera::connections` [protected]

The camera's event connections.

10.15.4.6 `int gazebo::rendering::Camera::imageFormat` [protected]

Format for saving images.

10.15.4.7 `int gazebo::rendering::Camera::imageHeight` [protected]

Save image height.

10.15.4.8 `int gazebo::rendering::Camera::imageWidth` [protected]

Save image width.

10.15.4.9 `bool gazebo::rendering::Camera::initialized` [protected]

True if initialized.

10.15.4.10 `common::Time gazebo::rendering::Camera::lastRenderWallTime` [protected]

Time the last frame was rendered.

10.15.4.11 `std::string gazebo::rendering::Camera::name` [protected]

Name of the camera.

10.15.4.12 `bool gazebo::rendering::Camera::newData` [protected]

True if new data is available.

10.15.4.13 `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()`.

10.15.4.14 `boost::function<void()> gazebo::rendering::Camera::onAnimationComplete` [protected]

User callback for when an animation completes.

10.15.4.15 `Ogre::SceneNode* gazebo::rendering::Camera::pitchNode` [protected]

Scene (p. 662) nod that controls camera pitch.

10.15.4.16 `common::Time gazebo::rendering::Camera::prevAnimTime` [protected]

Previous time the camera animation was updated.

10.15.4.17 `Ogre::RenderTarget* gazebo::rendering::Camera::renderTarget` [protected]

Target that renders frames.

10.15.4.18 `Ogre::Texture* gazebo::rendering::Camera::renderTexture` [protected]

Texture that receives results from rendering.

10.15.4.19 `std::list<msgs::Request> gazebo::rendering::Camera::requests` [protected]

List of requests.

10.15.4.20 `unsigned int gazebo::rendering::Camera::saveCount` [protected]

Number of saved frames.

10.15.4.21 `unsigned char* gazebo::rendering::Camera::saveFrameBuffer` [protected]

10.15.4.22 `ScenePtr gazebo::rendering::Camera::scene` [protected]

Pointer to the scene.

10.15.4.23 `Ogre::SceneNode* gazebo::rendering::Camera::sceneNode` [protected]

Scene (p. 662) node that controls camera position.

10.15.4.24 `sdf::ElementPtr gazebo::rendering::Camera::sdf` [protected]

Camera (p. 159)'s SDF values.

10.15.4.25 `unsigned int gazebo::rendering::Camera::textureHeight` [protected]

Height of the render texture.

10.15.4.26 `unsigned int gazebo::rendering::Camera::textureWidth` [protected]

Width of the render texture.

10.15.4.27 `Ogre::Viewport* gazebo::rendering::Camera::viewport` [protected]

Viewport the ogre camera uses.

10.15.4.28 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:

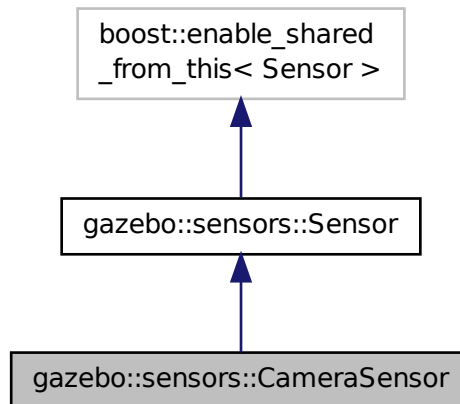
- **Camera.hh**

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. 159).
- 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.
- virtual void **SetParent** (const std::string &_name)
Set the parent of the sensor.

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. 687).

10.16.3.2 `rendering::CameraPtr gazebo::sensors::CameraSensor::GetCamera () const` `[inline]`

Returns a pointer to the `rendering::Camera` (p. 159).

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. 689).

10.16.3.7 `virtual void gazebo::sensors::CameraSensor::Init ()` `[virtual]`

Initialize the camera.

Reimplemented from `gazebo::sensors::Sensor` (p. 690).

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. 690).

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

in	<code>_sdf</code>	SDF Sensor (p. 683) parameters
in	<code>_worldName</code>	Name of world to load from

Reimplemented from `gazebo::sensors::Sensor` (p. 690).

10.16.3.10 `virtual void gazebo::sensors::CameraSensor::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. 690).

10.16.3.11 `bool gazebo::sensors::CameraSensor::SaveFrame (const std::string & _filename)`

Saves the image to the disk.

Parameters

in	<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::SetParent (const std::string & _name) [virtual]`

Set the parent of the sensor.

Parameters

<code>_name</code>	The name of the parent
--------------------	------------------------

Reimplemented from `gazebo::sensors::Sensor` (p. 691).

10.16.3.13 `virtual void gazebo::sensors::CameraSensor::UpdateImpl (bool _force)` `[protected], [virtual]`

Update the sensor information.

Parameters

<code>in</code>	<code><i>_force</i></code>	True if update is forced, false if not
-----------------	----------------------------	--

Reimplemented from `gazebo::sensors::Sensor` (p. 691).

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

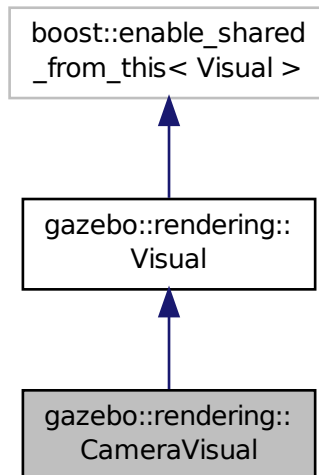
- `CameraSensor.hh`

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. 868).

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

<i>in</i>	<i>_name</i>	Name of the Visual (p. 868)
<i>in</i>	<i>_vis</i>	Pointer to the parent Visual (p. 868)

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. 868).

Parameters

<i>in</i>	<i>_width</i>	Width of the Camera (p. 159) image
<i>in</i>	<i>_height</i>	Height of the Camera (p. 159) image

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

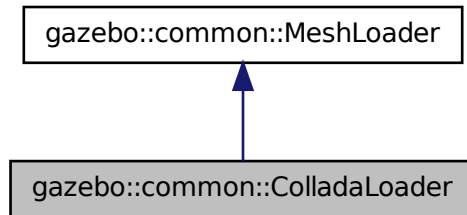
- **CameraVisual.hh**

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. 466)

Implements **gazebo::common::MeshLoader** (p. 475).

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

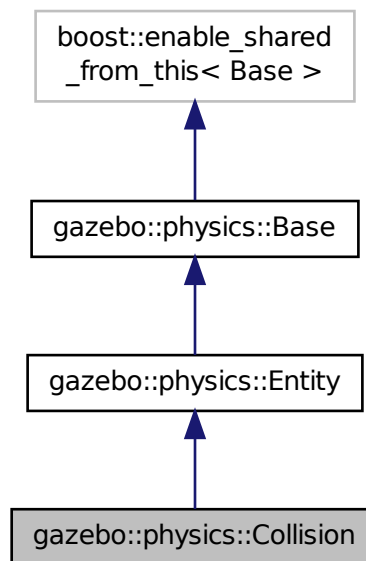
- **ColladaLoader.hh**

10.19 gazebo::physics::Collision Class Reference

Base (p. 135) 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)
Add an occurrence of a contact to this collision.

- `template<typename T >`
event::ConnectionPtr ConnectContact (T _subscriber)
Deprecated.
- `void DisconnectContact` (**event::ConnectionPtr** &_conn)
Deprecated.
- `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
Return true if contacts are on.
- `float GetLaserRetro` () const
Get the laser retro reflectiveness.
- **LinkPtr GetLink** () const
Get the link this collision belongs to.
- **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`)
Turn contact recording on or off.
- void **SetLaserRetro** (`float _retro`)
Set the laser retro reflectiveness.
- 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. 706).

Additional Inherited Members

10.19.1 Detailed Description

Base (p. 135) class for all collision entities.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 gazebo::physics::Collision::Collision (`LinkPtr _link`) `[explicit]`

Constructor.

Parameters

<code>in</code>	<code>_link</code>	Link (p. 412) 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 template<typename T > event::ConnectionPtr gazebo::physics::Collision::ConnectContact (T *_subscriber*)
[inline]

Deprecated.

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

10.19.3.3 void gazebo::physics::Collision::DisconnectContact (event::ConnectionPtr & *_conn*) [inline]

Deprecated.

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

10.19.3.4 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.
-----	-------------	---

10.19.3.5 virtual void gazebo::physics::Collision::Fini () [virtual]

Finalize the collision.

Reimplemented from **gazebo::physics::Entity** (p. 280).

10.19.3.6 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. 280).

10.19.3.7 `bool gazebo::physics::Collision::GetContactsEnabled () const`

Return true if contacts are on.

Returns

True if contacts are on.

10.19.3.8 `float gazebo::physics::Collision::GetLaserRetro () const`

Get the laser retro reflectiveness.

Returns

The laser retro value.

10.19.3.9 `LinkPtr gazebo::physics::Collision::GetLink () const`

Get the link this collision belongs to.

Returns

The parent **Link** (p. 412).

10.19.3.10 `ModelPtr gazebo::physics::Collision::GetModel () const`

Get the model this collision belongs to.

Returns

The parent model.

10.19.3.11 `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. 282).

10.19.3.12 `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. 282).

10.19.3.13 `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. 282).

10.19.3.14 `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. 283).

10.19.3.15 `ShapePtr gazebo::physics::Collision::GetShape () const`

Get the collision shape.

Returns

The collision shape.

10.19.3.16 `unsigned int gazebo::physics::Collision::GetShapeType ()`

Get the shape type.

Returns

The shape type.

See Also

EntityType (p. 138)

10.19.3.17 `CollisionState gazebo::physics::Collision::GetState ()`

Get the collision state.

Returns

The collision state.

10.19.3.18 **SurfaceParamsPtr** gazebo::physics::Collision::GetSurface () const [inline]

Get the surface parameters.

Returns

The surface parameters.

10.19.3.19 **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. 283).

10.19.3.20 **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. 283).

10.19.3.21 **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. 283).

10.19.3.22 **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. 284).

10.19.3.23 **virtual void** gazebo::physics::Collision::Init () [virtual]

Initialize the collision.

Reimplemented from **gazebo::physics::Base** (p. 143).

10.19.3.24 `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.

10.19.3.25 `virtual void gazebo::physics::Collision::Load (sdf::ElementPtr _sdf) [virtual]`

Load the collision.

Parameters

in	_sdf	SDF to load from.
----	------	-------------------

Reimplemented from `gazebo::physics::Entity` (p. 284).

10.19.3.26 `void gazebo::physics::Collision::ProcessMsg (const msgs::Collision & _msg)`

Update parameters from a message.

Parameters

in	_msg	Message to update from.
----	------	-------------------------

10.19.3.27 `virtual void gazebo::physics::Collision::SetCategoryBits (unsigned int _bits) [pure virtual]`

Set the category bits, used during collision detection.

Parameters

in	_bits	The bits to set.
----	-------	------------------

10.19.3.28 `virtual void gazebo::physics::Collision::SetCollideBits (unsigned int _bits) [pure virtual]`

Set the collide bits, used during collision detection.

Parameters

in	_bits	The bits to set.
----	-------	------------------

10.19.3.29 `void gazebo::physics::Collision::SetCollision (bool _placeable)`

Set the encapsulated collision object.

Parameters

in	<i>_placeable</i>	True to make the object movable.
----	-------------------	----------------------------------

10.19.3.30 void gazebo::physics::Collision::SetContactsEnabled (bool *_enable*)

Turn contact recording on or off.

Parameters

in	<i>_enable</i>	True to enable collision contacts.
----	----------------	------------------------------------

10.19.3.31 void gazebo::physics::Collision::SetLaserRetro (float *_retro*)

Set the laser retro reflectiveness.

Parameters

in	<i>_retro</i>	The laser retro value.
----	---------------	------------------------

10.19.3.32 void gazebo::physics::Collision::SetShape (ShapePtr *_shape*)

Set the shape for this collision.

Parameters

in	<i>_shape</i>	The shape for this collision object.
----	---------------	--------------------------------------

10.19.3.33 void gazebo::physics::Collision::SetState (const CollisionState & *_state*)

Set the current collision state.

Parameters

in	<i>The</i>	collision state.
----	------------	------------------

10.19.3.34 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.287).

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. 706).

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

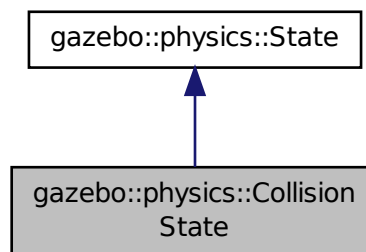
- **Collision.hh**

10.20 gazebo::physics::CollisionState Class Reference

Store state information of a **physics::Collision** (p. 192) 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. 192) 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. 192) object.

This class captures the entire state of a **Collision** (p. 192) at one specific time during a simulation run.

State (p. 744) of a **Collision** (p. 192) 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. 201) from an existing **Collision** (p. 192).

Parameters

in	_model	Pointer to the Link (p. 412) from which to gather state info.
----	--------	--

10.20.2.3 gazebo::physics::CollisionState::CollisionState (const **sdf::ElementPtr** _sdf) [explicit]

Constructor.

Build a **CollisionState** (p. 201) from SDF data

Parameters

in	_sdf	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	_sdf	SDF element to populate.
-----	------	--------------------------

10.20.3.2 const math::Pose& gazebo::physics::CollisionState::GetPose () const

Get the **Collision** (p. 192) pose.

Returns

The pose of the **CollisionState** (p. 201)

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. 201) information from stored data in and SDF::Element

Parameters

in	_elem	Pointer to the SDF::Element containing state info.
----	-------	--

Reimplemented from **gazebo::physics::State** (p. 747).

10.20.3.5 CollisionState gazebo::physics::CollisionState::operator+ (const CollisionState & *_state*) const

Addition operator.

Parameters

<i>in</i>	<i>_pt</i>	A (p. 109) state to add.
-----------	------------	---------------------------------

Returns

The resulting state.

10.20.3.6 CollisionState gazebo::physics::CollisionState::operator- (const CollisionState & *_state*) const

Subtraction operator.

Parameters

<i>in</i>	<i>_pt</i>	A (p. 109) 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. 744) 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. 192) 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

in	<i>_r</i>	Red value (range 0 to 1)
in	<i>_g</i>	Green value (range 0 to 1)
in	<i>_b</i>	Blue value (range 0 to 1)
in	<i>_a</i>	Alpha value (0=transparent, 1=opaque)

10.21.3.3 gazebo::common::Color::Color (const Color & *_clr*)

Copy Constructor.

Parameters

in	<i>_clr</i>	Color (p. 205) 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. 839) 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

<code>in</code>	<code>_pt</code>	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

<code>in</code>	<code>_pt</code>	Color (p. 205) 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

<code>in</code>	<code>_v</code>	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

<code>in</code>	<code>_pt</code>	Color (p. 205) to add
-----------------	------------------	------------------------------

Returns

The resulting color

10.21.4.14 `Color gazebo::common::Color::operator- (const Color & _pt) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_pt</code>	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

<code>in</code>	<code>_v</code>	Value to subtract
-----------------	-----------------	-------------------

Returns

The resulting color

10.21.4.16 `const Color& gazebo::common::Color::operator-= (const Color & _pt)`

Subtraction equal operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 205) to subtract
-----------------	------------------	-----------------------------------

Returns

The resulting color

10.21.4.17 `const Color gazebo::common::Color::operator/ (const Color & _pt) const`

Division operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 205) 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

<code>in</code>	<code>_v</code>	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

<code>in</code>	<code>_pt</code>	Color (p. 205) to divide by
-----------------	------------------	------------------------------------

Returns

The resulting color

10.21.4.20 `Color& gazebo::common::Color::operator= (const Color & _pt)`

Equal operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 205) to copy
-----------------	------------------	-------------------------------

Returns

Reference to this color

10.21.4.21 `bool gazebo::common::Color::operator==(const Color & _pt) const`

Equality operator.

Parameters

<code>in</code>	<code>_pt</code>	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

<code>in</code>	<code>_index</code>	Color (p. 205) 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

<code>in</code>	<code>_r</code>	Red value (range 0 to 1)
<code>in</code>	<code>_g</code>	Green value (range 0 to 1)
<code>in</code>	<code>_b</code>	Blue value (range 0 to 1)
<code>in</code>	<code>_a</code>	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	_h	Hue(0..360)
in	_s	Saturation(0..1)
in	_v	Value(0..1)

10.21.4.29 void gazebo::common::Color::SetFromRGBA (const RGBA _v)

Set from uint32 RGBA packed value.

Parameters

in	_v	the new color
----	----	---------------

10.21.4.30 void gazebo::common::Color::SetFromYUV (float _y, float _u, float _v)

Set from yuv.

Parameters

in	_y	value
in	_u	value
in	_v	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

<code>in</code>	<code>_out</code>	the output stream
<code>in</code>	<code>_pt</code>	the color

Returns

the output stream

10.21.5.2 `std::istream& operator>> (std::istream & _in, Color & _pt)` [friend]

Stream insertion operator.

Parameters

<code>in</code>	<code>_in</code>	the input stream
<code>in</code>	<code>_pt</code>	

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:

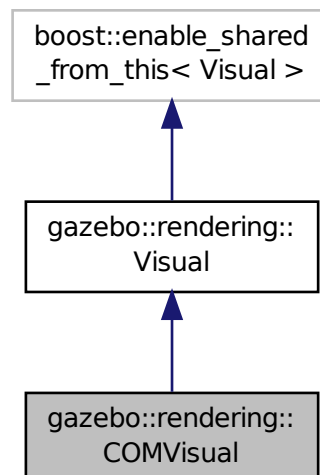
- **Color.hh**

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. 868) 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. 868)
in	<code>_vis</code>	Parent Visual (p. 868)

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. 868) 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	_msg	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 (p. 109) 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 (p. 109) 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	_e	Event (p. 288) pointer to connect with
in	_i	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

Referenced by gazebo::event::EventT< T >::Disconnect().

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

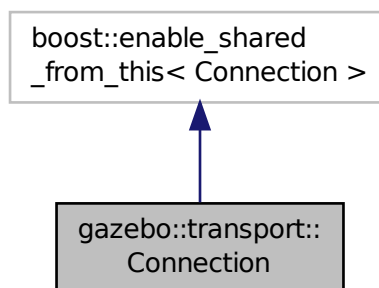
- **Event.hh**

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 **QueueMsg** (const std::string &_buffer, bool _force=false)
Write data to the socket.
- unsigned int **GetId** () const
Get the ID of the connection.
- 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	<code>_host</code>	The host to connect to
in	<code>_port</code>	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	<code>_subscriber</code>	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	<code>_subscriber</code>	Handle previously returned by ConnectToShutdown() (p. 222)
----	--------------------------	---

References `gazebo::event::EventT< T >::Disconnect()`.

10.24.4.6 `void gazebo::transport::Connection::EnqueueMsg (const std::string & _buffer, bool _force = false)`

Write data to the socket.

Parameters

in	<code>_buffer</code>	Data to write
in	<code>_force</code>	If true, block until the data has been written to the socket, otherwise just enqueue the data for asynchronous write

10.24.4.7 `unsigned int gazebo::transport::Connection::GetId () const`

Get the ID of the connection.

Returns

The connection's unique ID.

10.24.4.8 `std::string gazebo::transport::Connection::GetLocalAddress () const`

Get the local address of this connection.

Returns

The local address

10.24.4.9 `static std::string gazebo::transport::Connection::GetLocalHostname () [static]`

Get the local hostname.

Returns

The local hostname

10.24.4.10 `unsigned int gazebo::transport::Connection::GetLocalPort () const`

Get the port of this connection.

Returns

The local port

10.24.4.11 `std::string gazebo::transport::Connection::GetLocalURI () const`

Get the local URI.

Returns

The local URI

10.24.4.12 `std::string gazebo::transport::Connection::GetRemoteAddress () const`

Get the remote address.

Returns

The remote address

10.24.4.13 `std::string gazebo::transport::Connection::GetRemoteHostname () const`

Get the remote hostname.

Returns

The remote hostname

10.24.4.14 `unsigned int gazebo::transport::Connection::GetRemotePort () const`

Get the remote port number.

Returns

The remote port

10.24.4.15 `std::string gazebo::transport::Connection::GetRemoteURI () const`

Get the remote URI.

Returns

The remote URI

10.24.4.16 `bool gazebo::transport::Connection::IsOpen () const`

Is the connection open?

Returns

true if the connection is open; false otherwise

Referenced by AsyncRead().

10.24.4.17 `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

10.24.4.18 `void gazebo::transport::Connection::ProcessWriteQueue (bool _blocking = false)`

Handle on-write callbacks.

10.24.4.19 `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

10.24.4.20 void gazebo::transport::Connection::Shutdown ()

Shutdown the socket.

10.24.4.21 void gazebo::transport::Connection::StartRead (const ReadCallback & _cb)

Start a thread that reads from the connection and passes new message to the ReadCallback.

Parameters

in	_cb	The callback to invoke when a new message is received
----	-----	---

10.24.4.22 void gazebo::transport::Connection::StopRead ()

Stop the read loop.

10.24.4.23 static bool gazebo::transport::Connection::ValidateIP (const std::string & _ip) [static]

Return true if the _ip is a valid.

Parameters

in	_ip	Dotted quad to validate.
----	-----	--------------------------

Returns

True if the _ip is a valid.

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

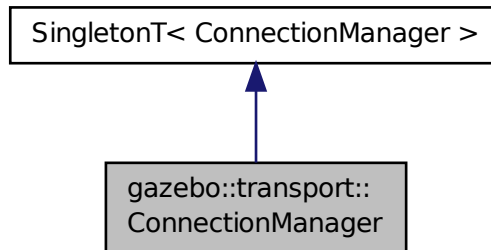
- **Connection.hh**

10.25 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 **RunUpdate** ()
Run the manager update loop once.
- void **Stop** ()
Stop the conneciton manager.
- void **Subscribe** (const std::string &_topic, const std::string &_msgType, bool _latching)
Subscribe to a topic.
- 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	<code>_publishers</code>	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	<code>_namespaces</code>	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	<code>_masterHost</code>	Host where the master is running
in	<code>_masterPort</code>	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	<code>_name</code>	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	<code>_conn</code>	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::RunUpdate ()

Run the manager update loop once.

10.25.2.12 void gazebo::transport::ConnectionManager::Stop ()

Stop the connecton manager.

10.25.2.13 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

10.25.2.14 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 (p. 109) 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]

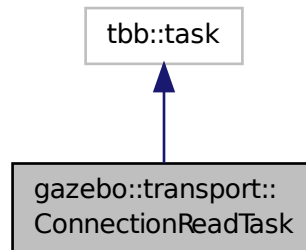
The documentation for this class was generated from the following file:

- **ConnectionManager.hh**

10.26 gazebo::transport::ConnectionReadTask Class Reference

```
#include <Connection.hh>
```

Inheritance diagram for gazebo::transport::ConnectionReadTask:



Public Member Functions

- **ConnectionReadTask** (boost::function< void(const std::string &);> _func, const std::string &_data)
Constructor.
- `tbb::task * execute ()`
Overridden function from tbb::task that executes the data callback.

10.26.1 Detailed Description

A (p. 109) task instance that is created when data is read from a socket and used by TBB

10.26.2 Constructor & Destructor Documentation

10.26.2.1 `gazebo::transport::ConnectionReadTask::ConnectionReadTask (boost::function< void(const std::string &);> _func, const std::string &_data)` [inline]

Constructor.

Parameters

	<code>_in]</code>	<code>_func</code> Boost function pointer, which is the function that receives the data.
<code>in</code>	<code>_data</code>	Data to send to the boost function pointer.

10.26.3 Member Function Documentation

10.26.3.1 `tbb::task*` `gazebo::transport::ConnectionReadTask::execute ()` `[inline]`

Overridden function from `tbb::task` that executes the data callback.

References NULL.

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

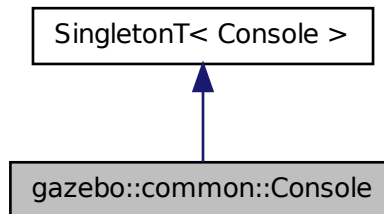
- [Connection.hh](#)

10.27 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.
- `void Init (const std::string &_logFilename)`
Load the message parameters.
- `std::ofstream & Log ()`
Use this to output a colored message to the terminal.
- `void SetQuiet (bool _q)`
Set quiet output.

Additional Inherited Members

10.27.1 Detailed Description

Message, error, warning functionality.

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

- **Console.hh**

10.28 gazebo::physics::Contact Class Reference

A (p. 109) 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

- std::string **collision1**
Name of the first collision object.
- std::string **collision2**
Name of 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.

- **JointWrench wrench** [32]

Array of forces for the contact.

10.28.1 Detailed Description

A (p. 109) contact between two collisions.

Each contact can consist of a number of contact points

10.28.2 Constructor & Destructor Documentation

10.28.2.1 gazebo::physics::Contact::Contact ()

Constructor.

10.28.2.2 gazebo::physics::Contact::Contact (const Contact & _contact)

Copy constructor.

Parameters

in	_contact	Contact (p. 232) to copy.
----	----------	----------------------------------

10.28.2.3 virtual gazebo::physics::Contact::~~Contact () [virtual]

Destructor.

10.28.3 Member Function Documentation

10.28.3.1 std::string gazebo::physics::Contact::DebugString () const

Produce a debug string.

Returns

A (p. 109) string that contains the values of the contact.

10.28.3.2 void gazebo::physics::Contact::FillMsg (msgs::Contact & _msg) const

Populate a msgs::Contact with data from this.

Parameters

out	_msg	Contact (p. 232) message the will hold the data.
-----	------	---

10.28.3.3 **Contact**& gazebo::physics::Contact::operator= (const **Contact** & *_contact*)

Operator =.

Parameters

in	_contact	Contact (p. 232) to copy.
----	----------	----------------------------------

Returns

Reference to this contact

10.28.3.4 **Contact**& gazebo::physics::Contact::operator= (const msgs::Contact & *_contact*)

Operator =.

Parameters

in	_contact	msgs::Contact to copy.
----	----------	------------------------

Returns

Reference to this contact

10.28.3.5 void gazebo::physics::Contact::Reset ()

Reset to default values.

10.28.4 Member Data Documentation

10.28.4.1 std::string gazebo::physics::Contact::collision1

Name of the first collision object.

10.28.4.2 std::string gazebo::physics::Contact::collision2

Name of the second collision object.

10.28.4.3 int gazebo::physics::Contact::count

Length of all the arrays.

10.28.4.4 double gazebo::physics::Contact::depths[32]

Array of contact depths.

10.28.4.5 `math::Vector3 gazebo::physics::Contact::normals[32]`

Array of force normals.

10.28.4.6 `math::Vector3 gazebo::physics::Contact::positions[32]`

Array of force positions.

10.28.4.7 `common::Time gazebo::physics::Contact::time`

Time at which the contact occurred.

10.28.4.8 `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**

10.29 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.
- **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. 235).
- **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.

10.29.1 Detailed Description

Aggregates all the contact information generated by the collision detection engine.

10.29.2 Constructor & Destructor Documentation

10.29.2.1 gazebo::physics::ContactManager::ContactManager ()

Constructor.

10.29.2.2 virtual gazebo::physics::ContactManager::~~ContactManager () [virtual]

Destructor.

10.29.3 Member Function Documentation

10.29.3.1 void gazebo::physics::ContactManager::Clear ()

Clear all stored contacts.

10.29.3.2 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. 236).

Parameters

<i>in</i>	<i>_index</i>	Index of the Contact (p. 232) to return.
-----------	---------------	---

Returns

Pointer to a contact, NULL If index is invalid.

10.29.3.3 unsigned int gazebo::physics::ContactManager::GetContactCount () const

Return the number of valid contacts.

10.29.3.4 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. 236)

Returns

Vector of contact pointers.

10.29.3.5 void gazebo::physics::ContactManager::Init (WorldPtr *_world*)

Initialize the **ContactManager** (p. 235).

This is required in order to publish contact messages via the **ContactManager::PublishContacts** (p. 237) method.

Parameters

<i>in</i>	<i>_world</i>	Pointer to the world that is initializing the contact manager.
-----------	---------------	--

10.29.3.6 **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.

10.29.3.7 void gazebo::physics::ContactManager::PublishContacts ()

Publish all contacts in a msgs::Contacts message.

10.29.3.8 void gazebo::physics::ContactManager::ResetCount ()

Set the contact count to zero.

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

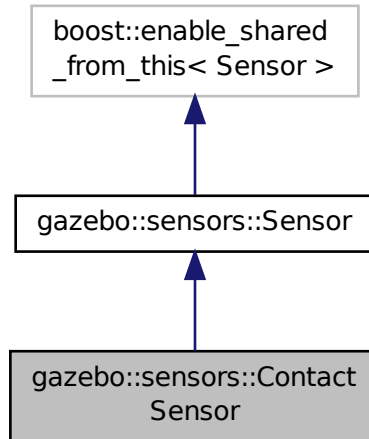
- **ContactManager.hh**

10.30 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. 237).*
- 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. 687).

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

<code>in</code>	<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

<code>in</code>	<code>_index</code>	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. 237).

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. 237) 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. 237) SDF. All collision pairs between **ContactSensor** (p. 237) `<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 `collision2`) are stored in an array of elements, (`position`, `normal`, `depth`, `wrench`). **A** (p. 109) 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. 690).

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. 690).

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. 683) parameters
in	<code>_worldName</code>	Name of world to load from

Reimplemented from `gazebo::sensors::Sensor` (p. 690).

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. 690).

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. 691).

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

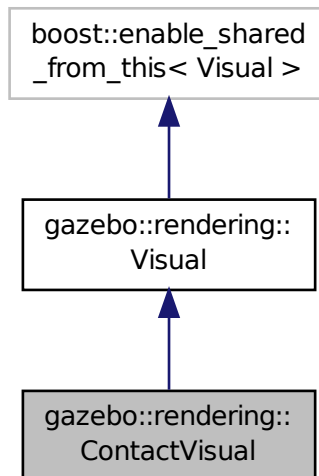
- **ContactSensor.hh**

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. 242)
in	<code><i>_vis</i></code>	Pointer the parent Visual (p. 868)
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. 243) **Conversions.hh** (p. 948) **rendering/Conversions.hh** (p. 948).

```
#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. 106) 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. 106) quaternion.*
- static **math::Quaternion Convert** (const Ogre::Quaternion &_v)
***Ogre** (p. 106) quaternion to Gazebo quaternion.*

10.32.1 Detailed Description

Conversions (p. 243) **Conversions.hh** (p. 948) **rendering/Conversions.hh** (p. 948).

A (p. 109) set of utility function to convert between Gazebo and **Ogre** (p. 106) 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. 106) 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. 106) 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. 106) Vector from Gazebo Vector3

Parameters

in	_v	Gazebo vector
----	----	---------------

Returns

Ogre (p. 106) 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. 106) 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. 106) quaternion.

Parameters

in	_v	Gazebo quaternion
----	----	-------------------

Returns

Ogre (p. 106) quaternion

10.32.2.6 `static math::Quaternion gazebo::rendering::Conversions::Convert (const Ogre::Quaternion & _v) [static]`

Ogre (p. 106) quaternion to Gazebo quaternion.

Parameters

in	_v	Ogre (p. 106) quaternion return Gazebo quaternion
----	----	--

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

- **Conversions.hh**

10.33 sdf::Converter Class Reference

Convert from one version of **SDF** (p. 680) to another.

```
#include <Converter.hh>
```

Static Public Member Functions

- static bool **Convert** (TiXmlDocument * _doc, const std::string & _toVersion, bool _quiet=false)
Convert **SDF** (p. 680) to the specified version.
- static void **Convert** (TiXmlDocument * _doc, TiXmlDocument * _convertDoc)
This is an internal function.

10.33.1 Detailed Description

Convert from one version of **SDF** (p. 680) to another.

10.33.2 Member Function Documentation

10.33.2.1 static bool sdf::Converter::Convert (TiXmlDocument * _doc, const std::string & _toVersion, bool _quiet = false)
[static]

Convert **SDF** (p. 680) to the specified version.

Parameters

in	<code>_doc</code>	SDF (p. 680) xml doc
in	<code>_toVersion</code>	Version number in string format
in	<code>_quiet</code>	False to be more verbose

10.33.2.2 static void sdf::Converter::Convert (TiXmlDocument * _doc, TiXmlDocument * _convertDoc) [static]

This is an internal function.

Generic convert function that converts the **SDF** (p. 680) based on the given Convert file.

Parameters

in	<code>_doc</code>	SDF (p. 680) xml doc
in	<code>_convertDoc</code>	Convert xml doc

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

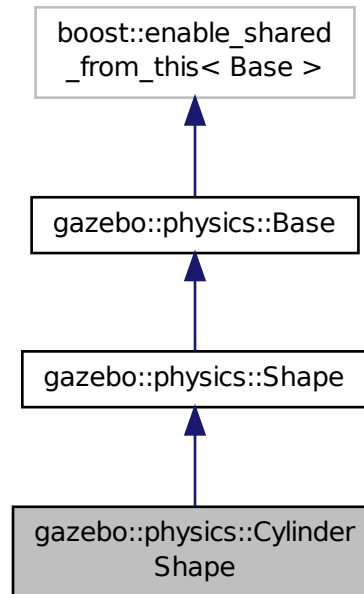
- **Converter.hh**

10.34 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 **SetSize** (double _radius, double _length)
Set the size of the cylinder.

Additional Inherited Members

10.34.1 Detailed Description

Cylinder collision.

10.34.2 Constructor & Destructor Documentation

10.34.2.1 `gazebo::physics::CylinderShape::CylinderShape (CollisionPtr _parent)` [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent of the shape.
----	----------------	----------------------

10.34.2.2 `virtual gazebo::physics::CylinderShape::~~CylinderShape ()` [virtual]

Destructor.

10.34.3 Member Function Documentation

10.34.3.1 `void gazebo::physics::CylinderShape::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. 708).

10.34.3.2 `double gazebo::physics::CylinderShape::GetLength ()` const

Get length.

Returns

The cylinder length.

10.34.3.3 `double gazebo::physics::CylinderShape::GetRadius ()` const

Get radius.

Returns

The cylinder radius.

10.34.3.4 void gazebo::physics::CylinderShape::Init () [virtual]

Initialize the cylinder.

Implements **gazebo::physics::Shape** (p. 708).

10.34.3.5 virtual void gazebo::physics::CylinderShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]

Update values based on a message.

Parameters

in	<i>_msg</i>	Message to update from.
----	-------------	-------------------------

Implements **gazebo::physics::Shape** (p. 708).

10.34.3.6 void gazebo::physics::CylinderShape::SetLength (double *_length*)

Set length.

Parameters

in	<i>_length</i>	New length of the cylinder.
----	----------------	-----------------------------

10.34.3.7 void gazebo::physics::CylinderShape::SetRadius (double *_radius*)

Set radius.

Parameters

<i>in</i> }	<i>_radius</i>	New radius of the cylinder.
-------------	----------------	-----------------------------

10.34.3.8 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.

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

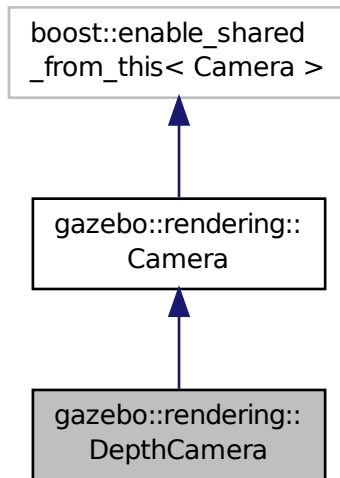
- **CylinderShape.hh**

10.35 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 parameters.

- void **Load** ()

Load the camera with default parameters.

- 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

10.35.1 Detailed Description

Depth camera used to render depth data into an image buffer.

10.35.2 Constructor & Destructor Documentation

10.35.2.1 gazebo::rendering::DepthCamera::DepthCamera (const std::string & *_namePrefix*, ScenePtr *_scene*, bool *_autoRender* =true)

Constructor.

Parameters

in	<i>_namePrefix</i>	Unique prefix name for the camera.
in	<i>_scene</i>	Scene (p. 662) that will contain the camera
in	<i>_autoRender</i>	Almost everyone should leave this as true.

10.35.2.2 virtual gazebo::rendering::DepthCamera::~DepthCamera () [virtual]

Destructor.

10.35.3 Member Function Documentation

10.35.3.1 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().

10.35.3.2 `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().

10.35.3.3 `void gazebo::rendering::DepthCamera::CreateDepthTexture (const std::string & _textureName)`

Create a texture which will hold the depth data.

Parameters

in	<code>_textureName</code>	Name of the texture to create
----	---------------------------	-------------------------------

10.35.3.4 `void gazebo::rendering::DepthCamera::DisconnectNewDepthFrame (event::ConnectionPtr & _c) [inline]`

Disconnect from an depth image singal.

Parameters

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

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

10.35.3.5 `void gazebo::rendering::DepthCamera::DisconnectNewRGBPointCloud (event::ConnectionPtr & c) [inline]`

Disconnect from an rgb point cloud singal.

Parameters

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

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

10.35.3.6 void gazebo::rendering::DepthCamera::Fini () [virtual]

Finalize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 168).

10.35.3.7 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

10.35.3.8 void gazebo::rendering::DepthCamera::Init () [virtual]

Initialize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 175).

10.35.3.9 void gazebo::rendering::DepthCamera::Load (sdf::ElementPtr & _sdf)

Load the camera with a set of parameters.

Parameters

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

10.35.3.10 void gazebo::rendering::DepthCamera::Load () [virtual]

Load the camera with default parameters.

Reimplemented from **gazebo::rendering::Camera** (p. 176).

10.35.3.11 virtual void gazebo::rendering::DepthCamera::PostRender () [virtual]

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 177).

10.35.3.12 virtual void gazebo::rendering::DepthCamera::SetDepthTarget (Ogre::RenderTarget* *_target*) [virtual]

Set the render target, which renders the depth data.

Parameters

in	<i>_target</i>	Pointer to the render target
----	----------------	------------------------------

10.35.4 Member Data Documentation

10.35.4.1 `Ogre::RenderTarget*` `gazebo::rendering::DepthCamera::depthTarget` [protected]

Pointer to the depth target.

10.35.4.2 `Ogre::Texture*` `gazebo::rendering::DepthCamera::depthTexture` [protected]

Pointer to the depth texture.

10.35.4.3 `Ogre::Viewport*` `gazebo::rendering::DepthCamera::depthViewport` [protected]

Pointer to the depth viewport.

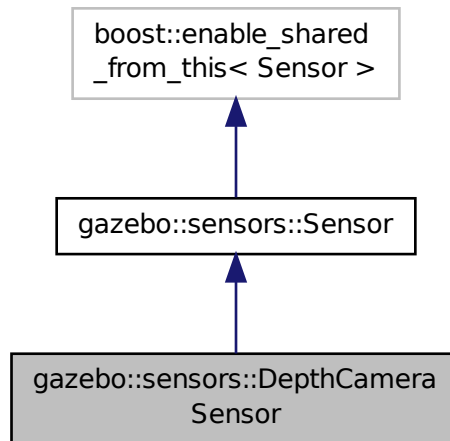
The documentation for this class was generated from the following file:

- **DepthCamera.hh**

10.36 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. 249).
- `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.
- virtual `void SetParent (const std::string &_name)`
Set the parent of the sensor.

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

10.36.1 Constructor & Destructor Documentation

10.36.1.1 gazebo::sensors::DepthCameraSensor::DepthCameraSensor ()

Constructor.

10.36.1.2 virtual gazebo::sensors::DepthCameraSensor::~~DepthCameraSensor () [virtual]

Destructor.

10.36.2 Member Function Documentation

10.36.2.1 virtual void gazebo::sensors::DepthCameraSensor::Fini () [protected],[virtual]

Finalize the camera.

Reimplemented from `gazebo::sensors::Sensor` (p. 687).

10.36.2.2 `rendering::DepthCameraPtr gazebo::sensors::DepthCameraSensor::GetDepthCamera () const` `[inline]`

Returns a pointer to the `rendering::DepthCamera` (p. 249).

Returns

Depth Camera pointer

10.36.2.3 `virtual void gazebo::sensors::DepthCameraSensor::Init ()` `[protected]`, `[virtual]`

Initialize the camera.

Reimplemented from `gazebo::sensors::Sensor` (p. 690).

10.36.2.4 `virtual void gazebo::sensors::DepthCameraSensor::Load (const std::string & _worldName, sdf::ElementPtr & _sdf)` `[protected]`, `[virtual]`

Load the sensor with SDF parameters.

Parameters

<code>in</code>	<code>_sdf</code>	SDF Sensor (p. 683) parameters
<code>in</code>	<code>_worldName</code>	Name of world to load from

10.36.2.5 `virtual void gazebo::sensors::DepthCameraSensor::Load (const std::string & _worldName)` `[protected]`, `[virtual]`

Load the sensor with default parameters.

Parameters

<code>in</code>	<code>_worldName</code>	Name of world to load from
-----------------	-------------------------	----------------------------

Reimplemented from `gazebo::sensors::Sensor` (p. 690).

10.36.2.6 `bool gazebo::sensors::DepthCameraSensor::SaveFrame (const std::string & _filename)`

Saves an image frame of depth camera sensor to file.

Parameters

<code>in</code>	<code>Name</code>	of file to save as
-----------------	-------------------	--------------------

Returns

True if saved, false if not

10.36.2.7 `virtual void gazebo::sensors::DepthCameraSensor::SetActive (bool _value)` `[virtual]`

Set whether the sensor is active or not.

Parameters

in	<code>_value</code>	True if active, false if not
----	---------------------	------------------------------

Reimplemented from `gazebo::sensors::Sensor` (p. 691).

10.36.2.8 `virtual void gazebo::sensors::DepthCameraSensor::SetParent (const std::string & _name) [virtual]`

Set the parent of the sensor.

Parameters

in	<code>_name</code>	Name of parent
----	--------------------	----------------

Reimplemented from `gazebo::sensors::Sensor` (p. 691).

10.36.2.9 `virtual void gazebo::sensors::DepthCameraSensor::UpdateImpl (bool _force) [protected],[virtual]`

Update the sensor information.

Parameters

in	<code>_force</code>	True if update is forced, false if not
----	---------------------	--

Reimplemented from `gazebo::sensors::Sensor` (p. 691).

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

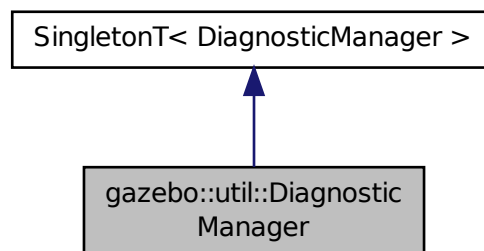
- `DepthCameraSensor.hh`

10.37 gazebo::util::DiagnosticManager Class Reference

A (p. 109) 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

10.37.1 Detailed Description

A (p. 109) diagnostic manager class.

10.37.2 Member Function Documentation

10.37.2.1 `std::string gazebo::util::DiagnosticManager::GetLabel (int _index) const`

Get a label for a timer.

Parameters

<code>in</code>	<code>_index</code>	Index of a timer instance
-----------------	---------------------	---------------------------

Returns

Label of the specified timer

10.37.2.2 `boost::filesystem::path gazebo::util::DiagnosticManager::GetLogPath () const`

Get the path in which logs are stored.

Returns

The path in which logs are stored.

10.37.2.3 common::Time gazebo::util::DiagnosticManager::GetTime (int *_index*) const

Get the time of a timer instance.

Parameters

<i>in</i>	<i>_index</i>	The index of a timer instance
-----------	---------------	-------------------------------

Returns

Time of the specified timer

10.37.2.4 common::Time gazebo::util::DiagnosticManager::GetTime (const std::string & *_label*) const

Get a time based on a label.

Parameters

<i>in</i>	<i>_label</i>	Name of the timer instance
-----------	---------------	----------------------------

Returns

Time of the specified timer

10.37.2.5 int gazebo::util::DiagnosticManager::GetTimerCount () const

Get the number of timers.

Returns

The number of timers

10.37.2.6 void gazebo::util::DiagnosticManager::Init (const std::string & *_worldName*)

Initialize to report diagnostics about a world.

Parameters

<i>in</i>	<i>_worldName</i>	Name of the world.
-----------	-------------------	--------------------

10.37.2.7 void gazebo::util::DiagnosticManager::Lap (const std::string & *_name*, const std::string & *_prefix*)

Output the current elapsed time of an active timer with a prefix string.

This also resets the timer and keeps it running.

Parameters

<i>in</i>	<i>_name</i>	Name of the timer to access.
<i>in</i>	<i>_prefix</i>	Informational string that is output with the elapsed time.

10.37.2.8 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 (p. 109) pointer to the new diagnostic timer

10.37.2.9 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:

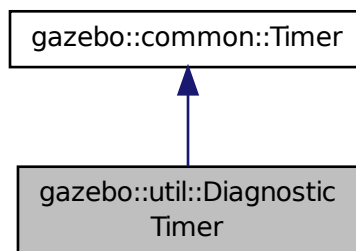
- **Diagnostics.hh**

10.38 gazebo::util::DiagnosticTimer Class Reference

A (p. 109) 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.

10.38.1 Detailed Description

A (p. 109) timer designed for diagnostics.

10.38.2 Constructor & Destructor Documentation

10.38.2.1 gazebo::util::DiagnosticTimer::DiagnosticTimer (const std::string & _name)

Constructor.

Parameters

in	<code>_name</code>	Name of the timer
----	--------------------	-------------------

10.38.2.2 virtual gazebo::util::DiagnosticTimer::~~DiagnosticTimer () [virtual]

Destructor.

10.38.3 Member Function Documentation

10.38.3.1 const std::string gazebo::util::DiagnosticTimer::GetName () const [inline]

Get the name of the timer.

Returns

The name of timer

10.38.3.2 void gazebo::util::DiagnosticTimer::Lap (const std::string & _prefix)

Output a lap time.

Parameters

in	<code>_prefix</code>	Annotation to output with the elapsed time.
----	----------------------	---

10.38.3.3 virtual void gazebo::util::DiagnosticTimer::Start () [virtual]

Start the timer.

Reimplemented from **gazebo::common::Timer** (p. 799).

10.38.3.4 virtual void gazebo::util::DiagnosticTimer::Stop () [virtual]

Stop the timer.

Reimplemented from **gazebo::common::Timer** (p. 799).

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

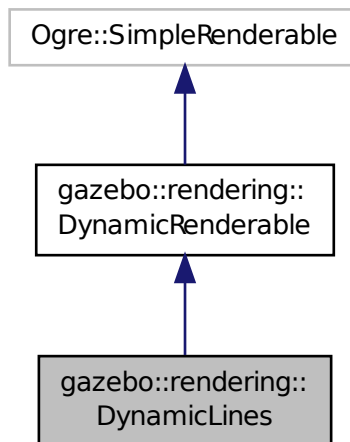
- **Diagnostics.hh**

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)
Add a point to the point list.
- void **AddPoint** (double _x, double _y, double _z)
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. 106)'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 **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.

Protected Member Functions

- virtual void **CreateVertexDeclaration** ()
*Implementation **DynamicRenderable** (p. 265), creates a simple vertex-only decl.*
- virtual void **FillHardwareBuffers** ()
*Implementation **DynamicRenderable** (p. 265), pushes point list out to hardware memory.*

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

<code>in</code>	<code>_opType</code>	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)`

Add a point to the point list.

Parameters

in	<i>pt</i>	math::Vector3 (p. 839) point
----	-----------	-------------------------------------

10.39.3.2 `void gazebo::rendering::DynamicLines::AddPoint (double _x, double _y, double _z)`

Add a point to the point list.

Parameters

in	_x	X position.
in	_y	Y position.
in	_z	Z position.

10.39.3.3 `void gazebo::rendering::DynamicLines::Clear ()`

Remove all points from the point list.

10.39.3.4 `virtual void gazebo::rendering::DynamicLines::CreateVertexDeclaration () [protected],[virtual]`

Implementation **DynamicRenderable** (p. 265), creates a simple vertex-only decl.

Implements **gazebo::rendering::DynamicRenderable** (p. 267).

10.39.3.5 `virtual void gazebo::rendering::DynamicLines::FillHardwareBuffers () [protected],[virtual]`

Implementation **DynamicRenderable** (p. 265), pushes point list out to hardware memory.

Implements **gazebo::rendering::DynamicRenderable** (p. 267).

10.39.3.6 `static std::string gazebo::rendering::DynamicLines::GetMovableType () [static]`

Get type of movable.

Returns

This returns "gazebo::dynamiclines"

10.39.3.7 `virtual const Ogre::String& gazebo::rendering::DynamicLines::getMovableType () const` [virtual]

Overridden function from **Ogre** (p. 106)'s base class.

Returns

Returns "gazebo::ogredynamicslines"

10.39.3.8 `const math::Vector3& gazebo::rendering::DynamicLines::GetPoint (unsigned int _index) const`

Return the location of an existing point in the point list.

Parameters

<i>in</i>	<i>_index</i>	Number of the point to return
-----------	---------------	-------------------------------

Returns

math::Vector3 (p. 839) value of the point

10.39.3.9 `unsigned int gazebo::rendering::DynamicLines::GetPointCount () const`

Return the total number of points in the point list.

Returns

Number of points

10.39.3.10 `void gazebo::rendering::DynamicLines::SetPoint (unsigned int _index, const math::Vector3 & _value)`

Change the location of an existing point in the point list.

Parameters

<i>in</i>	<i>_index</i>	Index of the point to set
<i>in</i>	<i>_value</i>	math::Vector3 (p. 839) value to set the point to

10.39.3.11 `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:

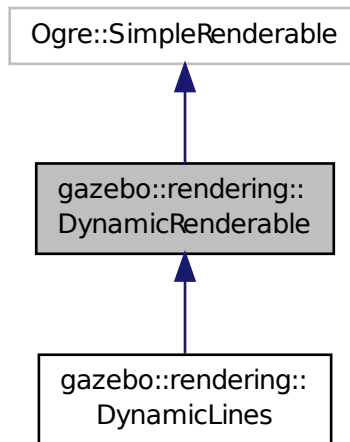
- **DynamicLines.hh**

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.
- **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.

Implemented in `gazebo::rendering::DynamicLines` (p. 264).

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.

Implemented in `gazebo::rendering::DynamicLines` (p. 264).

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 `RenderOpType gazebo::rendering::DynamicRenderable::GetOperationType () const`

Get the render operation type.

Returns

The render operation type.

10.40.3.5 `virtual Ogre::Real gazebo::rendering::DynamicRenderable::getSquaredViewDepth (const Ogre::Camera * _cam) const` [virtual]

Implementation of `Ogre::SimpleRenderable`.

Parameters

<code>in</code>	<code>_cam</code>	Pointer to the Ogre (p. 106) camera that views the renderable.
-----------------	-------------------	---

Returns

The squared depth in the **Camera** (p. 159)'s view

10.40.3.6 `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. 267).

Parameters

<code>in</code>	<code>_opType</code>	The type of render operation to perform.
<code>in</code>	<code>_useIndices</code>	Specifies whether to use indices to determine the vertices to use as input.

10.40.3.7 `void gazebo::rendering::DynamicRenderable::PrepareHardwareBuffers (size_t _vertexCount, size_t _indexCount)` [protected]

Prepares the hardware buffers for the requested vertex and index counts.

Remarks

This function must be called before locking the buffers in `fillHardwareBuffers()`. It guarantees that the hardware buffers are large enough to hold at least the requested number of vertices and indices (if using indices). The buffers are possibly reallocated to achieve this.

The vertex and index count in the render operation are set to

the values of `vertexCount` and `indexCount` respectively.

Parameters

<code>in</code>	<code>_vertexCount</code>	The number of vertices the buffer must hold.
<code>in</code>	<code>_indexCount</code>	The number of indices the buffer must hold. This parameter is ignored if not using indices.

10.40.3.8 `void gazebo::rendering::DynamicRenderable::SetOperationType (RenderOpType _opType)`

Set the render operation type.

Parameters

<code>in</code>	<code>_opType</code>	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:

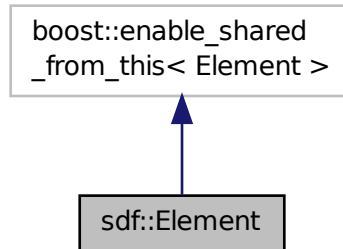
- `DynamicRenderable.hh`

10.41 sdf::Element Class Reference

SDF (p. 680) **Element** (p. 269) class.

```
#include <SDF.hh>
```

Inheritance diagram for sdf::Element:



Public Member Functions

- **Element** ()
- virtual **~Element** ()
- void **AddAttribute** (const std::string &_key, const std::string &_type, const std::string &_defaultvalue, bool _required, const std::string &_description="")
- **ElementPtr AddElement** (const std::string &_name)
- void **AddElementDescription** (**ElementPtr** _elem)
 - Add a new element description.*
- void **AddValue** (const std::string &_type, const std::string &_defaultValue, bool _required, const std::string &_description="")
- void **ClearElements** ()
 - Remove all child elements.*
- **Element * Clone** () const
- void **Copy** (const **ElementPtr** _elem)
 - Copy values from an **Element** (p. 269).*
- **ParamPtr GetAttribute** (const std::string &_key)
 - Get the param of an attribute.*
- **ParamPtr GetAttribute** (unsigned int _index) const
 - Get an attribute using an index.*
- unsigned int **GetAttributeCount** () const
 - Get the number of attributes.*
- bool **GetAttributeSet** (const std::string &_key)
 - Return true if the attribute was set (i.e. not default value)*
- bool **GetCopyChildren** () const
- std::string **GetDescription** () const
 - Get a text description of the element.*
- **ElementPtr GetElement** (const std::string &_name) const
- **ElementPtr GetElement** (const std::string &_name)
- **ElementPtr GetElementDescription** (unsigned int _index) const
 - Get an element description using an index.*

- **ElementPtr GetElementDescription** (const std::string &_key) const
Get an element descriptio using a key.
- unsigned int **GetElementDescriptionCount** () const
Get the number of element descriptions.
- **ElementPtr GetElementImpl** (const std::string &_name) const
- **ElementPtr GetFirstElement** () const
- std::string **GetInclude** () const
- const std::string & **GetName** () const
- **ElementPtr GetNextElement** (const std::string &_name="") const
- **ElementPtr GetParent** () const
- const std::string & **GetRequired** () const
- **ParamPtr GetValue** ()
Get the param of the elements value.
- bool **GetValueBool** (const std::string &_key="")
- char **GetValueChar** (const std::string &_key="")
- **gazebo::common::Color GetValueColor** (const std::string &_key="")
- double **GetValueDouble** (const std::string &_key="")
- float **GetValueFloat** (const std::string &_key="")
- int **GetValueInt** (const std::string &_key="")
- **gazebo::math::Pose GetValuePose** (const std::string &_key="")
- **gazebo::math::Quaternion GetValueQuaternion** (const std::string &_key="")
- std::string **GetValueString** (const std::string &_key="")
- **gazebo::common::Time GetValueTime** (const std::string &_key="")
- unsigned int **GetValueUInt** (const std::string &_key="")
- **gazebo::math::Vector2d GetValueVector2d** (const std::string &_key="")
- **gazebo::math::Vector3 GetValueVector3** (const std::string &_key="")
- bool **HasAttribute** (const std::string &_key)
- bool **HasElement** (const std::string &_name) const
- bool **HasElementDescription** (const std::string &_name)
Return true if an element description exists.
- void **InsertElement** (**ElementPtr** _elem)
- void **PrintDescription** (std::string _prefix)
- void **PrintDocLeftPane** (std::string &_html, int _spacing, int &_index)
*Helper function for **SDF::PrintDoc** (p. 681).*
- void **PrintDocRightPane** (std::string &_html, int _spacing, int &_index)
*Helper function for **SDF::PrintDoc** (p. 681).*
- void **PrintValues** (std::string _prefix)
- void **PrintWiki** (std::string _prefix)
- void **RemoveChild** (**ElementPtr** _child)
Remove a child element.
- void **RemoveFromParent** ()
Remove this element from its parent.
- void **Reset** ()
- bool **Set** (const bool &_value)
- bool **Set** (const int &_value)
- bool **Set** (const unsigned int &_value)
- bool **Set** (const float &_value)
- bool **Set** (const double &_value)
- bool **Set** (const char &_value)

- bool **Set** (const std::string &_value)
- bool **Set** (const char *_value)
- bool **Set** (const gazebo::math::Vector3 &_value)
- bool **Set** (const gazebo::math::Vector2i &_value)
- bool **Set** (const gazebo::math::Vector2d &_value)
- bool **Set** (const gazebo::math::Quaternion &_value)
- bool **Set** (const gazebo::math::Pose &_value)
- bool **Set** (const gazebo::common::Color &_value)
- bool **Set** (const gazebo::common::Time &_value)
- void **SetCopyChildren** (bool _value)
- void **SetDescription** (const std::string &_desc)
 - Set a text description for the element.*
- void **SetInclude** (const std::string &_filename)
- void **SetName** (const std::string &_name)
- void **SetParent** (const ElementPtr _parent)
- void **SetRequired** (const std::string &_req)
- std::string **ToString** (const std::string &_prefix) const
- void **Update** ()

10.41.1 Detailed Description

SDF (p. 680) **Element** (p. 269) class.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 sdf::Element::Element ()

10.41.2.2 virtual sdf::Element::~~Element () [virtual]

10.41.3 Member Function Documentation

10.41.3.1 void sdf::Element::AddAttribute (const std::string &_key, const std::string &_type, const std::string &_defaultvalue, bool _required, const std::string &_description = " ")

10.41.3.2 ElementPtr sdf::Element::AddElement (const std::string &_name)

10.41.3.3 void sdf::Element::AddElementDescription (ElementPtr _elem)

Add a new element description.

10.41.3.4 void sdf::Element::AddValue (const std::string &_type, const std::string &_defaultValue, bool _required, const std::string &_description = " ")

10.41.3.5 void sdf::Element::ClearElements ()

Remove all child elements.

10.41.3.6 `Element*` `sdf::Element::Clone ()` `const`

10.41.3.7 `void` `sdf::Element::Copy (const ElementPtr _elem)`

Copy values from an **Element** (p. 269).

10.41.3.8 `ParamPtr` `sdf::Element::GetAttribute (const std::string & _key)`

Get the param of an attribute.

Parameters

<code>_key</code>	the name of the attribute
-------------------	---------------------------

10.41.3.9 `ParamPtr` `sdf::Element::GetAttribute (unsigned int _index)` `const`

Get an attribute using an index.

10.41.3.10 `unsigned int` `sdf::Element::GetAttributeCount ()` `const`

Get the number of attributes.

10.41.3.11 `bool` `sdf::Element::GetAttributeSet (const std::string & _key)`

Return true if the attribute was set (i.e. not default value)

10.41.3.12 `bool` `sdf::Element::GetCopyChildren ()` `const`

10.41.3.13 `std::string` `sdf::Element::GetDescription ()` `const`

Get a text description of the element.

10.41.3.14 `ElementPtr` `sdf::Element::GetElement (const std::string & _name)` `const`

Referenced by `gazebo::physics::ScrewJoint< T >::Load()`, and `gazebo::physics::Hinge2Joint< T >::Load()`.

10.41.3.15 `ElementPtr` `sdf::Element::GetElement (const std::string & _name)`

10.41.3.16 `ElementPtr` `sdf::Element::GetElementDescription (unsigned int _index)` `const`

Get an element description using an index.

10.41.3.17 `ElementPtr` `sdf::Element::GetElementDescription (const std::string & _key)` `const`

Get an element descriptio using a key.

10.41.3.18 `unsigned int sdf::Element::GetElementDescriptionCount () const`

Get the number of element descriptions.

10.41.3.19 `ElementPtr sdf::Element::GetElementImpl (const std::string & _name) const`

10.41.3.20 `ElementPtr sdf::Element::GetFirstElement () const`

10.41.3.21 `std::string sdf::Element::GetInclude () const`

10.41.3.22 `const std::string& sdf::Element::GetName () const`

10.41.3.23 `ElementPtr sdf::Element::GetNextElement (const std::string & _name = " ") const`

10.41.3.24 `ElementPtr sdf::Element::GetParent () const`

10.41.3.25 `const std::string& sdf::Element::GetRequired () const`

10.41.3.26 `ParamPtr sdf::Element::GetValue ()`

Get the param of the elements value.

10.41.3.27 `bool sdf::Element::GetValueBool (const std::string & _key = " ")`

10.41.3.28 `char sdf::Element::GetValueChar (const std::string & _key = " ")`

10.41.3.29 `gazebo::common::Color sdf::Element::GetValueColor (const std::string & _key = " ")`

10.41.3.30 `double sdf::Element::GetValueDouble (const std::string & _key = " ")`

Referenced by `gazebo::physics::ScrewJoint< T >::Load()`.

10.41.3.31 `float sdf::Element::GetValueFloat (const std::string & _key = " ")`

10.41.3.32 `int sdf::Element::GetValueInt (const std::string & _key = " ")`

10.41.3.33 `gazebo::math::Pose sdf::Element::GetValuePose (const std::string & _key = " ")`

10.41.3.34 `gazebo::math::Quaternion sdf::Element::GetValueQuaternion (const std::string & _key = " ")`

10.41.3.35 `std::string sdf::Element::GetValueString (const std::string & _key = " ")`

10.41.3.36 `gazebo::common::Time sdf::Element::GetValueTime (const std::string & _key = " ")`

10.41.3.37 `unsigned int sdf::Element::GetValueUInt (const std::string & _key = " ")`

10.41.3.38 `gazebo::math::Vector2d sdf::Element::GetValueVector2d (const std::string & _key = " ")`

10.41.3.39 `gazebo::math::Vector3 sdf::Element::GetValueVector3 (const std::string & _key = " ")`

Referenced by `gazebo::physics::ScrewJoint< T >::Load()`, and `gazebo::physics::Hinge2Joint< T >::Load()`.

10.41.3.40 `bool sdf::Element::HasAttribute (const std::string & _key)`

10.41.3.41 `bool sdf::Element::HasElement (const std::string & _name) const`

Referenced by `gazebo::physics::ScrewJoint< T >::Load()`.

10.41.3.42 `bool sdf::Element::HasElementDescription (const std::string & _name)`

Return true if an element description exists.

10.41.3.43 `void sdf::Element::InsertElement (ElementPtr _elem)`

10.41.3.44 `void sdf::Element::PrintDescription (std::string _prefix)`

10.41.3.45 `void sdf::Element::PrintDocLeftPane (std::string & _html, int _spacing, int & _index)`

Helper function for **SDF::PrintDoc** (p. 681).

This generates the **SDF** (p. 680) html documentation.

Parameters

out	<i>_html</i>	Accumulated HTML for output.
in	<i>_spacing</i>	Amount of spacing for this element.
in	<i>_index</i>	Unique index for this element.

10.41.3.46 `void sdf::Element::PrintDocRightPane (std::string & _html, int _spacing, int & _index)`

Helper function for **SDF::PrintDoc** (p. 681).

This generates the **SDF** (p. 680) html documentation.

Parameters

out	<i>_html</i>	Accumulated HTML for output
in	<i>_spacing</i>	Amount of spacing for this element.

10.41.3.47 `void sdf::Element::PrintValues (std::string _prefix)`

10.41.3.48 `void sdf::Element::PrintWiki (std::string _prefix)`

10.41.3.49 `void sdf::Element::RemoveChild (ElementPtr _child)`

Remove a child element.

Parameters

in	<i>_child</i>	Pointer to the child to remove.
----	---------------	---------------------------------

10.41.3.50 void sdf::Element::RemoveFromParent ()

Remove this element from its parent.

10.41.3.51 void sdf::Element::Reset ()

10.41.3.52 bool sdf::Element::Set (const bool & *_value*)

10.41.3.53 bool sdf::Element::Set (const int & *_value*)

10.41.3.54 bool sdf::Element::Set (const unsigned int & *_value*)

10.41.3.55 bool sdf::Element::Set (const float & *_value*)

10.41.3.56 bool sdf::Element::Set (const double & *_value*)

10.41.3.57 bool sdf::Element::Set (const char & *_value*)

10.41.3.58 bool sdf::Element::Set (const std::string & *_value*)

10.41.3.59 bool sdf::Element::Set (const char * *_value*)

10.41.3.60 bool sdf::Element::Set (const gazebo::math::Vector3 & *_value*)

10.41.3.61 bool sdf::Element::Set (const gazebo::math::Vector2i & *_value*)

10.41.3.62 bool sdf::Element::Set (const gazebo::math::Vector2d & *_value*)

10.41.3.63 bool sdf::Element::Set (const gazebo::math::Quaternion & *_value*)

10.41.3.64 bool sdf::Element::Set (const gazebo::math::Pose & *_value*)

10.41.3.65 bool sdf::Element::Set (const gazebo::common::Color & *_value*)

10.41.3.66 bool sdf::Element::Set (const gazebo::common::Time & *_value*)

10.41.3.67 void sdf::Element::SetCopyChildren (bool *_value*)

10.41.3.68 void sdf::Element::SetDescription (const std::string & *_desc*)

Set a text description for the element.

10.41.3.69 void sdf::Element::SetInclude (const std::string & *_filename*)

10.41.3.70 void sdf::Element::SetName (const std::string & *_name*)

10.41.3.71 void sdf::Element::SetParent (const ElementPtr *_parent*)

10.41.3.72 void sdf::Element::SetRequired (const std::string & *_req*)

10.41.3.73 `std::string sdf::Element::ToString (const std::string & _prefix) const`

10.41.3.74 `void sdf::Element::Update ()`

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

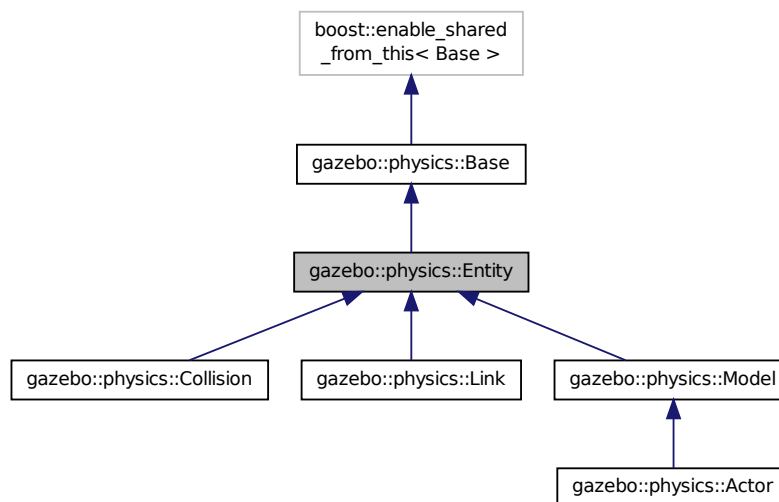
- **SDF.hh**

10.42 gazebo::physics::Entity Class Reference

Base (p. 135) class for all physics objects in Gazebo.

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

Inheritance diagram for gazebo::physics::Entity:



Public Member Functions

- **Entity** (**BasePtr** _parent)
Constructor.
- virtual **~Entity** ()
Destructor.
- virtual void **Fini** ()
Finalize the entity.
- virtual **math::Box GetBoundingBox** () const
Return the bounding box for the entity.
- **CollisionPtr GetChildCollision** (const std::string &_name)
Get a child collision entity, if one exists.
- **LinkPtr GetChildLink** (const std::string &_name)

- Get a child linke entity, if one exists.*

 - **math::Box GetCollisionBoundingBox** () const

Returns collision bounding box.
- const **math::Pose & GetDirtyPose** () const

*Returns **Entity::dirtyPose** (p. 288).*
- **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** (p. 109) 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. 277).*
- 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 (p. 109) helper that prevents numerous dynamic_casts.
- msgs::Pose * poseMsg
Pose message container.
- **common::Time** prevAnimationTime
Previous time an animation was updated.
- **transport::PublisherPtr** requestPub
Request publisher.

- **transport::PublisherPtr visPub**

Visual publisher.

- **msgs::Visual * visualMsg**

Visual message container.

Additional Inherited Members

10.42.1 Detailed Description

Base (p. 135) class for all physics objects in Gazebo.

10.42.2 Constructor & Destructor Documentation

10.42.2.1 gazebo::physics::Entity::Entity (BasePtr _parent) [explicit]

Constructor.

Parameters

in	<code>_parent</code>	Parent of the entity.
----	----------------------	-----------------------

10.42.2.2 virtual gazebo::physics::Entity::~Entity () [virtual]

Destructor.

10.42.3 Member Function Documentation

10.42.3.1 virtual void gazebo::physics::Entity::Fini () [virtual]

Finalize the entity.

Reimplemented from **gazebo::physics::Base** (p. 140).

Reimplemented in **gazebo::physics::Actor** (p. 112), **gazebo::physics::Model** (p. 485), **gazebo::physics::Link** (p. 419), and **gazebo::physics::Collision** (p. 195).

10.42.3.2 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. 420), **gazebo::physics::Model** (p. 485), and **gazebo::physics::Collision** (p. 195).

10.42.3.3 CollisionPtr gazebo::physics::Entity::GetChildCollision (const std::string & _name)

Get a child collision entity, if one exists.

Parameters

in	_name	Name of the child collision object.
----	-------	-------------------------------------

Returns

Pointer to the **Collision** (p. 192) object, or NULL if not found.

10.42.3.4 LinkPtr gazebo::physics::Entity::GetChildLink (const std::string & _name)

Get a child linked entity, if one exists.

Parameters

in	_name	Name of the child Link (p. 412) object.
----	-------	--

Returns

Pointer to the **Link** (p. 412) object, or NULL if not found.

10.42.3.5 math::Box gazebo::physics::Entity::GetCollisionBoundingBox () const

Returns collision bounding box.

Returns

Collision bounding box.

10.42.3.6 const math::Pose& gazebo::physics::Entity::GetDirtyPose () const

Returns **Entity::dirtyPose** (p. 288).

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.

10.42.3.7 math::Pose gazebo::physics::Entity::GetInitialRelativePose () const

Get the initial relative pose.

Returns

The initial relative pose.

10.42.3.8 `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.

10.42.3.9 `ModelPtr gazebo::physics::Entity::GetParentModel ()`

Get the parent model, if one exists.

Returns

Pointer to a model, or NULL if no parent model exists.

10.42.3.10 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularAccel () const [inline],[virtual]`

Get the angular acceleration of the entity.

Returns

A (p. 109) `math::Vector3` (p. 839) for the acceleration.

Reimplemented in `gazebo::physics::Link` (p. 422), `gazebo::physics::Collision` (p. 196), and `gazebo::physics::Model` (p. 487).

10.42.3.11 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularVel () const [inline],[virtual]`

Get the angular velocity of the entity.

Returns

A (p. 109) `math::Vector3` (p. 839) for the velocity.

Reimplemented in `gazebo::physics::Link` (p. 422), `gazebo::physics::Collision` (p. 196), and `gazebo::physics::Model` (p. 487).

10.42.3.12 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearAccel () const [inline],[virtual]`

Get the linear acceleration of the entity.

Returns

A (p. 109) `math::Vector3` (p. 839) for the acceleration.

Reimplemented in `gazebo::physics::Link` (p. 423), `gazebo::physics::Collision` (p. 197), and `gazebo::physics::Model` (p. 487).

10.42.3.13 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearVel () const [inline],[virtual]`

Get the linear velocity of the entity.

Returns

A (p. 109) `math::Vector3` (p. 839) for the linear velocity.

Reimplemented in `gazebo::physics::Link` (p. 423), `gazebo::physics::Collision` (p. 197), and `gazebo::physics::Model` (p. 488).

10.42.3.14 `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.

10.42.3.15 `virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularAccel () const [inline],[virtual]`

Get the angular acceleration of the entity in the world frame.

Returns

A (p. 109) `math::Vector3` (p. 839) for the acceleration.

Reimplemented in `gazebo::physics::Link` (p. 424), `gazebo::physics::Collision` (p. 198), and `gazebo::physics::Model` (p. 488).

10.42.3.16 `virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularVel () const [inline],[virtual]`

Get the angular velocity of the entity in the world frame.

Returns

A (p. 109) `math::Vector3` (p. 839) for the velocity.

Reimplemented in `gazebo::physics::Collision` (p. 198), and `gazebo::physics::Model` (p. 488).

10.42.3.17 `virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearAccel () const [inline],[virtual]`

Get the linear acceleration of the entity in the world frame.

Returns

A (p. 109) `math::Vector3` (p. 839) for the acceleration.

Reimplemented in `gazebo::physics::Link` (p. 425), `gazebo::physics::Collision` (p. 198), and `gazebo::physics::Model` (p. 489).

10.42.3.18 `virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearVel () const [inline],[virtual]`

Get the linear velocity of the entity in the world frame.

Returns

A (p. 109) **math::Vector3** (p. 839) for the linear velocity.

Reimplemented in **gazebo::physics::Collision** (p. 198), and **gazebo::physics::Model** (p. 489).

10.42.3.19 `const math::Pose& gazebo::physics::Entity::GetWorldPose () const [inline]`

Get the absolute pose of the entity.

Returns

The absolute pose of the entity.

Referenced by `gazebo::sensors::RFIDTag::GetTagPose()`.

10.42.3.20 `bool gazebo::physics::Entity::IsCanonicalLink () const [inline]`

A (p. 109) helper function that checks if this is a canonical body.

Returns

True if the link is canonical.

10.42.3.21 `bool gazebo::physics::Entity::IsStatic () const`

Return whether this entity is static.

Returns

True if static.

10.42.3.22 `virtual void gazebo::physics::Entity::Load (sdf::ElementPtr _sdf) [virtual]`

Load the entity.

Parameters

<code>in</code>	<code>_sdf</code>	Pointer to an SDF element.
-----------------	-------------------	----------------------------

Reimplemented from **gazebo::physics::Base** (p. 143).

Reimplemented in **gazebo::physics::Actor** (p. 113), **gazebo::physics::Model** (p. 489), **gazebo::physics::Link** (p. 426), and **gazebo::physics::Collision** (p. 199).

10.42.3.23 `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. 426), and `gazebo::physics::Model` (p. 489).

10.42.3.24 `void gazebo::physics::Entity::PlaceOnEntity (const std::string & _entityName)`

Move this entity to be ontop of another entity by name.

Parameters

<code>in</code>	<code>_entityName</code>	Name of the Entity (p. 277) this Entity (p. 277) should be ontop of.
-----------------	--------------------------	--

10.42.3.25 `void gazebo::physics::Entity::PlaceOnNearestEntityBelow ()`

Move this entity to be ontop of the nearest entity below.

10.42.3.26 `virtual void gazebo::physics::Entity::Reset () [virtual]`

Reset the entity.

Reimplemented from `gazebo::physics::Base` (p. 144).

Reimplemented in `gazebo::physics::Model` (p. 490), and `gazebo::physics::Link` (p. 426).

10.42.3.27 `void gazebo::physics::Entity::SetAnimation (const common::PoseAnimationPtr & _anim, boost::function< void()> _onComplete)`

Set an animation for this entity.

Parameters

<code>in</code>	<code>_anim</code>	Pose animation.
<code>in</code>	<code>_onComplete</code>	Callback for when the animation completes.

10.42.3.28 `void gazebo::physics::Entity::SetAnimation (common::PoseAnimationPtr _anim)`

Set an animation for this entity.

Parameters

<code>in</code>	<code>_anim</code>	Pose animation.
-----------------	--------------------	-----------------

10.42.3.29 `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.
----	---------------------	--------------------------------

10.42.3.30 `void gazebo::physics::Entity::SetInitialRelativePose (const math::Pose & _pose)`

Set the initial pose.

Parameters

in	<code>_pose</code>	The initial pose.
----	--------------------	-------------------

10.42.3.31 `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. 145).

10.42.3.32 `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.

10.42.3.33 `void gazebo::physics::Entity::SetStatic (const bool & _static)`

Set whether this entity is static: immovable.

Parameters

in	<code>_static</code>	True = static.
----	----------------------	----------------

10.42.3.34 `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.
----	-----------------------	---------------------------

10.42.3.35 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. 277).

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.

10.42.3.36 virtual void gazebo::physics::Entity::StopAnimation () [virtual]

Stop the current animation, if any.

Reimplemented in **gazebo::physics::Model** (p. 493).

10.42.3.37 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. 146).

Reimplemented in **gazebo::physics::Actor** (p. 113), **gazebo::physics::Model** (p. 493), **gazebo::physics::Link** (p. 430), and **gazebo::physics::Collision** (p. 200).

10.42.4 Member Data Documentation

10.42.4.1 common::PoseAnimationPtr gazebo::physics::Entity::animation [protected]

Current pose animation.

10.42.4.2 event::ConnectionPtr gazebo::physics::Entity::animationConnection [protected]

Connection used to update an animation.

10.42.4.3 math::Pose gazebo::physics::Entity::animationStartPose [protected]

Start pose of an animation.

10.42.4.4 `std::vector<event::ConnectionPtr> gazebo::physics::Entity::connections` [protected]

All our event connections.

10.42.4.5 `math::Pose gazebo::physics::Entity::dirtyPose` [protected]

The pose set by a physics engine.

10.42.4.6 `transport::NodePtr gazebo::physics::Entity::node` [protected]

Communication node.

10.42.4.7 `EntityPtr gazebo::physics::Entity::parentEntity` [protected]

A (p. 109) helper that prevents numerous `dynamic_casts`.

10.42.4.8 `msgs::Pose* gazebo::physics::Entity::poseMsg` [protected]

Pose message container.

10.42.4.9 `common::Time gazebo::physics::Entity::prevAnimationTime` [protected]

Previous time an animation was updated.

10.42.4.10 `transport::PublisherPtr gazebo::physics::Entity::requestPub` [protected]

Request publisher.

10.42.4.11 `transport::PublisherPtr gazebo::physics::Entity::visPub` [protected]

Visual publisher.

10.42.4.12 `msgs::Visual* gazebo::physics::Entity::visualMsg` [protected]

Visual message container.

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

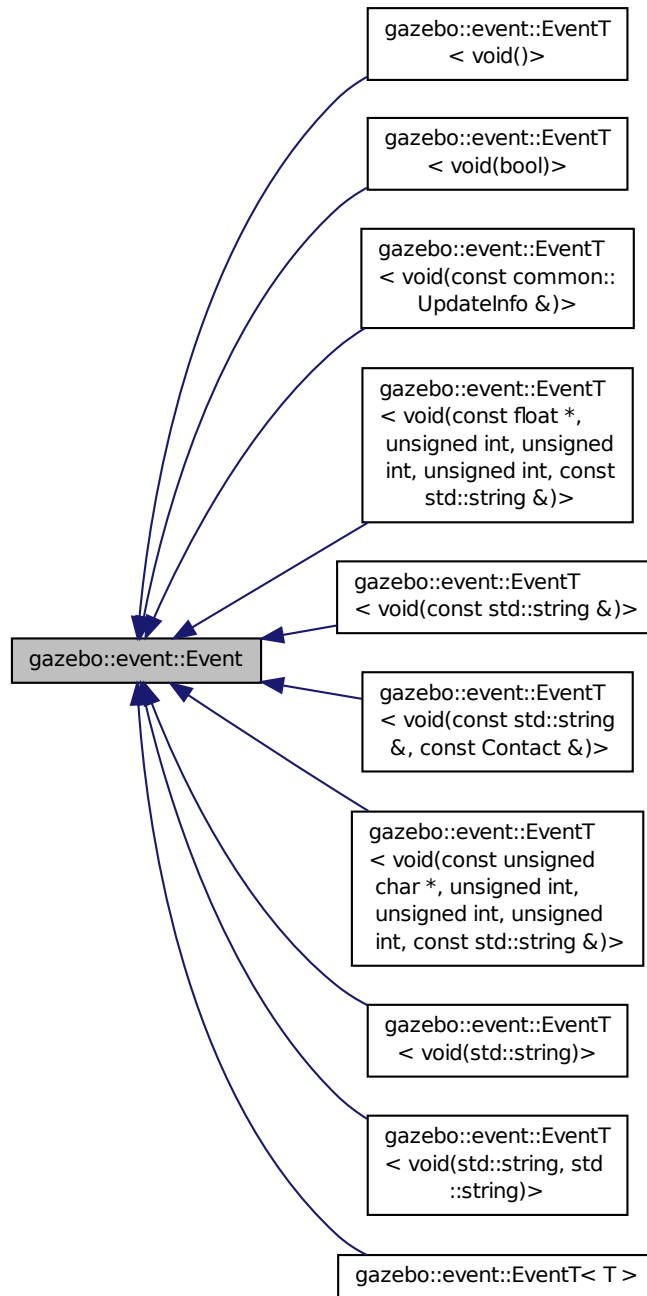
- **Entity.hh**

10.43 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.

10.43.1 Detailed Description

Base class for all events.

10.43.2 Constructor & Destructor Documentation

10.43.2.1 virtual gazebo::event::Event::~Event () [inline],[virtual]

Constructor.

10.43.3 Member Function Documentation

10.43.3.1 virtual void gazebo::event::Event::Disconnect (**ConnectionPtr** _c) [pure virtual]

Disconnect.

Parameters

in	_c	A (p. 109) pointer to a connection
----	----	------------------------------------

Implemented in **gazebo::event::EventT< T >** (p. 39), **gazebo::event::EventT< void(std::string)>** (p. 39), **gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 39), **gazebo::event::EventT< void(const std::string &)>** (p. 39), **gazebo::event::EventT< void()>** (p. 39), **gazebo::event::EventT< void(const common::UpdateInfo &)>** (p. 39), **gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 39), **gazebo::event::EventT< void(const std::string &, const Contact &)>** (p. 39), **gazebo::event::EventT< void(std::string, std::string)>** (p. 39), and **gazebo::event::EventT< void(bool)>** (p. 39).

10.43.3.2 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. 40), **gazebo::event::EventT< void(std::string)>** (p. 40), **gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 40), **gazebo::event::EventT< void(const std::string &)>** (p. 40), **gazebo::event::EventT< void()>** (p. 40), **gazebo::event::EventT< void(const common::UpdateInfo &)>** (p. 40), **gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 40), **gazebo::event::EventT< void(const std::string &, const Contact &)>** (p. 40), **gazebo::event::EventT< void(std::string, std::string)>** (p. 40), and **gazebo::event::EventT< void(bool)>** (p. 40).

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

- [Event.hh](#)

10.44 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.

10.44.1 Detailed Description

Base class for rendering events.

10.44.2 Member Function Documentation

10.44.2.1 `template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectCreateScene (T _subscriber) [inline],[static]`

Connect to a scene created event.

Parameters

in	_subscriber	Callback to trigger when event occurs.
----	-------------	--

Returns

Pointer the connection. This must stay in scope.

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

10.44.2.2 `template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectRemoveScene (T
_subscriber) [inline],[static]`

Connect to a scene removed event.

Parameters

in	<code>_subscriber</code>	Callback to trigger when event occurs.
----	--------------------------	--

Returns

Pointer the connection. This must stay in scope.

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

10.44.2.3 `static void gazebo::rendering::Events::DisconnectCreateScene (event::ConnectionPtr _connection) [inline],
[static]`

Disconnect from a scene created event.

Parameters

in	<code>_connection</code>	The connection to disconnect.
----	--------------------------	-------------------------------

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

10.44.2.4 `static void gazebo::rendering::Events::DisconnectRemoveScene (event::ConnectionPtr _connection)
[inline],[static]`

Disconnect from a scene removed event.

Parameters

in	<code>_connection</code>	The connection to disconnect.
----	--------------------------	-------------------------------

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

10.44.3 Member Data Documentation

10.44.3.1 `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().

10.44.3.2 `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**

10.45 gazebo::event::Events Class Reference

An **Event** (p. 288) 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 to the add entity signal.
- `template<typename T >`
static ConnectionPtr ConnectCreateEntity (T _subscriber)
Connect a boost::slot to the add entity signal.
- `template<typename T >`
static ConnectionPtr ConnectDeleteEntity (T _subscriber)
Connect a boost::slot to the delete entity.
- `template<typename T >`
static ConnectionPtr ConnectDiagTimerStart (T _subscriber)
Connect a boost::slot to the diagnostic timer start signal.
- `template<typename T >`
static ConnectionPtr ConnectDiagTimerStop (T _subscriber)
Connect a boost::slot to the diagnostic timer stop signal.
- `template<typename T >`
static ConnectionPtr ConnectPause (T _subscriber)
Connect a boost::slot to the pause signal.
- `template<typename T >`
static ConnectionPtr ConnectPostRender (T _subscriber)
Connect a boost::slot to 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 to the render update signal.
- `template<typename T >`
static ConnectionPtr ConnectSetSelectedEntity (T _subscriber)
Connect a boost::slot to the set selected entity.
- `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.
 - `template<typename T >`
 static **ConnectionPtr ConnectWorldUpdateStart** (T _subscriber)
 Connect a boost::slot the the world update start 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 **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 void **DisconnectWorldUpdateStart** (ConnectionPtr _subscriber) **GAZEBO_DEPRECATED**
 Disconnect a boost::slot the the world update start 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()> **step**
Step the simulation once signal.
- static **EventT**< void()> **stop**
Simulation stop signal.
- static **EventT**< void(std::string)> **worldCreated**
A (p. 109) world has been created.
- static **EventT**< void(const **common::UpdateInfo** &)> **worldUpdateBegin**
World update has started.
- static **EventT**< void()> **worldUpdateEnd**
World update has ended.
- static **EventT**< void()> **worldUpdateStart**
World update has started.

10.45.1 Detailed Description

An **Event** (p. 288) class to get notifications for simulator events.

10.45.2 Member Function Documentation

10.45.2.1 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectAddEntity (T _subscriber) [inline], [static]`

Connect a boost::slot the the add entity signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

10.45.2.2 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectCreateEntity (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the the add entity signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

10.45.2.3 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDeleteEntity (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the delete entity.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

10.45.2.4 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStart (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the diagnostic timer start signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

10.45.2.5 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStop (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the diagnostic timer stop signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

Returns

a connection

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

10.45.2.6 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPause (T _subscriber)` `[inline],`
`[static]`

Connect a boost::slot the the pause 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 pause.

10.45.2.7 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPostRender (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the post render update signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

Returns

a connection

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

10.45.2.8 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPreRender (T _subscriber)`
`[inline],[static]`

Render start signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

10.45.2.9 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectRender (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the render update signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

10.45.2.10 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectSetSelectedEntity (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the set selected entity.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

10.45.2.11 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStep (T _subscriber)` `[inline],`
`[static]`

Connect a boost::slot the the step signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

10.45.2.12 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStop (T _subscriber)` `[inline], [static]`

Connect a boost::slot the the stop 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 stop.

10.45.2.13 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldCreated (T _subscriber)` `[inline], [static]`

Connect a boost::slot the the world created 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 worldCreated.

10.45.2.14 `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.

10.45.2.15 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldUpdateEnd (T _subscriber)` `[inline], [static]`

Connect a boost::slot the the world update end signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

10.45.2.16 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldUpdateStart (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the the world update start signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

10.45.2.17 `static void gazebo::event::Events::DisconnectAddEntity (ConnectionPtr _subscriber)` `[inline],[static]`

Disconnect a boost::slot the the add entity signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

10.45.2.18 `static void gazebo::event::Events::DisconnectCreateEntity (ConnectionPtr _subscriber)` `[inline],[static]`

Disconnect a boost::slot the the add entity signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

10.45.2.19 `static void gazebo::event::Events::DisconnectDeleteEntity (ConnectionPtr _subscriber)` `[inline],[static]`

Disconnect a boost::slot the delete entity.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

10.45.2.20 static void gazebo::event::Events::DisconnectDiagTimerStart (ConnectionPtr *_subscriber*) [inline],
[static]

Disconnect a boost::slot the diagnostic timer start signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

10.45.2.21 static void gazebo::event::Events::DisconnectDiagTimerStop (ConnectionPtr *_subscriber*) [inline],
[static]

Disconnect a boost::slot the diagnostic timer stop signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

10.45.2.22 static void gazebo::event::Events::DisconnectPause (ConnectionPtr *_subscriber*) [inline], [static]

Disconnect a boost::slot the the pause signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

10.45.2.23 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.

10.45.2.24 `static void gazebo::event::Events::DisconnectPreRender (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a render start signal.

Parameters

<code>in</code>	<code>_subscriber</code>	the subscriber to this event
-----------------	--------------------------	------------------------------

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

10.45.2.25 `static void gazebo::event::Events::DisconnectRender (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the render update signal.

Parameters

<code>in</code>	<code>_subscriber</code>	the subscriber to this event
-----------------	--------------------------	------------------------------

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

10.45.2.26 `static void gazebo::event::Events::DisconnectSetSelectedEntity (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the set selected entity.

Parameters

<code>in</code>	<code>_subscriber</code>	the subscriber to this event
-----------------	--------------------------	------------------------------

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

10.45.2.27 `static void gazebo::event::Events::DisconnectStep (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the the step signal.

Parameters

<code>in</code>	<code>_subscriber</code>	the subscriber to this event
-----------------	--------------------------	------------------------------

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

10.45.2.28 `static void gazebo::event::Events::DisconnectStop (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the the stop signal.

Parameters

<code>in</code>	<code>_subscriber</code>	the subscriber to this event
-----------------	--------------------------	------------------------------

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

10.45.2.29 `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.

10.45.2.30 `static void gazebo::event::Events::DisconnectWorldUpdateBegin (ConnectionPtr _subscriber) [static]`

Disconnect a boost::slot the the world update start signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

10.45.2.31 `static void gazebo::event::Events::DisconnectWorldUpdateEnd (ConnectionPtr _subscriber) [inline], [static]`

Disconnect a boost::slot the the world update end signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

10.45.2.32 `static void gazebo::event::Events::DisconnectWorldUpdateStart (ConnectionPtr _subscriber) [static]`

Disconnect a boost::slot the the world update start signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

10.45.3 Member Data Documentation

10.45.3.1 `EventT<void (std::string)> gazebo::event::Events::addEntity [static]`

An entity has been added.

Referenced by ConnectAddEntity(), and DisconnectAddEntity().

10.45.3.2 `EventT<void (std::string)> gazebo::event::Events::deleteEntity [static]`

An entity has been deleted.

Referenced by ConnectDeleteEntity(), and DisconnectDeleteEntity().

10.45.3.3 `EventT<void (std::string)> gazebo::event::Events::diagTimerStart [static]`

Diagnostic timer start.

Referenced by `ConnectDiagTimerStart()`, and `DisconnectDiagTimerStart()`.

10.45.3.4 `EventT<void (std::string)> gazebo::event::Events::diagTimerStop` [static]

Diagnostic timer stop.

Referenced by `ConnectDiagTimerStop()`, and `DisconnectDiagTimerStop()`.

10.45.3.5 `EventT<void (std::string)> gazebo::event::Events::entityCreated` [static]

An entity has been created.

Referenced by `ConnectCreateEntity()`, and `DisconnectCreateEntity()`.

10.45.3.6 `EventT<void (bool)> gazebo::event::Events::pause` [static]

Pause signal.

Referenced by `ConnectPause()`, and `DisconnectPause()`.

10.45.3.7 `EventT<void ()> gazebo::event::Events::postRender` [static]

Post-Render.

Referenced by `ConnectPostRender()`, and `DisconnectPostRender()`.

10.45.3.8 `EventT<void ()> gazebo::event::Events::preRender` [static]

Pre-render.

Referenced by `ConnectPreRender()`, and `DisconnectPreRender()`.

10.45.3.9 `EventT<void ()> gazebo::event::Events::render` [static]

Render.

Referenced by `ConnectRender()`, and `DisconnectRender()`.

10.45.3.10 `EventT<void (std::string, std::string)> gazebo::event::Events::setSelectedEntity` [static]

An entity has been selected.

Referenced by `ConnectSetSelectedEntity()`, and `DisconnectSetSelectedEntity()`.

10.45.3.11 `EventT<void ()> gazebo::event::Events::step` [static]

Step the simulation once signal.

Referenced by `ConnectStep()`, and `DisconnectStep()`.

10.45.3.12 **EventT<void ()>** gazebo::event::Events::stop [static]

Simulation stop signal.

Referenced by ConnectStop(), and DisconnectStop().

10.45.3.13 **EventT<void (std::string)>** gazebo::event::Events::worldCreated [static]

A (p. 109) world has been created.

Referenced by ConnectWorldCreated(), and DisconnectWorldCreated().

10.45.3.14 **EventT<void (const common::UpdateInfo &)>** gazebo::event::Events::worldUpdateBegin [static]

World update has started.

Referenced by ConnectWorldUpdateBegin().

10.45.3.15 **EventT<void ()>** gazebo::event::Events::worldUpdateEnd [static]

World update has ended.

Referenced by ConnectWorldUpdateEnd(), and DisconnectWorldUpdateEnd().

10.45.3.16 **EventT<void ()>** gazebo::event::Events::worldUpdateStart [static]

World update has started.

Referenced by ConnectWorldUpdateStart().

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

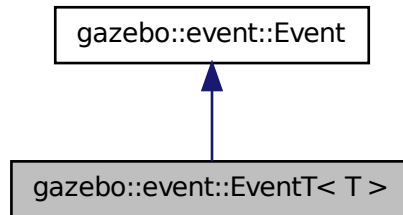
- **Events.hh**

10.46 gazebo::event::EventT< T > Class Template Reference

A (p. 109) 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.

10.46.1 Detailed Description

`template<typename T>class gazebo::event::EventT< T >`

A (p. 109) class for event processing.

10.46.2 Member Function Documentation

10.46.2.1 `template<typename T> void gazebo::event::EventT< T >::operator()() [inline]`

Access the signal.

10.46.2.2 `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
----	----	---------------

10.46.2.3 `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

10.46.2.4 `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

10.46.2.5 `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

10.46.2.6 `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

10.46.2.7 `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

10.46.2.8 `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

10.46.2.9 `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

10.46.2.10 `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

10.46.2.11 `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

10.46.2.12 `template<typename T> void gazebo::event::EventT< T >::Signal () [inline]`

Signal the event for all subscribers.

Referenced by `gazebo::event::EventT< void(bool)>::operator()()`.

10.46.2.13 `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
----	-----------------	-----------

10.46.2.14 `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

10.46.2.15 `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

10.46.2.16 `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

10.46.2.17 `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

10.46.2.18 `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

10.46.2.19 `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

10.46.2.20 `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

10.46.2.21 `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

10.46.2.22 `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:

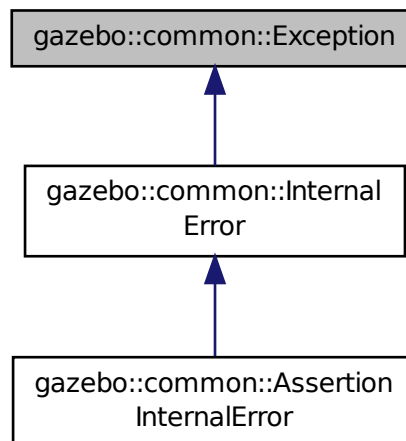
- **Event.hh**

10.47 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

10.47.1 Detailed Description

Class for generating exceptions.

10.47.2 Constructor & Destructor Documentation

10.47.2.1 gazebo::common::Exception::Exception ()

Constructor.

10.47.2.2 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

10.47.2.3 virtual gazebo::common::Exception::~~Exception () [virtual]

Destructor.

10.47.3 Member Function Documentation

10.47.3.1 `std::string gazebo::common::Exception::GetErrorFile () const`

Return the error function.

Returns

The error function name

10.47.3.2 `std::string gazebo::common::Exception::GetErrorStr () const`

Return the error string.

Returns

The error string

10.47.3.3 `void gazebo::common::Exception::Print () const`

Print the exception to std out.

10.47.4 Friends And Related Function Documentation

10.47.4.1 `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:

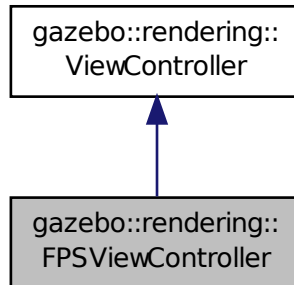
- **Exception.hh**

10.48 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. 159)	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. 867).

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. 867).

10.48.3.4 virtual void gazebo::rendering::FPSViewController::HandleMouseEvent (const common::MouseEvent & *_event*) [virtual]

Handle a mouse event.

Parameters

in	<i>_event</i>	The mouse position.
----	---------------	---------------------

Implements **gazebo::rendering::ViewController** (p. 867).

10.48.3.5 virtual void gazebo::rendering::FPSViewController::Init () [virtual]

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 867).

10.48.3.6 virtual void gazebo::rendering::FPSViewController::Update () [virtual]

Update the camera position.

Implements **gazebo::rendering::ViewController** (p. 868).

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

- **FPSViewController.hh**

10.49 urdf2gazebo::GazeboExtension Class Reference

```
#include <parser_urdf.hh>
```

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

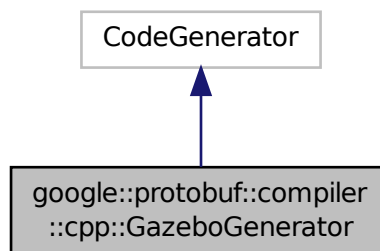
- **parser_urdf.hh**

10.50 google::protobuf::compiler::cpp::GazeboGenerator Class Reference

Google protobuf message generator for **gazebo::msgs** (p. 89).

```
#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

10.50.1 Detailed Description

Google protobuf message generator for **gazebo::msgs** (p. 89).

10.50.2 Constructor & Destructor Documentation

10.50.2.1 `google::protobuf::compiler::cpp::GazeboGenerator::GazeboGenerator (const std::string & _name)`

10.50.2.2 `virtual google::protobuf::compiler::cpp::GazeboGenerator::~~GazeboGenerator () [virtual]`

10.50.3 Member Function Documentation

10.50.3.1 `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:

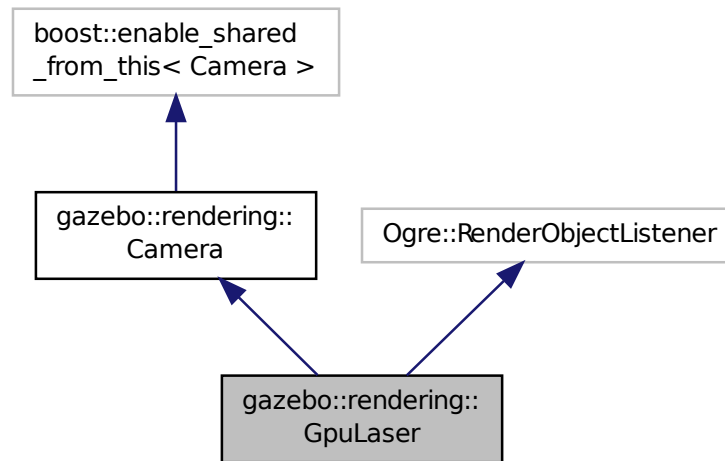
- **GazeboGenerator.hh**

10.51 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, **Scene** *_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.*
- const float * **GetLaserData** ()
 - All things needed to get back z buffer for laser data.*
- virtual void **Init** ()
 - Initialize the camera.*
- 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 **SetParentSensor** (**sensors::GpuRaySensor** *_parent)

Set the parent sensor.

- void **SetRangeCount** (unsigned int _w, unsigned int _h=1)

Set the number of laser samples in the width and height.

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, Scene * _scene, bool _autoRender = true)`

Constructor.

Parameters

in	<code>_namePrefix</code>	Unique prefix name for the camera.
in	<code>_scene</code>	Scene (p. 662) 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 (p. 109) 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. 168).

10.51.3.5 `const float* gazebo::rendering::GpuLaser::GetLaserData ()`

All things needed to get back z buffer for laser data.

Returns

Array of laser data.

10.51.3.6 `virtual void gazebo::rendering::GpuLaser::Init () [virtual]`

Initialize the camera.

Reimplemented from `gazebo::rendering::Camera` (p. 175).

10.51.3.7 `virtual void gazebo::rendering::GpuLaser::Load (sdf::ElementPtr & _sdf) [virtual]`

10.51.3.8 `virtual void gazebo::rendering::GpuLaser::Load () [virtual]`

Load the camera with default parameters.

Reimplemented from `gazebo::rendering::Camera` (p. 176).

10.51.3.9 `virtual void gazebo::rendering::GpuLaser::notifyRenderSingleObject (Ogre::Renderable * _rend, const Ogre::Pass * _p, const Ogre::AutoParamDataSource * _s, const Ogre::LightList * _ll, bool _supp) [virtual]`

10.51.3.10 `virtual void gazebo::rendering::GpuLaser::PostRender () [virtual]`

Post render.

Called after the render signal.

Reimplemented from `gazebo::rendering::Camera` (p. 177).

10.51.3.11 void gazebo::rendering::GpuLaser::SetParentSensor (sensors::GpuRaySensor * *_parent*)

Set the parent sensor.

Parameters

in	<i>_parent</i>	Pointer to a sensors::GpuRaySensor (p. 324)
----	----------------	--

10.51.3.12 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

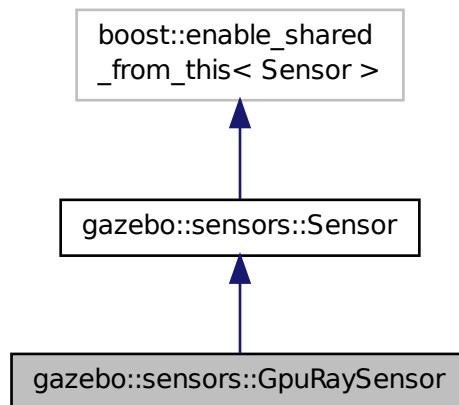
The documentation for this class was generated from the following file:

- **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. 320).
- 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) const

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.
- 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.
- 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

- unsigned int **cameraCount**
Number of cameras.
- **sdf::ElementPtr cameraElem**
Camera SDF element.
- 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.
- **sdf::ElementPtr horzElem**
Horizontal SDF element.
- double **horzHalfAngle**
Horizontal half angle.
- unsigned int **horzRangeCount**
Horizontal range count.
- unsigned int **horzRayCount**
Horizontal ray count.
- bool **isHorizontal**
True if the sensor is horizontal only.
- double **near**
Near clip plane.
- double **rangeCountRatio**
Range count ratio.
- **sdf::ElementPtr rangeElem**
Range SDF element.
- double **rayCountRatio**
Ray count ratio.
- **sdf::ElementPtr scanElem**
Scan SDF elementz.
- **sdf::ElementPtr vertElem**
Vertical SDF element.
- double **vertHalfAngle**
Vertical half angle.
- unsigned int **vertRangeCount**
Vertical range count.
- unsigned int **vertRayCount**
Vertical ray count.
- double **vfov**
Vertical field-of-view.

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

in	_subscriber	Event callback.
----	-------------	-----------------

10.52.2.2 `void gazebo::sensors::GpuRaySensor::DisconnectNewLaserFrame (event::ConnectionPtr & _conn)`

Disconnect Laser Frame.

Parameters

in, out	_conn	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. 687).

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

in	<i>_index</i>	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

(horizontal_max_angle + 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. 320).

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 (p. 109) 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) const`

Get all the ranges.

Parameters

out	<i>_range</i>	A (p. 109) 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 (p. 109) 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

in	<i>_index</i>	Index of specific ray
----	---------------	-----------------------

Returns

Intensity value of ray

10.52.2.24 `double gazebo::sensors::GpuRaySensor::GetVertFOV () const`

Get the vertical field-of-view.

10.52.2.25 `double gazebo::sensors::GpuRaySensor::GetVertHalfAngle () const`

Get $(\text{vertical_max_angle} + \text{vertical_min_angle}) * 0.5$.

Returns

$(\text{vertical_max_angle} + \text{vertical_min_angle}) * 0.5$

10.52.2.26 `math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMax () const`

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.52.2.27 `math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMin () const`

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.52.2.28 `int gazebo::sensors::GpuRaySensor::GetVerticalRangeCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.52.2.29 `int gazebo::sensors::GpuRaySensor::GetVerticalRayCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.52.2.30 `virtual void gazebo::sensors::GpuRaySensor::Init () [virtual]`

Initialize the ray.

Reimplemented from **`gazebo::sensors::Sensor`** (p. 690).

10.52.2.31 `bool gazebo::sensors::GpuRaySensor::IsHorizontal () const`

Gets if sensor is horizontal.

Returns

True if horizontal, false if not

10.52.2.32 `virtual void gazebo::sensors::GpuRaySensor::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. 683) parameters
<code>in</code>	<code>_worldName</code>	Name of world to load from

10.52.2.33 `virtual void gazebo::sensors::GpuRaySensor::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. 690).

10.52.2.34 `void gazebo::sensors::GpuRaySensor::SetAngleMax (double _angle)`

Set the scan maximum angle.

Parameters

<code>in</code>	<code>_angle</code>	The maximum angle
-----------------	---------------------	-------------------

10.52.2.35 `void gazebo::sensors::GpuRaySensor::SetAngleMin (double _angle)`

Set the scan minimum angle.

Parameters

<code>in</code>	<code><i>_angle</i></code>	The minimum angle
-----------------	----------------------------	-------------------

10.52.2.36 `void gazebo::sensors::GpuRaySensor::SetVerticalAngleMax (double _angle)`

Set the vertical scan line top angle.

Parameters

<code>in</code>	<code><i>_angle</i></code>	The Maximum angle of the scan block
-----------------	----------------------------	-------------------------------------

10.52.2.37 `void gazebo::sensors::GpuRaySensor::SetVerticalAngleMin (double _angle)`

Set the vertical scan bottom angle.

Parameters

<code>in</code>	<code><i>_angle</i></code>	The minimum angle of the scan block
-----------------	----------------------------	-------------------------------------

10.52.2.38 `virtual void gazebo::sensors::GpuRaySensor::UpdateImpl (bool _force)` `[protected]`, `[virtual]`

Update the sensor information.

Parameters

<code>in</code>	<code><i>_force</i></code>	True if update is forced, false if not
-----------------	----------------------------	--

Reimplemented from `gazebo::sensors::Sensor` (p. 691).

10.52.3 Member Data Documentation

10.52.3.1 `unsigned int gazebo::sensors::GpuRaySensor::cameraCount` `[protected]`

Number of cameras.

10.52.3.2 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::cameraElem` `[protected]`

Camera SDF element.

10.52.3.3 `double gazebo::sensors::GpuRaySensor::chfov` `[protected]`

Cos horizontal field-of-view.

10.52.3.4 `double gazebo::sensors::GpuRaySensor::cvfov` [protected]

Cos vertical field-of-view.

10.52.3.5 `double gazebo::sensors::GpuRaySensor::far` [protected]

Far clip plane.

10.52.3.6 `double gazebo::sensors::GpuRaySensor::hfov` [protected]

Horizontal field-of-view.

10.52.3.7 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::horzElem` [protected]

Horizontal SDF element.

10.52.3.8 `double gazebo::sensors::GpuRaySensor::horzHalfAngle` [protected]

Horizontal half angle.

10.52.3.9 `unsigned int gazebo::sensors::GpuRaySensor::horzRangeCount` [protected]

Horizontal range count.

10.52.3.10 `unsigned int gazebo::sensors::GpuRaySensor::horzRayCount` [protected]

Horizontal ray count.

10.52.3.11 `bool gazebo::sensors::GpuRaySensor::isHorizontal` [protected]

True if the sensor is horizontal only.

10.52.3.12 `double gazebo::sensors::GpuRaySensor::near` [protected]

Near clip plane.

10.52.3.13 `double gazebo::sensors::GpuRaySensor::rangeCountRatio` [protected]

Range count ratio.

10.52.3.14 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::rangeElem` [protected]

Range SDF element.

10.52.3.15 `double gazebo::sensors::GpuRaySensor::rayCountRatio` [protected]

Ray count ratio.

10.52.3.16 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::scanElem` [protected]

Scan SDF elementz.

10.52.3.17 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::vertElem` [protected]

Vertical SDF element.

10.52.3.18 `double gazebo::sensors::GpuRaySensor::vertHalfAngle` [protected]

Vertical half angle.

10.52.3.19 `unsigned int gazebo::sensors::GpuRaySensor::vertRangeCount` [protected]

Vertical range count.

10.52.3.20 `unsigned int gazebo::sensors::GpuRaySensor::vertRayCount` [protected]

Vertical ray count.

10.52.3.21 `double gazebo::sensors::GpuRaySensor::vfov` [protected]

Vertical field-of-view.

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. 106) 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** (p. 109) 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	<code>_scene</code>	The scene this object is part of
in	<code>_cellCount</code>	The number of cells to draw
in	<code>_cellLength</code>	The size of each cell
in	<code>_lineWidth</code>	The width of the lines to use
in	<code>_color</code>	The color of the grid

10.53.2.2 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

<i>in</i>	<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. 106) scene node associated with this grid.

Returns

The **Ogre** (p. 106) 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

in	_count	The number of cells
----	--------	---------------------

10.53.3.10 `void gazebo::rendering::Grid::SetCellLength (float _len)`

Set the cell length.

Parameters

in	_len	The cell length
----	------	-----------------

10.53.3.11 `void gazebo::rendering::Grid::SetColor (const common::Color & _color)`

Sets the color of the grid.

Parameters

in	_color	The grid color
----	--------	----------------

10.53.3.12 `void gazebo::rendering::Grid::SetHeight (uint32_t _count)`

Set the height of the grid.

Parameters

in	_count	Grid (p. 336) height
----	--------	-----------------------------

10.53.3.13 `void gazebo::rendering::Grid::SetLineWidth (float _width)`

Set the line width.

Parameters

<code>in</code>	<code>_width</code>	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

<code>in</code>	<code>_data</code>	The user data
-----------------	--------------------	---------------

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

- **Grid.hh**

10.54 gazebo::physics::Gripper Class Reference

A (p. 109) gripper abstraction.

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

Public Member Functions

- **Gripper (ModelPtr _model)**
Constructor.
- virtual `~Gripper ()`
Destructor.
- virtual void **Init ()**
Initialize.
- virtual void **Load (sdf::ElementPtr _sdf)**
Load the gripper.

10.54.1 Detailed Description

A (p. 109) gripper abstraction.

A (p. 109) 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

<code>in</code>	<code>_model</code>	The model which contains the Gripper (p. 340).
-----------------	---------------------	---

10.54.2.2 virtual gazebo::physics::Gripper::~~Gripper () [virtual]

Destructor.

10.54.3 Member Function Documentation

10.54.3.1 virtual void gazebo::physics::Gripper::Init () [virtual]

Initialize.

10.54.3.2 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.
----	------	--

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

- **Gripper.hh**

10.55 gazebo::rendering::GUIOverlay Class Reference

A (p. 109) 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. 159) to and overlay window.*
- bool **AttachCameraToImage** (**DepthCameraPtr** &_camera, const std::string &_windowName)
*Use this function to draw the output from a **rendering::DepthCamera** (p. 249) 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 (p. 109) 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. 159) 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. 249) 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

in	<code>_key</code>	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

<code>in</code>	<code>_key</code>	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

<code>in</code>	<code>_evt</code>	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

<code>in</code>	<code>_renderTarget</code>	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

<i>in</i>	<i>_filename</i>	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.

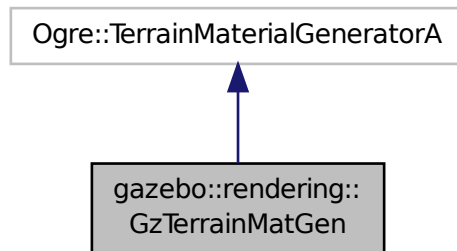
The documentation for this class was generated from the following file:

- **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.
- double **GetHeight** (double _x, double _y, double _z=1000)
Get the height at a location.
- 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.

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 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

10.57.3.2 Ogre::TerrainGroup* gazebo::rendering::Heightmap::GetOgreTerrain () const

Get a pointer to the OGRE terrain group object.

Returns

Pointer to the OGRE terrain.

10.57.3.3 void gazebo::rendering::Heightmap::Load ()

Load the heightmap.

10.57.3.4 void gazebo::rendering::Heightmap::LoadFromMsg (ConstVisualPtr & *_msg*)

Load the heightmap from a visual message.

Parameters

in	<i>_msg</i>	The visual message containing heightmap info
----	-------------	--

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

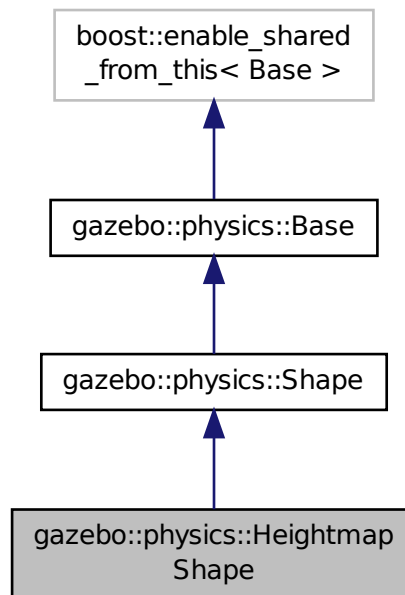
- **Heightmap.hh**

10.58 gazebo::physics::HeightmapShape Class Reference

HeightmapShape (p. 348) 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)
Get a height at a position.
- 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.

Protected Attributes

- std::vector< float > **heights**
Lookup table of heights.
- **common::Image img**
Image used to generate the heights.
- **math::Vector3 scale**
Scaling factor.
- int **subSampling**
Level of subsampling.
- unsigned int **vertSize**
Size of the height lookup table.

Additional Inherited Members

10.58.1 Detailed Description

HeightmapShape (p. 348) 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

in	<code>_parent</code>	Parent Collision (p. 192) 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

<code>in</code>	<code>_msg</code>	Message to fill.
-----------------	-------------------	------------------

Implements `gazebo::physics::Shape` (p. 708).

10.58.3.2 `float gazebo::physics::HeightmapShape::GetHeight (int _x, int _y)`

Get a height at a position.

Parameters

<code>in</code>	<code>_x</code>	X position.
<code>in</code>	<code>_y</code>	Y position.

Returns

The height at a the specified location.

10.58.3.3 `float gazebo::physics::HeightmapShape::GetMaxHeight () const`

Get the maximum height.

Returns

The maximum height.

10.58.3.4 `float gazebo::physics::HeightmapShape::GetMinHeight () const`

Get the minimum height.

Returns

The minimum height.

10.58.3.5 `math::Vector3 gazebo::physics::HeightmapShape::GetPos () const`

Get the origin in world coordinate frame.

Returns

The origin in world coordinate frame.

10.58.3.6 `math::Vector3 gazebo::physics::HeightmapShape::GetSize () const`

Get the size in meters.

Returns

The size in meters.

10.58.3.7 `int gazebo::physics::HeightmapShape::GetSubSampling () const`

Get the amount of subsampling.

Returns

Amount of subsampling.

10.58.3.8 `std::string gazebo::physics::HeightmapShape::GetURI () const`

Get the URI of the heightmap image.

Returns

The heightmap image URI.

10.58.3.9 `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. 830), result.x = width, result.y = length/height.

10.58.3.10 `virtual void gazebo::physics::HeightmapShape::Init () [virtual]`

Initialize the heightmap.

Implements **gazebo::physics::Shape** (p. 708).

10.58.3.11 `virtual void gazebo::physics::HeightmapShape::Load (sdf::ElementPtr _sdf) [virtual]`

Load the heightmap.

Parameters

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

Reimplemented from `gazebo::physics::Base` (p. 143).

10.58.3.12 `virtual void gazebo::physics::HeightmapShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]`

Update the heightmap from a message.

Parameters

<code>in</code>	<code>_msg</code>	Message to update from.
-----------------	-------------------	-------------------------

Implements `gazebo::physics::Shape` (p. 708).

10.58.4 Member Data Documentation

10.58.4.1 `std::vector<float> gazebo::physics::HeightmapShape::heights [protected]`

Lookup table of heights.

10.58.4.2 `common::Image gazebo::physics::HeightmapShape::img [protected]`

Image used to generate the heights.

10.58.4.3 `math::Vector3 gazebo::physics::HeightmapShape::scale [protected]`

Scaling factor.

10.58.4.4 `int gazebo::physics::HeightmapShape::subSampling [protected]`

Level of subsampling.

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:

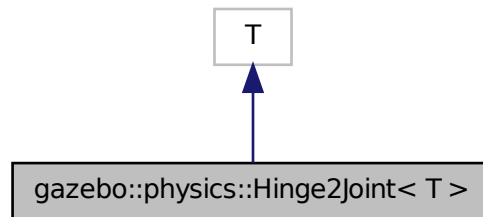
- `HeightmapShape.hh`

10.59 `gazebo::physics::Hinge2Joint< T >` Class Template Reference

A (p. 109) 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 (p. 109) 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	<code>_parent</code>	Parent link.
----	----------------------	--------------

References gazebo::physics::Base::HINGE2_JOINT.

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	_sdf	SDF values to load from.
----	------	--------------------------

References `sdf::Element::GetElement()`, and `sdf::Element::GetValueVector3()`.

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

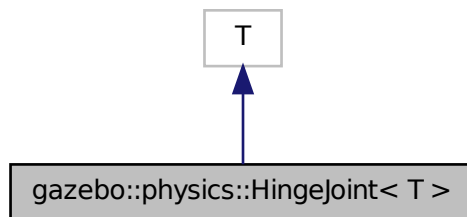
- [Hinge2Joint.hh](#)

10.60 gazebo::physics::HingeJoint< T > Class Template Reference

A (p. 109) 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 (p. 109) 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

<code>in</code>	<code>_parent</code>	Parent link
-----------------	----------------------	-------------

References `gazebo::physics::Base::HINGE_JOINT`.

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.

References `gazebo::msgs::Init()`.

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
----	------	------------------------

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. 357).*

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

<code>in</code>	<code>_filename</code>	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. 357).

Parameters

<code>in</code>	<code>_format</code>	Pixel format string.
-----------------	----------------------	----------------------

See Also

`Image::PixelFormatNames`

Returns

Image::PixelFormat (p. 357)

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	<code>_data</code>	Pointer to a NULL array of char.
out	<code>_count</code>	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	<i>_x</i>	Column location in the image
in	<i>_y</i>	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	<i>_data</i>	Pointer to a NULL array of char.
out	<i>_count</i>	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

in	<i>_filename</i>	the path to the image file
----	------------------	----------------------------

10.61.4.14 void gazebo::common::Image::Rescale (int *_width*, int *_height*)

Rescale the image.

Parameters

in	<i>_width</i>	New image width
in	<i>_height</i>	New image height

10.61.4.15 void gazebo::common::Image::SavePNG (const std::string & *_filename*)

Save the image in PNG format.

Parameters

in	<i>_filename</i>	The name of the saved image
----	------------------	-----------------------------

10.61.4.16 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

in	<i>_data</i>	Pointer to the raw image data
in	<i>_width</i>	Width in pixels
in	<i>_height</i>	Height in pixels
in	<i>_format</i>	Pixel format of the provided data

10.61.4.17 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:

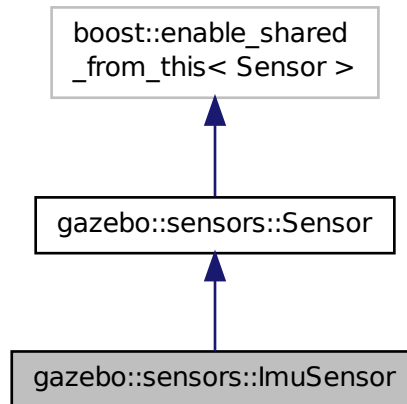
- **Image.hh**

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 \sim **ImuSensor** ()
Destructor.
- **math::Vector3 GetAngularVelocity** () const
Returns the angular velocity.
- **math::Vector3 GetLinearAcceleration** () const
Returns the imu linear acceleration.
- **math::Quaternion GetOrientation** () const
get orientation of the IMU relative to the reference pose
- void **SetReferencePose** ()
Sets the current pose as the IMU reference pose.

Protected Member Functions

- virtual void **Fini** ()
Finalize the sensor.
- virtual void **Init** ()
Initialize the IMU.
- 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 ()

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. 687).

10.62.3.2 math::Vector3 gazebo::sensors::ImuSensor::GetAngularVelocity () const

Returns the angular velocity.

Returns

Angular velocity.

10.62.3.3 math::Vector3 gazebo::sensors::ImuSensor::GetLinearAcceleration () const

Returns the imu linear acceleration.

Returns

Linear acceleration.

10.62.3.4 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.

10.62.3.5 `virtual void gazebo::sensors::ImuSensor::Init ()` [protected],[virtual]

Initialize the IMU.

Reimplemented from `gazebo::sensors::Sensor` (p. 690).

10.62.3.6 `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 <code>Sensor</code> (p. 683) parameters.
in	<code>_worldName</code>	Name of world to load from.

Reimplemented from `gazebo::sensors::Sensor` (p. 690).

10.62.3.7 `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. 690).

10.62.3.8 `void gazebo::sensors::ImuSensor::SetReferencePose ()`

Sets the current pose as the IMU reference pose.

10.62.3.9 `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. 691).

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

- `ImuSensor.hh`

10.63 gazebo::physics::Inertial Class Reference

A (p. 109) 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.
- 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.
- const **math::Pose** **GetPose** () const
*Get the pose about which the mass and inertia matrix is specified in the **Link** (p. 412) 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 **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 **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 (p. 109) 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

in	<i>_mass</i>	Mass value in kg if using metric.
----	--------------	-----------------------------------

10.63.2.3 gazebo::physics::Inertial::Inertial (const Inertial & *_inertial*)

Copy constructor.

Parameters

in	<i>_inertial</i>	Inertial (p. 365) 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.

10.63.3.2 double gazebo::physics::Inertial::GetIXX () const

Get IXX.

Returns

IXX value

10.63.3.3 double gazebo::physics::Inertial::GetIYX () const

Get IYX.

Returns

IYX value

10.63.3.4 `double gazebo::physics::Inertial::GetIXZ () const`

Get IXZ.

Returns

IXZ value

10.63.3.5 `double gazebo::physics::Inertial::GetIYY () const`

Get IYY.

Returns

IYY value

10.63.3.6 `double gazebo::physics::Inertial::GetIYZ () const`

Get IYZ.

Returns

IYZ value

10.63.3.7 `double gazebo::physics::Inertial::GetIZZ () const`

Get IZZ.

Returns

IZZ value

10.63.3.8 `double gazebo::physics::Inertial::GetMass () const`

Get the mass.

10.63.3.9 `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. 412) frame.

Returns

The inertial pose.

10.63.3.10 `math::Vector3 gazebo::physics::Inertial::GetPrincipalMoments () const`

Get the principal moments of inertia (Ixx, Iyy, Izz).

Returns

The principal moments.

10.63.3.11 `math::Vector3 gazebo::physics::Inertial::GetProductsofInertia () const`

Get the products of inertia (Ixy, Ixy, Iyz).

Returns

The products of inertia.

10.63.3.12 `void gazebo::physics::Inertial::Load (sdf::ElementPtr _sdf)`

Load from SDF values.

Parameters

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

10.63.3.13 `Inertial gazebo::physics::Inertial::operator+ (const Inertial & _inertial) const`

Addition operator.

Parameters

<code>in</code>	<code>_inertial</code>	Inertial (p. 365) to add.
-----------------	------------------------	----------------------------------

Returns

The result of the addition.

10.63.3.14 `const Inertial& gazebo::physics::Inertial::operator+= (const Inertial & _inertial)`

Addition equal operator.

Parameters

<code>in</code>	<code>_inertial</code>	Inertial (p. 365) to add.
-----------------	------------------------	----------------------------------

Returns

Reference to this object.

10.63.3.15 `Inertial& gazebo::physics::Inertial::operator= (const Inertial & _inertial)`

Equal operator.

Parameters

<code>in</code>	<code>_inertial</code>	Inertial (p. 365) to copy.
-----------------	------------------------	-----------------------------------

Returns

Reference to this object.

10.63.3.16 `void gazebo::physics::Inertial::ProcessMsg (const msgs::Inertial & _msg)`

Update parameters from a message.

Parameters

<code>in</code>	<code>_msg</code>	Message to read
-----------------	-------------------	-----------------

10.63.3.17 `void gazebo::physics::Inertial::Reset ()`

Reset all the mass properties.

10.63.3.18 `void gazebo::physics::Inertial::Rotate (const math::Quaternion & _rot)`

Rotate this mass.

Parameters

<code>in</code>	<code>_rot</code>	Rotation amount.
-----------------	-------------------	------------------

10.63.3.19 `void gazebo::physics::Inertial::SetCoG (double _cx, double _cy, double _cz)`

Set the center of gravity.

Parameters

<code>in</code>	<code>_cx</code>	X position.
<code>in</code>	<code>_cy</code>	Y position.
<code>in</code>	<code>_cz</code>	Z position.

10.63.3.20 `void gazebo::physics::Inertial::SetCoG (const math::Vector3 & _center)`

Set the center of gravity.

Parameters

<code>in</code>	<code>_center</code>	Center of the gravity.
-----------------	----------------------	------------------------

10.63.3.21 `void gazebo::physics::Inertial::SetInertiaMatrix (double _ixx, double _iyy, double _izz, double _ixy, double _ixz, double iyz)`

Set the mass matrix.

Parameters

in	<code>_ixx</code>	X second moment of inertia about x axis.
in	<code>_jyy</code>	Y second moment of inertia about y axis.
in	<code>_izz</code>	Z second moment of inertia about z axis.
in	<code>_ixy</code>	XY inertia.
in	<code>_ixz</code>	XZ inertia.
in	<code>_iyz</code>	YZ inertia.

10.63.3.22 void gazebo::physics::Inertial::SetIXX (double `_v`)

Set IXX.

Parameters

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

10.63.3.23 void gazebo::physics::Inertial::SetIXY (double `_v`)

Set IXY.

Parameters

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

10.63.3.24 void gazebo::physics::Inertial::SetIXZ (double `_v`)

Set IXZ.

Parameters

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

10.63.3.25 void gazebo::physics::Inertial::SetIYY (double `_v`)

Set IYY.

Parameters

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

10.63.3.26 void gazebo::physics::Inertial::SetIYZ (double `_v`)

Set IYZ.

Parameters

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

10.63.3.27 void gazebo::physics::Inertial::SetIZZ (double *_v*)

Set IZZ.

Parameters

<i>in</i>	<i>_v</i>	IZZ value
-----------	-----------	-----------

10.63.3.28 void gazebo::physics::Inertial::SetMass (double *m*)

Set the mass.

10.63.3.29 void gazebo::physics::Inertial::UpdateParameters (sdf::ElementPtr *_sdf*)

update the parameters using new sdf values.

Parameters

<i>in</i>	<i>_sdf</i>	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

<i>in</i>	<i>_out</i>	Output stream.
<i>in</i>	<i>_inertial</i>	Inertial (p. 365) object to output.

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

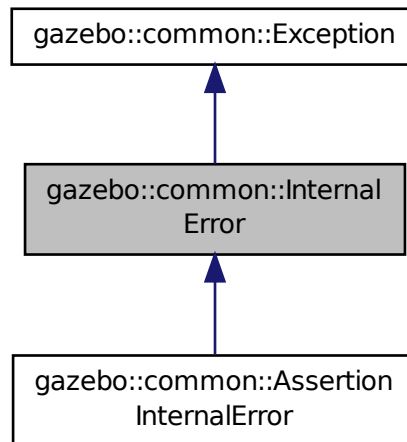
- **Inertial.hh**

10.64 gazebo::common::InternalError Class Reference

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

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

Inheritance diagram for gazebo::common::InternalError:



Public Member Functions

- **InternalError** ()
Constructor.
- **InternalError** (const char *_file, int _line, const std::string _msg)
Default constructor.
- virtual \sim **InternalError** ()
Destructor.

10.64.1 Detailed Description

Class for generating Internal Gazebo Errors: those errors which should never happen 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:

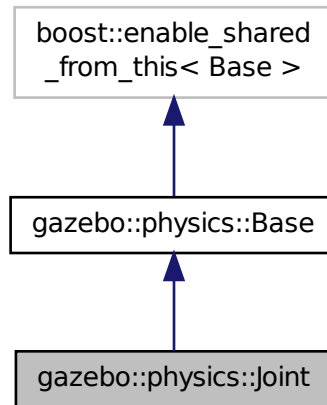
- **IOManager.hh**

10.66 gazebo::physics::Joint Class Reference

Base (p. 135) 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. 375) 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.
- template<typename T >
event::ConnectionPtr ConnectJointUpdate (T _subscriber)
Connect a boost::slot to the joint update signal.
- virtual void **Detach** ()
Detach this joint from all links.
- void **DisconnectJointUpdate** (**event::ConnectionPtr** &_conn)
Disconnect a boost::slot to 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.
- virtual double **GetEffortLimit** (int _index)
Get the effort limit on axis(index).
- virtual double **GetForce** (int _index) **GAZEBO_DEPRECATED**
- virtual double **GetForce** (unsigned int _index)
- virtual **JointWrench GetForceTorque** (int _index) **GAZEBO_DEPRECATED=0**
get internal force and torque values at a joint Note that you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.
- virtual **JointWrench GetForceTorque** (unsigned int _index)=0
get internal force and torque values at a joint Note that you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const =0
Get the axis of rotation in global coordinate 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. 412) due to the existence of this **Joint** (p. 375).*
- virtual **math::Vector3 GetLinkTorque** (unsigned int _index) const =0
*Get the torque applied to the center of mass of a **physics::Link** (p. 412) due to the existence of this **Joint** (p. 375).*
- **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 (replacée 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. 375).*
 - void **Load** (LinkPtr _parent, LinkPtr _child, const math::Vector3 &_pos) **GAZEBO_DEPRECATED**
 - Set parent and child links of a **physics::Joint** (p. 375) and its anchor offset position.*
 - virtual void **Load** (sdf::ElementPtr _sdf)
 - Load **physics::Joint** (p. 375) from a SDF **sdf::Element** (p. 269).*
 - 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. 375) is static, Gazebo stores the state of this **Joint** (p. 375) as a scalar inside the **Joint** (p. 375) class, so this call will NOT move the joint dynamically for a static **Model** (p. 480).*
 - 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.*
 - virtual void **SetDamping** (int _index, double _damping)=0
 - Set the joint damping.*
 - virtual void **SetForce** (int _index, double _force)
 - Set the force applied to this **physics::Joint** (p. 375).*
 - 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.*
 - 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. 375) effort limit as specified in SDF.*
- double **forceApplied** [2]
Save force applied by user This plus the joint feedback (joint constraint forces) is the equivalent of simulated force torque sensor reading Allocate a 2 vector in case hinge2 joint is used.
- double **inertiaRatio** [2]
*Store **Joint** (p. 375) inertia ratio.*
- **math::Angle lowerLimit** [2]
*Store **Joint** (p. 375) position lower limit as specified in SDF.*
- **ModelPtr model**
Pointer to the parent model.
- **LinkPtr parentLink**
The second link this joint connects to.
- **math::Angle upperLimit** [2]
*Store **Joint** (p. 375) position upper limit as specified in SDF.*
- bool **useCFMDamping**
option to use CFM damping
- double **velocityLimit** [2]
*Store **Joint** (p. 375) velocity limit as specified in SDF.*

10.66.1 Detailed Description

Base (p. 135) class for all joints.

10.66.2 Member Enumeration Documentation

10.66.2.1 enum gazebo::physics::Joint::Attribute

Joint (p. 375) 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

in	Joint (p. 375)	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 if the two bodies are connected by a joint.

Parameters

in	_one	First link.
in	_two	Second link.

Returns

True if the two links are connected by a joint.

10.66.4.3 `virtual void gazebo::physics::Joint::Attach (LinkPtr _parent, LinkPtr _child) [virtual]`

Attach the two bodies with this joint.

Parameters

in	<code>_parent</code>	Parent link.
in	<code>_child</code>	Child link.

10.66.4.4 `template<typename T > event::ConnectionPtr gazebo::physics::Joint::ConnectJointUpdate (T _subscriber)`
`[inline]`

Connect a boost::slot the the joint update signal.

Parameters

in	<code>_subscriber</code>	Callback for the connection.
----	--------------------------	------------------------------

Returns

Connection pointer, which must be kept in scope.

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

10.66.4.5 `virtual void gazebo::physics::Joint::Detach ()` `[virtual]`

Detach this joint from all links.

10.66.4.6 `void gazebo::physics::Joint::DisconnectJointUpdate (event::ConnectionPtr & _conn)` `[inline]`

Disconnect a boost::slot the the joint update signal.

Parameters

in	<code>_conn</code>	Connection to disconnect.
----	--------------------	---------------------------

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

10.66.4.7 `void gazebo::physics::Joint::FillMsg (msgs::Joint & _msg)`

Fill a joint message.

Parameters

out	<code>_msg</code>	Message to fill with this joint's properties.
-----	-------------------	---

10.66.4.8 `virtual math::Vector3 gazebo::physics::Joint::GetAnchor (int _index) const` `[pure virtual]`

Get the anchor point.

Parameters

in	<code>_index</code>	Index of the axis.
----	---------------------	--------------------

Returns

Anchor value for the axis.

10.66.4.9 math::Angle gazebo::physics::Joint::GetAngle (int *_index*) const

Get the angle of rotation of an axis(index)

Parameters

<i>in</i>	<i>_index</i>	Index of the axis.
-----------	---------------	--------------------

Returns

Angle of the axis.

10.66.4.10 virtual unsigned int gazebo::physics::Joint::GetAngleCount () const [pure virtual]

Get the angle count.

Returns

The number of DOF for the joint.

Referenced by GetLowerLimit(), and GetUpperLimit().

10.66.4.11 virtual math::Angle gazebo::physics::Joint::GetAngleImpl (int *_index*) const [protected], [pure 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.

10.66.4.12 virtual double gazebo::physics::Joint::GetAttribute (const std::string & *_key*, unsigned int *_index*) [pure virtual]

Get a non-generic parameter for the joint.

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.

10.66.4.13 `LinkPtr gazebo::physics::Joint::GetChild () const`

Get the child link.

Returns

Pointer to the child link.

10.66.4.14 `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

10.66.4.15 `virtual double gazebo::physics::Joint::GetForce (int _index) [virtual]`

Todo : not yet implemented. Get the forces applied at this **Joint** (p. 375). Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

The force applied to an axis.

10.66.4.16 `virtual double gazebo::physics::Joint::GetForce (unsigned int _index) [virtual]`

Todo : not yet implemented. Get the forces applied at this **Joint** (p. 375). Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

The force applied to an axis.

10.66.4.17 virtual **JointWrench** gazebo::physics::Joint::GetForceTorque (int *_index*) [pure virtual]

get internal force and torque values at a joint Note that you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.

Parameters

in	<i>_index</i>	Force and torque on child link if <i>_index</i> = 0 and on parent link of <i>_index</i> = 1
----	---------------	---

Returns

The force and torque at the joint

10.66.4.18 virtual **JointWrench** gazebo::physics::Joint::GetForceTorque (unsigned int *_index*) [pure virtual]

get internal force and torque values at a joint Note that you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.

Parameters

in	<i>_index</i>	Force and torque on child link if <i>_index</i> = 0 and on parent link of <i>_index</i> = 1
----	---------------	---

Returns

The force and torque at the joint

10.66.4.19 virtual **math::Vector3** gazebo::physics::Joint::GetGlobalAxis (int *_index*) const [pure 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.

10.66.4.20 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

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

Angle of the high stop value.

10.66.4.21 `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. 375) axis index.
-----------------	----------------------------	-----------------------------------

Returns

returns the inertia ratio across specified joint axis.

10.66.4.22 `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.

10.66.4.23 `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. 412) due to the existence of this **Joint** (p. 375).

Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

<code>in</code>	<code><i>index</i></code>	The index of the link(0 or 1).
-----------------	---------------------------	--------------------------------

Returns

Force applied to the link.

10.66.4.24 `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. 412) due to the existence of this **Joint** (p. 375).

Note that the unit of torque should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons-Meters. If using imperial units (sorry), then unit of force is lb-force-inches not (lb-mass-inches), etc.

Parameters

<code>in</code>	<code>index</code>	The index of the link(0 or 1)
-----------------	--------------------	-------------------------------

Returns

Torque applied to the link.

10.66.4.25 `math::Vector3 gazebo::physics::Joint::GetLocalAxis (int index) const`

Get the axis of rotation.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to get.
-----------------	---------------------	---------------------------

Returns

Axis value for the provided index.

10.66.4.26 `math::Angle gazebo::physics::Joint::GetLowerLimit (unsigned int index) const` `[inline]`

: get the joint upper limit (replaces GetLowStop and GetHighStop)

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Upper limit of the axis.

References GetAngleCount(), gzwarn, and lowerLimit.

10.66.4.27 `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>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Angle of the low stop value.

10.66.4.28 `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. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

The maximum force.

10.66.4.29 `LinkPtr gazebo::physics::Joint::GetParent () const`

Get the parent link.

Returns

Pointer to the parent link.

10.66.4.30 `math::Angle gazebo::physics::Joint::GetUpperLimit (unsigned int _index) const [inline]`

: get the joint lower limit (replaces GetLowStop and GetHighStop)

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Upper limit of the axis.

References GetAngleCount(), gzwarn, and upperLimit.

10.66.4.31 `virtual double gazebo::physics::Joint::GetVelocity (int _index) const [pure virtual]`

Get the rotation rate of an axis(index)

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

The rotational velocity of the joint axis.

10.66.4.32 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

10.66.4.33 virtual void gazebo::physics::Joint::Init () [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Base** (p. 143).

10.66.4.34 void gazebo::physics::Joint::Load (LinkPtr *_parent*, LinkPtr *_child*, const math::Pose & *_pose*)

Set pose, parent and child links of a **physics::Joint** (p. 375).

Parameters

in	<i>_parent</i>	Parent link.
in	<i>_child</i>	Child link.
in	<i>_pose</i>	Pose containing Joint (p. 375) Anchor offset from child link.

10.66.4.35 void gazebo::physics::Joint::Load (LinkPtr *_parent*, LinkPtr *_child*, const math::Vector3 & *_pos*)

Set parent and child links of a **physics::Joint** (p. 375) and its anchor offset position.

This function is deprecated, use **Load(LinkPtr *_parent*, LinkPtr *_child*, const math::Pose & *_pose*)** (p. 388)

Parameters

in	<i>_parent</i>	Parent link.
in	<i>_child</i>	Child link.
in	<i>_pos</i>	Joint (p. 375) Anchor offset from child link.

10.66.4.36 virtual void gazebo::physics::Joint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load **physics::Joint** (p. 375) from a SDF **sdf::Element** (p. 269).

Parameters

in	<i>_sdf</i>	SDF values to load from.
----	-------------	--------------------------

Reimplemented from **gazebo::physics::Base** (p. 143).

10.66.4.37 virtual void gazebo::physics::Joint::Reset () [virtual]

Reset the joint.

Reimplemented from **gazebo::physics::Base** (p. 144).

10.66.4.38 virtual void gazebo::physics::Joint::SetAnchor (int *_index*, const math::Vector3 & *_anchor*) [pure virtual]

Set the anchor point.

Parameters

in	<i>_index</i>	Indx of the axis.
in	<i>_anchor</i>	Anchor value.

10.66.4.39 void gazebo::physics::Joint::SetAngle (int *_index*, math::Angle *_angle*)

If the **Joint** (p. 375) is static, Gazebo stores the state of this **Joint** (p. 375) as a scalar inside the **Joint** (p. 375) class, so this call will NOT move the joint dynamically for a static **Model** (p. 480).

But if this **Model** (p. 480) is not static, then it is updated dynamically, all the conected children **Link** (p. 412)'s are moved as a result of the **Joint** (p. 375) angle setting. Dynamic **Joint** (p. 375) angle update is accomplished by calling **JointController::SetJointPosition** (p. 394).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_angle</i>	Angle to set the joint to.

10.66.4.40 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	<i>_key</i>	String key.
in	<i>_index</i>	Index of the axis.
in	<i>_value</i>	Value of the attribute.

10.66.4.41 virtual void gazebo::physics::Joint::SetAxis (int *_index*, const math::Vector3 & *_axis*) [pure virtual]

Set the axis of rotation.

Parameters

in	<i>_index</i>	Index of the axis to set.
in	<i>_axis</i>	Vector in world frame of axis direction (must have length greater than zero).

10.66.4.42 `virtual void gazebo::physics::Joint::SetDamping (int _index, double _damping) [pure virtual]`

Set the joint damping.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to set.
<code>in</code>	<code>_damping</code>	Damping value for the axis.

10.66.4.43 `virtual void gazebo::physics::Joint::SetForce (int _index, double _force) [virtual]`

Set the force applied to this **physics::Joint** (p. 375).

Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_force</code>	Force value.

10.66.4.44 `virtual void gazebo::physics::Joint::SetHighStop (int _index, const math::Angle & _angle) [virtual]`

Set the high stop of an axis(index).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_angle</code>	High stop angle.

10.66.4.45 `virtual void gazebo::physics::Joint::SetLowStop (int _index, const math::Angle & _angle) [virtual]`

Set the low stop of an axis(index).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_angle</code>	Low stop angle.

10.66.4.46 `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. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_force</code>	Maximum force that can be applied to the axis.

10.66.4.47 void gazebo::physics::Joint::SetModel (ModelPtr *_model*)

Set the model this joint belongs too.

Parameters

in	<i>_model</i>	Pointer to a model.
----	---------------	---------------------

10.66.4.48 void gazebo::physics::Joint::SetState (const JointState & *_state*)

Set the joint state.

Parameters

in	<i>_state</i>	Joint (p. 375) state
----	---------------	-----------------------------

10.66.4.49 virtual void gazebo::physics::Joint::SetVelocity (int *_index*, double *_vel*) [pure virtual]

Set the velocity of an axis(index).

Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_vel</i>	Velocity.

10.66.4.50 void gazebo::physics::Joint::Update () [virtual]

Update the joint.

Reimplemented from **gazebo::physics::Base** (p. 146).

10.66.4.51 virtual void gazebo::physics::Joint::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::Base** (p. 146).

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. 375) effort limit as specified in SDF.

10.66.5.8 `double gazebo::physics::Joint::forceApplied[2]` [protected]

Save force applied by user This plus the joint feedback (joint constraint forces) is the equivalent of simulated force torque sensor reading Allocate a 2 vector in case hinge2 joint is used.

10.66.5.9 `double gazebo::physics::Joint::inertiaRatio[2]` [protected]

Store **Joint** (p. 375) inertia ratio.

This is a measure of how well this model behaves using interative LCP solvers.

10.66.5.10 `math::Angle gazebo::physics::Joint::lowerLimit[2]` [protected]

Store **Joint** (p. 375) position lower limit as specified in SDF.

Referenced by GetLowerLimit().

10.66.5.11 `ModelPtr gazebo::physics::Joint::model` [protected]

Pointer to the parent model.

10.66.5.12 `LinkPtr gazebo::physics::Joint::parentLink` [protected]

The second link this joint connects to.

10.66.5.13 `math::Angle gazebo::physics::Joint::upperLimit[2]` [protected]

Store **Joint** (p. 375) position upper limit as specified in SDF.

Referenced by `GetUpperLimit()`.

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. 375) velocity limit as specified in SDF.

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

- **Joint.hh**

10.67 gazebo::physics::JointController Class Reference

A (p. 109) class for manipulating **physics::Joint** (p. 375).

```
#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)
*Set the positions of a **Joint** (p. 375) by name.*
- void **SetJointPosition** (**JointPtr** _joint, double _position)
*Set the positions of a **Joint** (p. 375) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 733) and radians for **HingeJoint** (p. 354), etc.*
- void **SetJointPositions** (const std::map< std::string, double > &_jointPositions)
*Set the positions of a set of **Joint** (p. 375)'s.*
- void **Update** ()
Update the joint control.

10.67.1 Detailed Description

A (p. 109) class for manipulating **physics::Joint** (p. 375).

10.67.2 Constructor & Destructor Documentation

10.67.2.1 gazebo::physics::JointController::JointController (**ModelPtr** *_model*) [explicit]

Constructor.

Parameters

in	<i>_model</i>	Model (p. 480) that uses this joint controller.
----	---------------	--

10.67.3 Member Function Documentation

10.67.3.1 void gazebo::physics::JointController::AddJoint (**JointPtr** *_joint*)

Add a joint to control.

Parameters

in	<i>_joint</i>	Joint (p. 375) to control.
----	---------------	-----------------------------------

10.67.3.2 void gazebo::physics::JointController::Reset ()

Reset all commands.

10.67.3.3 void gazebo::physics::JointController::SetJointPosition (const std::string & *_name*, double *_position*)

Set the positions of a **Joint** (p. 375) by name.

See Also

JointController::SetJointPosition(JointPtr, double) (p. 394)

10.67.3.4 void gazebo::physics::JointController::SetJointPosition (**JointPtr** *_joint*, double *_position*)

Set the positions of a **Joint** (p. 375) by name. The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 733) and radians for **HingeJoint** (p. 354), etc.

Implementation: In order to change the position of a **Joint** (p. 375) inside a **Model** (p. 480), this call must recursively crawl through all the connected children **Link** (p. 412)'s in this **Model** (p. 480), and update each **Link** (p. 412) Pose affected by this **Joint** (p. 375) 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

in	<i>_joint</i>	Joint (p. 375) to set.
in	<i>_position</i>	Position of the joint.

10.67.3.5 void gazebo::physics::JointController::SetJointPositions (const std::map< std::string, double > & *_jointPositions*)

Set the positions of a set of **Joint** (p. 375)'s.

See Also

JointController::SetJointPosition(JointPtr, double) (p. 394)

10.67.3.6 void gazebo::physics::JointController::Update ()

Update the joint control.

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

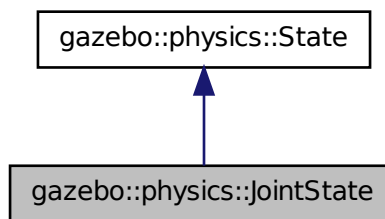
- **JointController.hh**

10.68 gazebo::physics::JointState Class Reference

keeps track of state of a **physics::Joint** (p. 375)

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

Inheritance diagram for gazebo::physics::JointState:



Public Member Functions

- **JointState** ()
Default 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.

- `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

10.68.1 Detailed Description

keeps track of state of a **physics::Joint** (p. 375)

10.68.2 Constructor & Destructor Documentation

10.68.2.1 `gazebo::physics::JointState::JointState ()`

Default constructor.

10.68.2.2 `gazebo::physics::JointState::JointState (JointPtr joint) [explicit]`

Constructor.

Parameters

<code>in</code>	<code>_joint</code>	Joint (p. 375) to get the state of.
-----------------	---------------------	--

10.68.2.3 `gazebo::physics::JointState::JointState (const sdf::ElementPtr sdf) [explicit]`

Constructor.

Build a **JointState** (p. 395) from SDF data

Parameters

in	_sdf	SDF data to load a joint state from.
----	------	--------------------------------------

10.68.2.4 virtual gazebo::physics::JointState::~~JointState () [virtual]

Destructor.

10.68.3 Member Function Documentation

10.68.3.1 void gazebo::physics::JointState::FillSDF (sdf::ElementPtr _sdf)

Populate a state SDF element with data from the object.

Parameters

out	_sdf	SDF element to populate.
-----	------	--------------------------

10.68.3.2 math::Angle gazebo::physics::JointState::GetAngle (unsigned int _axis) const

Get the joint angle.

Parameters

in	_axis	The axis index.
----	-------	-----------------

Returns

Angle of the axis.

Exceptions

common::Exception (p. 314)	When _axis is invalid.
--------------------------------------	------------------------

10.68.3.3 unsigned int gazebo::physics::JointState::GetAngleCount () const

Get the number of angles.

Returns

The number of angles.

10.68.3.4 const std::vector<math::Angle>& gazebo::physics::JointState::GetAngles () const

Get the angles.

Returns

Vector of angles.

10.68.3.5 `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.

10.68.3.6 `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. 747).

10.68.3.7 `JointState gazebo::physics::JointState::operator+ (const JointState & _state) const`

Addition operator.

Parameters

<code>in</code>	<code>_pt</code>	A (p. 109) state to add.
-----------------	------------------	---------------------------------

Returns

The resulting state.

10.68.3.8 `JointState gazebo::physics::JointState::operator- (const JointState & _state) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_pt</code>	A (p. 109) state to subtract.
-----------------	------------------	--------------------------------------

Returns

The resulting state.

10.68.3.9 `JointState& gazebo::physics::JointState::operator= (const JointState & _state)`

Assignment operator.

Parameters

in	<code>_state</code>	State (p. 744) value
----	---------------------	-----------------------------

Returns

this

10.68.4 Friends And Related Function Documentation

10.68.4.1 `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. 375) state to output.

Returns

The stream.

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

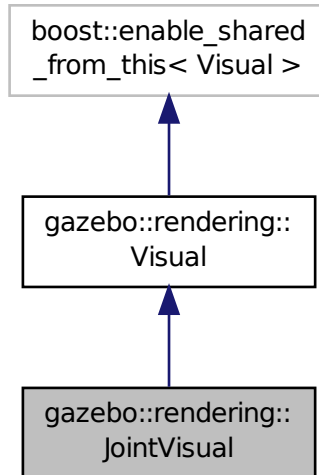
- [JointState.hh](#)

10.69 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

10.69.1 Detailed Description

Visualization for joints.

10.69.2 Constructor & Destructor Documentation

10.69.2.1 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

10.69.2.2 virtual gazebo::rendering::JointVisual::~~JointVisual() [virtual]

Destructor.

10.69.3 Member Function Documentation

10.69.3.1 void gazebo::rendering::JointVisual::Load (ConstJointPtr & _msg)

Load the visual based on a message.

Parameters

in	_msg	Joint message
----	------	---------------

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

- **JointVisual.hh**

10.70 gazebo::physics::JointWrench Class Reference

Wrench information from a joint.

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

Public Member Functions

- **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.

10.70.1 Detailed Description

Wrench information from a joint.

These are forces and torques on parent and child Links, relative to the **Link** (p. 412)'s center of mass.

10.70.2 Member Function Documentation

10.70.2.1 `JointWrench& gazebo::physics::JointWrench::operator= (const JointWrench & _wrench) [inline]`

Operator =.

Parameters

<code>in</code>	<code>_wrench</code>	Joint (p. 375) wrench to set from.
-----------------	----------------------	---

Returns

`*this`

References `body1Force`, `body1Torque`, `body2Force`, and `body2Torque`.

10.70.3 Member Data Documentation

10.70.3.1 `math::Vector3 gazebo::physics::JointWrench::body1Force`

Force on the first link.

Referenced by `operator=()`.

10.70.3.2 `math::Vector3 gazebo::physics::JointWrench::body1Torque`

Torque on the first link.

Referenced by `operator=()`.

10.70.3.3 `math::Vector3 gazebo::physics::JointWrench::body2Force`

Force on the second link.

Referenced by `operator=()`.

10.70.3.4 `math::Vector3 gazebo::physics::JointWrench::body2Torque`

Torque on the second link.

Referenced by `operator=()`.

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

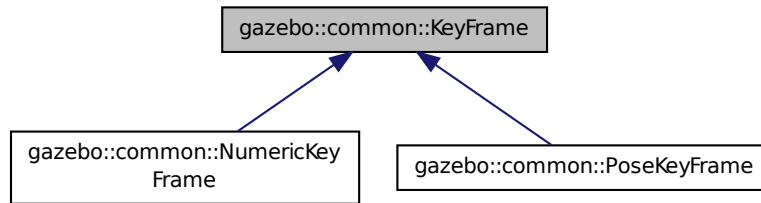
- **JointWrench.hh**

10.71 `gazebo::common::KeyFrame` Class Reference

A (p. 109) 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

10.71.1 Detailed Description

A (p. 109) key frame in an animation.

10.71.2 Constructor & Destructor Documentation

10.71.2.1 gazebo::common::KeyFrame::KeyFrame (double *_time*)

Constructor.

Parameters

in	<i>_time</i>	Time (p. 776) of the keyframe in seconds
----	--------------	---

10.71.2.2 virtual gazebo::common::KeyFrame::~~KeyFrame () [virtual]

Destructor.

10.71.3 Member Function Documentation

10.71.3.1 `double gazebo::common::KeyFrame::GetTime () const`

Get the time of the keyframe.

Returns

the time

10.71.4 Member Data Documentation

10.71.4.1 `double gazebo::common::KeyFrame::time [protected]`

time of key frame

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

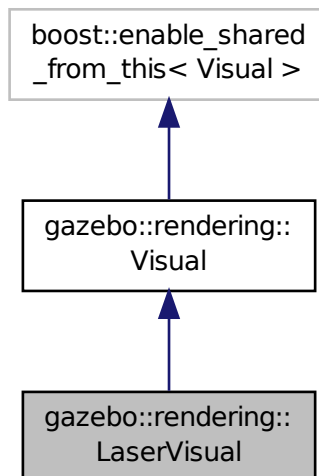
- **KeyFrame.hh**

10.72 `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

10.72.1 Detailed Description

Visualization for laser data.

10.72.2 Constructor & Destructor Documentation

10.72.2.1 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. 868).
in	<code>_topicName</code>	Name of the topic that has laser data.

10.72.2.2 virtual gazebo::rendering::LaserVisual::~~LaserVisual () [virtual]

Destructor.

10.72.3 Member Function Documentation

10.72.3.1 virtual void gazebo::rendering::LaserVisual::SetEmissive (const **common::Color** & _color) [virtual]

Documentation inherited from parent.

Reimplemented from **gazebo::rendering::Visual** (p. 883).

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

- **LaserVisual.hh**

10.73 gazebo::rendering::Light Class Reference

A (p. 109) 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.

10.73.1 Detailed Description

A (p. 109) 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.

10.73.2 Constructor & Destructor Documentation

10.73.2.1 gazebo::rendering::Light::Light (ScenePtr _scene)

Constructor.

Parameters

in	_scene	Pointer to the scene that contains the Light (p. 405).
----	--------	---

10.73.2.2 virtual gazebo::rendering::Light::~~Light () [virtual]

Destructor.

10.73.3 Member Function Documentation

10.73.3.1 void gazebo::rendering::Light::FillMsg (msgs::Light & _msg) const

Fill the contents of a light message.

Parameters

out	_msg	Message to fill.
-----	------	------------------

10.73.3.2 `common::Color gazebo::rendering::Light::GetDiffuseColor () const`

Get the diffuse color.

Returns

The light's diffuse color.

10.73.3.3 `math::Vector3 gazebo::rendering::Light::GetDirection () const`

Get the direction.

Returns

The light's direction.

10.73.3.4 `std::string gazebo::rendering::Light::GetName () const`

Get the name of the visual.

Returns

The light's name.

10.73.3.5 `math::Vector3 gazebo::rendering::Light::GetPosition () const`

Get the position of the light.

Returns

The position of the light

10.73.3.6 `common::Color gazebo::rendering::Light::GetSpecularColor () const`

Get the specular color.

Returns

The specular color

10.73.3.7 `std::string gazebo::rendering::Light::GetType () const`

Get the type of the light.

Returns

The light type: "point", "spot", "directional".

10.73.3.8 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. 405) description.
----	-------------	--

10.73.3.9 void gazebo::rendering::Light::Load ()

Load the light using default parameters.

10.73.3.10 void gazebo::rendering::Light::LoadFromMsg (ConstLightPtr & *_msg*)

Load from a light message.

Parameters

in	<i>_msg</i>	Containing the light information.
----	-------------	-----------------------------------

10.73.3.11 virtual void gazebo::rendering::Light::OnPoseChange () [inline],[protected],[virtual]

On pose change callback.

10.73.3.12 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

10.73.3.13 void gazebo::rendering::Light::SetCastShadows (const bool & *_cast*)

Set cast shadows.

Parameters

in	<i>_cast</i>	Set to true to cast shadows.
----	--------------	------------------------------

10.73.3.14 void gazebo::rendering::Light::SetDiffuseColor (const common::Color & *_color*)

Set the diffuse color.

Parameters

in	<code>_color</code>	Light (p. 405) diffuse color.
----	---------------------	--------------------------------------

10.73.3.15 `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.
----	-------------------	--

10.73.3.16 `void gazebo::rendering::Light::SetLightType (const std::string & _type)`

Set the light type.

Parameters

in	<code>_type</code>	The light type: "point", "spot", "directional"
----	--------------------	--

10.73.3.17 `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.
----	--------------------	---------------------------

10.73.3.18 `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
----	-----------------	----------------------------

10.73.3.19 `void gazebo::rendering::Light::SetRange (const double & _range)`

Set the range.

Parameters

in	<code>_range</code>	Range of the light in meters.
----	---------------------	-------------------------------

10.73.3.20 `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.
----	----	---

10.73.3.21 void gazebo::rendering::Light::SetSpecularColor (const common::Color & *_color*)

Set the specular color.

Parameters

in	_color	The specular color
----	--------	--------------------

10.73.3.22 void gazebo::rendering::Light::SetSpotFalloff (const double & *_value*)

Set the spot light falloff.

Parameters

in	_value	Falloff value
----	--------	---------------

10.73.3.23 void gazebo::rendering::Light::SetSpotInnerAngle (const double & *_angle*)

Set the spot light inner angle.

Parameters

in	_angle	Inner angle in radians
----	--------	------------------------

10.73.3.24 void gazebo::rendering::Light::SetSpotOuterAngle (const double & *_angle*)

Set the spot light outer angle.

Parameters

in	_angle	Outer angle in radians
----	--------	------------------------

10.73.3.25 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.
----	----	--

10.73.3.26 void gazebo::rendering::Light::ToggleShowVisual ()

10.73.3.27 void gazebo::rendering::Light::UpdateFromMsg (ConstLightPtr & _msg)

Update a light source from a message.

Parameters

in	_msg	Light (p. 405) message to update from
----	------	--

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

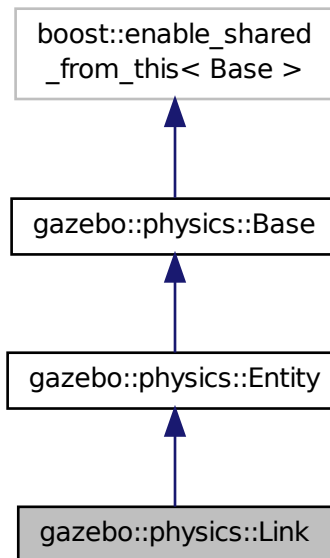
- **Light.hh**

10.74 gazebo::physics::Link Class Reference

Link (p. 412) 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. 412) as a parent **Link** (p. 412).*
- 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. 412) as a child **Link** (p. 412).*
- 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.
- **Link_V** **GetChildJointsLinks** () const
Returns a vector of children Links connected by joints.
- **CollisionPtr** **GetCollision** (const std::string &_name)
Get a child collision by name.
- **CollisionPtr** **GetCollision** (unsigned int _index) const
Get a child collision by index.
- **CollisionPtr** **GetCollisionById** (unsigned int _id) const
This is an internal function
- **Collision_V** **GetCollisions** () const
Get all the child collisions.
- virtual bool **GetEnabled** () const =0

- Get whether this body is enabled in the physics engine.*

 - virtual bool **GetGravityMode** () const =0

Get the gravity mode.
- **InertialPtr GetInertial** () const
- Get the inertia of the link.*
- virtual bool **GetKinematic** () const
- Implement this function.*
- double **GetLinearDamping** () const
- Get the linear damping factor.*
- **ModelPtr GetModel** () const
- Get the model that this body belongs to.*
- **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** ()
- 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.*
 - void **RemoveChildJoint** (JointPtr _joint)
 - Remove Joints that have this **Link** (p. 412) as a parent **Link** (p. 412).*
 - void **RemoveParentJoint** (JointPtr _joint)
 - Remove Joints that have this **Link** (p. 412) as a child **Link** (p. 412).*
 - 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 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.
- virtual void **Update** ()
Update the body.
- 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 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. 365) properties.*
- **math::Vector3** **linearAccel**
Linear acceleration.
- **std::vector**< **std::string** > **visuals**
***Link** (p. 412) visual elements.*

Additional Inherited Members

10.74.1 Detailed Description

Link (p. 412) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

10.74.2 Constructor & Destructor Documentation

10.74.2.1 gazebo::physics::Link::Link (**EntityPtr** `_parent`) [explicit]

Constructor.

Parameters

<code>in</code>	<code>_parent</code>	Parent of this link.
-----------------	----------------------	----------------------

10.74.2.2 virtual gazebo::physics::Link::~~Link () [virtual]

Destructor.

10.74.3 Member Function Documentation

10.74.3.1 void gazebo::physics::Link::AddChildJoint (JointPtr *joint*)

Joints that have this **Link** (p. 412) as a parent **Link** (p. 412).

Parameters

in	<i>joint</i>	Joint (p. 375) that is a child of this link.
----	--------------	---

10.74.3.2 virtual void gazebo::physics::Link::AddForce (const math::Vector3 & *force*) [pure virtual]

Add a force to the body.

Parameters

in	<i>force</i>	Force to add.
----	--------------	---------------

10.74.3.3 virtual void gazebo::physics::Link::AddForceAtRelativePosition (const math::Vector3 & *force*, const math::Vector3 & *relPos*) [pure virtual]

Add a force to the body at position expressed to the body's own frame of reference.

Parameters

in	<i>force</i>	Force to add.
in	<i>relPos</i>	Position on the link to add the force.

10.74.3.4 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

in	<i>force</i>	Force to add.
in	<i>pos</i>	Position in global coord frame to add the force.

10.74.3.5 void gazebo::physics::Link::AddParentJoint (JointPtr *joint*)

Joints that have this **Link** (p. 412) as a child **Link** (p. 412).

Parameters

in	<code>_joint</code>	Joint (p. 375) that is a parent of this link.
----	---------------------	--

10.74.3.6 `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

in	<code>_force</code>	Force to add.
----	---------------------	---------------

10.74.3.7 `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

in	<code>_torque</code>	Torque value to add.
----	----------------------	----------------------

10.74.3.8 `virtual void gazebo::physics::Link::AddTorque (const math::Vector3 & _torque) [pure virtual]`

Add a torque to the body.

Parameters

in	<code>_torque</code>	Torque value to add to the link.
----	----------------------	----------------------------------

10.74.3.9 `void gazebo::physics::Link::AttachStaticModel (ModelIPtr & _model, const math::Pose & _offset)`

Attach a static model to this link.

Parameters

in	<code>_model</code>	Pointer to a static model.
in	<code>_offset</code>	Pose relative to this link to place the model.

10.74.3.10 `template<typename T > event::ConnectionPtr gazebo::physics::Link::ConnectEnabled (T _subscriber) [inline]`

Connect to the add entity signal.

Parameters

in	<code>_subscriber</code>	Subscriber callback function.
----	--------------------------	-------------------------------

Returns

Pointer to the connection, which must be kept in scope.

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

10.74.3.11 void gazebo::physics::Link::DetachAllStaticModels ()

Detach all static models from this link.

10.74.3.12 void gazebo::physics::Link::DetachStaticModel (const std::string & *_modelName*)

Detach a static model from this link.

Parameters

in	<i>_modelName</i>	Name of an attached model to detach.
----	-------------------	--------------------------------------

10.74.3.13 void gazebo::physics::Link::DisconnectEnabled (event::ConnectionPtr & *_conn*) [inline]

Disconnect to the add entity signal.

Parameters

in	<i>_conn</i>	Connection pointer to disconnect.
----	--------------	-----------------------------------

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

10.74.3.14 void gazebo::physics::Link::FillMsg (msgs::Link & *_msg*)

Fill a link message.

Parameters

out	<i>_msg</i>	Message to fill
-----	-------------	-----------------

10.74.3.15 void gazebo::physics::Link::Fini () [virtual]

Finalize the body.

Reimplemented from **gazebo::physics::Entity** (p. 280).

10.74.3.16 double gazebo::physics::Link::GetAngularDamping () const

Get the angular damping factor.

Returns

Angular damping.

10.74.3.17 `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. 280).

10.74.3.18 `Link_V gazebo::physics::Link::GetChildJointsLinks () const`

Returns a vector of children Links connected by joints.

Returns

A (p. 109) vector of children Links connected by joints.

10.74.3.19 `CollisionPtr gazebo::physics::Link::GetCollision (const std::string & _name)`

Get a child collision by name.

Parameters

<code>in</code>	<code>_name</code>	Name of the collision object.
-----------------	--------------------	-------------------------------

Returns

Pointer to the collision, NULL if the name was not found.

10.74.3.20 `CollisionPtr gazebo::physics::Link::GetCollision (unsigned int _index) const`

Get a child collision by index.

Parameters

<code>in</code>	<code>_index</code>	Index of the collision object.
-----------------	---------------------	--------------------------------

Returns

Pointer to the collision, NULL if the name was not found.

10.74.3.21 `CollisionPtr gazebo::physics::Link::GetCollisionById (unsigned int _id) const`

This is an internal function

Get a collision by id.

Parameters

<code>in</code>	<code>_id</code>	Id of the collision object to find.
-----------------	------------------	-------------------------------------

Returns

Pointer to the collision, NULL if the id is invalid.

10.74.3.22 Collision_V gazebo::physics::Link::GetCollisions () const

Get all the child collisions.

Returns

A (p. 109) `std::vector` of all the child collisions.

10.74.3.23 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.

10.74.3.24 virtual bool gazebo::physics::Link::GetGravityMode () const `[pure virtual]`

Get the gravity mode.

Returns

True if gravity is enabled.

10.74.3.25 InertialPtr gazebo::physics::Link::GetInertial () const `[inline]`

Get the inertia of the link.

Returns

Inertia of the link.

References inertial.

10.74.3.26 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.

10.74.3.27 `double gazebo::physics::Link::GetLinearDamping () const`

Get the linear damping factor.

Returns

Linear damping.

10.74.3.28 `ModelPtr gazebo::physics::Link::GetModel () const`

Get the model that this body belongs to.

Returns

Model (p. 480) that this body belongs to.

10.74.3.29 `Link_V gazebo::physics::Link::GetParentJointsLinks () const`

Returns a vector of parent Links connected by joints.

Returns

Vector of parent Links connected by joints.

10.74.3.30 `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. 282).

10.74.3.31 `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. 282).

10.74.3.32 `math::Vector3 gazebo::physics::Link::GetRelativeForce () const`

Get the force applied to the body.

Returns

Force applied to the body.

10.74.3.33 `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. 282).

10.74.3.34 `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. 283).

10.74.3.35 `math::Vector3 gazebo::physics::Link::GetRelativeTorque () const`

Get the torque applied to the body.

Returns

Torque applied to the body.

10.74.3.36 `bool gazebo::physics::Link::GetSelfCollide ()`

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.

10.74.3.37 `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. 424).

Returns

The number of sensors created by the link.

10.74.3.38 `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. 423).

Note

A (p. 109) **Link** (p. 412) does not manage or maintain a pointer to a **sensors::Sensor** (p. 683). Access to a Sensor object is accomplished through the **sensors::SensorManager** (p. 695). 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.

10.74.3.39 `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. 283).

10.74.3.40 `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.

10.74.3.41 `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.

10.74.3.42 `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.

10.74.3.43 `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. 283).

10.74.3.44 `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. 412) frame will be returned.

Parameters

<code>in</code>	<code>_offset</code>	Offset of the point from the origin of the Link (p. 412) frame, expressed in the body-fixed frame.
-----------------	----------------------	---

Returns

Linear velocity of the point on the body

10.74.3.45 `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.

10.74.3.46 `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.

10.74.3.47 `virtual void gazebo::physics::Link::Init () [virtual]`

Initialize the body.

Reimplemented from **gazebo::physics::Base** (p. 143).

10.74.3.48 `virtual void gazebo::physics::Link::Load (sdf::ElementPtr _sdf) [virtual]`

Load the body based on an SDF element.

Parameters

in	_sdf	SDF parameters.
----	------	-----------------

Reimplemented from **gazebo::physics::Entity** (p. 284).

10.74.3.49 `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. 285).

10.74.3.50 `void gazebo::physics::Link::ProcessMsg (const msgs::Link & _msg)`

Update parameters from a message.

Parameters

in	_msg	Message to read.
----	------	------------------

10.74.3.51 `void gazebo::physics::Link::RemoveChildJoint (JointPtr _joint)`

Remove Joints that have this **Link** (p. 412) as a parent **Link** (p. 412).

Parameters

in	_joint	Joint (p. 375) that is a child of this link.
----	--------	---

10.74.3.52 `void gazebo::physics::Link::RemoveParentJoint (JointPtr _joint)`

Remove Joints that have this **Link** (p. 412) as a child **Link** (p. 412).

Parameters

in	_joint	Joint (p. 375) that is a parent of this link.
----	--------	--

10.74.3.53 `void gazebo::physics::Link::Reset () [virtual]`

Reset the link.

Reimplemented from **gazebo::physics::Entity** (p. 285).

10.74.3.54 void gazebo::physics::Link::ResetPhysicsStates ()

Reset the link.

10.74.3.55 void gazebo::physics::Link::SetAngularAccel (const math::Vector3 & *_accel*)

Set the angular acceleration of the body.

Parameters

in	<i>_accel</i>	Angular acceleration.
----	---------------	-----------------------

10.74.3.56 virtual void gazebo::physics::Link::SetAngularDamping (double *_damping*) [pure virtual]

Set the angular damping factor.

Parameters

in	<i>_damping</i>	Angular damping factor.
----	-----------------	-------------------------

10.74.3.57 virtual void gazebo::physics::Link::SetAngularVel (const math::Vector3 & *_vel*) [pure virtual]

Set the angular velocity of the body.

Parameters

in	<i>_vel</i>	Angular velocity.
----	-------------	-------------------

10.74.3.58 virtual void gazebo::physics::Link::SetAutoDisable (bool *_disable*) [pure virtual]

Allow the link to auto disable.

Parameters

in	<i>_disable</i>	If true, the link is allowed to auto disable.
----	-----------------	---

10.74.3.59 void gazebo::physics::Link::SetCollideMode (const std::string & *_mode*)

Set the collide mode of the body.

Parameters

in	<i>_mode</i>	Collision (p. 192) 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
----	--------------	--

10.74.3.60 `virtual void gazebo::physics::Link::SetEnabled (bool _enable) const [pure virtual]`

Set whether this body is enabled.

Parameters

<code>in</code>	<code><i>_enable</i></code>	True to enable the link in the physics engine.
-----------------	-----------------------------	--

10.74.3.61 `virtual void gazebo::physics::Link::SetForce (const math::Vector3 & _force) [pure virtual]`

Set the force applied to the body.

Parameters

<code>in</code>	<code><i>_force</i></code>	Force value.
-----------------	----------------------------	--------------

10.74.3.62 `virtual void gazebo::physics::Link::SetGravityMode (bool _mode) [pure virtual]`

Set whether gravity affects this body.

Parameters

<code>in</code>	<code><i>_mode</i></code>	True to enable gravity.
-----------------	---------------------------	-------------------------

10.74.3.63 `void gazebo::physics::Link::SetInertial (const InertialPtr & _inertial)`

Set the mass of the link.

[in] `_inertial` **Inertial** (p. 365) value for the link.

10.74.3.64 `virtual void gazebo::physics::Link::SetKinematic (const bool & _kinematic) [virtual]`

Implement this function.

Set whether this body is in the kinematic state.

Parameters

<code>in</code>	<code><i>_kinematic</i></code>	True to make the link kinematic only.
-----------------	--------------------------------	---------------------------------------

10.74.3.65 `void gazebo::physics::Link::SetLaserRetro (float _retro)`

Set the laser retro reflectiveness.

Parameters

<code>in</code>	<code><i>_retro</i></code>	Retro value for all child collisions.
-----------------	----------------------------	---------------------------------------

10.74.3.66 `void gazebo::physics::Link::SetLinearAccel (const math::Vector3 & _accel)`

Set the linear acceleration of the body.

Parameters

in	_accel	Linear acceleration.
----	--------	----------------------

10.74.3.67 `virtual void gazebo::physics::Link::SetLinearDamping (double _damping) [pure virtual]`

Set the linear damping factor.

Parameters

in	_damping	Linear damping factor.
----	----------	------------------------

10.74.3.68 `virtual void gazebo::physics::Link::SetLinearVel (const math::Vector3 & _vel) [pure virtual]`

Set the linear velocity of the body.

Parameters

in	_vel	Linear velocity.
----	------	------------------

10.74.3.69 `virtual bool gazebo::physics::Link::SetSelected (bool _set) [virtual]`

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

Parameters

in	_set	True to set the link as selected.
----	------	-----------------------------------

Reimplemented from **gazebo::physics::Base** (p. 145).

10.74.3.70 `virtual void gazebo::physics::Link::SetSelfCollide (bool _collide) [pure virtual]`

Set whether this body will collide with others in the model.

Parameters

in	_collid	True to enable collisions.
----	---------	----------------------------

10.74.3.71 `void gazebo::physics::Link::SetState (const LinkState & _state)`

Set the current link state.

Parameters

in	_state	The state to set the link to.
----	--------	-------------------------------

10.74.3.72 `virtual void gazebo::physics::Link::SetTorque (const math::Vector3 & _torque) [pure virtual]`

Set the torque applied to the body.

Parameters

in	_torque	Torque value.
----	---------	---------------

10.74.3.73 `virtual void gazebo::physics::Link::Update () [virtual]`

Update the body.

Reimplemented from `gazebo::physics::Base` (p. 146).

10.74.3.74 `virtual void gazebo::physics::Link::UpdateMass () [inline],[virtual]`

Update the mass matrix.

10.74.3.75 `virtual void gazebo::physics::Link::UpdateParameters (sdf::ElementPtr _sdf) [virtual]`

Update the parameters using new sdf values.

Parameters

in	_sdf	SDF values to load from.
----	------	--------------------------

Reimplemented from `gazebo::physics::Entity` (p. 287).

10.74.3.76 `virtual void gazebo::physics::Link::UpdateSurface () [inline],[virtual]`

Update surface parameters.

10.74.4 Member Data Documentation

10.74.4.1 `math::Vector3 gazebo::physics::Link::angularAccel [protected]`

Angular acceleration.

10.74.4.2 `std::vector<math::Pose> gazebo::physics::Link::attachedModelsOffset [protected]`

Offsets for the attached models.

10.74.4.3 `std::vector<std::string> gazebo::physics::Link::cgVisuals [protected]`

Center of gravity visual elements.

10.74.4.4 InertialPtr gazebo::physics::Link::inertial [protected]

Inertial (p. 365) properties.

Referenced by GetInertial().

10.74.4.5 math::Vector3 gazebo::physics::Link::linearAccel [protected]

Linear acceleration.

10.74.4.6 std::vector<std::string> gazebo::physics::Link::visuals [protected]

Link (p. 412) visual elements.

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

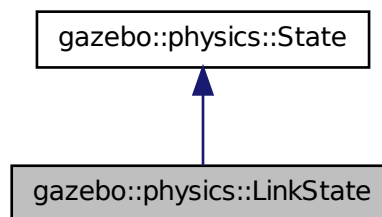
- **Link.hh**

10.75 gazebo::physics::LinkState Class Reference

Store state information of a **physics::Link** (p. 412) object.

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

Inheritance diagram for gazebo::physics::LinkState:



Public Member Functions

- **LinkState** ()
Default 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. 412).*
- 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.
- **LinkState** **operator+** (const **LinkState** &_state) const
Addition operator.
- **LinkState** **operator-** (const **LinkState** &_state) const
Subtraction operator.
- **LinkState** & **operator=** (const **LinkState** &_state)
Assignment operator.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::LinkState** &_state)
Stream insertion operator.

Additional Inherited Members

10.75.1 Detailed Description

Store state information of a **physics::Link** (p. 412) object.

This class captures the entire state of a **Link** (p. 412) at one specific time during a simulation run.

State (p. 744) of a **Link** (p. 412) includes the state of itself all its child **Collision** (p. 192) entities.

10.75.2 Constructor & Destructor Documentation

10.75.2.1 gazebo::physics::LinkState::LinkState ()

Default constructor.

10.75.2.2 gazebo::physics::LinkState::LinkState (const LinkPtr *link*) [explicit]

Constructor.

Build a **LinkState** (p. 431) from an existing **Link** (p. 412).

Parameters

in	<i>_model</i>	Pointer to the Link (p. 412) from which to gather state info.
----	---------------	--

10.75.2.3 gazebo::physics::LinkState::LinkState (const sdf::ElementPtr *sdf*) [explicit]

Constructor.

Build a **LinkState** (p. 431) from SDF data

Parameters

in	<i>_sdf</i>	SDF data to load a link state from.
----	-------------	-------------------------------------

10.75.2.4 virtual gazebo::physics::LinkState::~~LinkState () [virtual]

Destructor.

10.75.3 Member Function Documentation

10.75.3.1 void gazebo::physics::LinkState::FillSDF (sdf::ElementPtr *sdf*)

Populate a state SDF element with data from the object.

Parameters

out	<i>_sdf</i>	SDF element to populate.
-----	-------------	--------------------------

10.75.3.2 const math::Pose& gazebo::physics::LinkState::GetAcceleration () const

Get the link acceleration.

Returns

The acceleration represented as a **math::Pose** (p. 584).

10.75.3.3 CollisionState gazebo::physics::LinkState::GetCollisionState (unsigned int *_index*) const

Get a collision state.

Get a **Collision** (p. 192) **State** (p. 744) based on an index, where index is in the range of 0...**LinkState::GetCollisionStateCount** (p. 434).

Parameters

<i>in</i>	<i>_index</i>	Index of the CollisionState (p. 201).
-----------	---------------	--

Returns

State (p. 744) of the **Collision** (p. 192).

Exceptions

<i>common::Exception</i> (p. 314)	When <i>_index</i> is invalid.
---	--------------------------------

10.75.3.4 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. 201) with the matching name, if any.

Parameters

<i>in</i>	<i>_collisionName</i>	Name of the CollisionState (p. 201)
-----------	-----------------------	--

Returns

State (p. 744) of the **Collision** (p. 192).

Exceptions

<i>common::Exception</i> (p. 314)	When <i>_collisionName</i> is invalid
---	---------------------------------------

10.75.3.5 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. 201) recorded.

10.75.3.6 const std::vector<CollisionState>& gazebo::physics::LinkState::GetCollisionStates () const

Get the collision states.

Returns

A (p. 109) vector of collision states.

10.75.3.7 `const math::Pose& gazebo::physics::LinkState::GetPose () const`

Get the link pose.

Returns

The **math::Pose** (p. 584) of the **Link** (p. 412).

10.75.3.8 `const math::Pose& gazebo::physics::LinkState::GetVelocity () const`

Get the link velocity.

Returns

The velocity represented as a **math::Pose** (p. 584).

10.75.3.9 `const math::Pose& gazebo::physics::LinkState::GetWrench () const`

Get the force applied to the **Link** (p. 412).

Returns

Magnitude of the force.

10.75.3.10 `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.

10.75.3.11 `virtual void gazebo::physics::LinkState::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Load **LinkState** (p. 431) 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. 747).

10.75.3.12 `LinkState gazebo::physics::LinkState::operator+ (const LinkState & _state) const`

Addition operator.

Parameters

in	_pt	A (p. 109) state to add.
----	-----	---------------------------------

Returns

The resulting state.

10.75.3.13 `LinkState gazebo::physics::LinkState::operator- (const LinkState & _state) const`

Subtraction operator.

Parameters

in	_pt	A (p. 109) state to subtract.
----	-----	--------------------------------------

Returns

The resulting state.

10.75.3.14 `LinkState& gazebo::physics::LinkState::operator= (const LinkState & _state)`

Assignment operator.

Parameters

in	_state	State (p. 744) value
----	--------	-----------------------------

Returns

this

10.75.4 Friends And Related Function Documentation

10.75.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::physics::LinkState & _state)` [friend]

Stream insertion operator.

Parameters

in	_out	output stream
in	_state	Link (p. 412) state to output

Returns

the stream

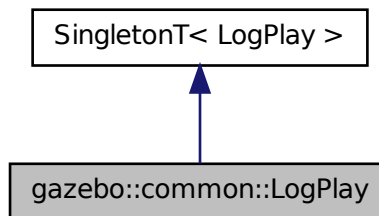
The documentation for this class was generated from the following file:

- **LinkState.hh**

10.76 gazebo::common::LogPlay Class Reference

```
#include <LogPlay.hh>
```

Inheritance diagram for gazebo::common::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 **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

10.76.1 Member Function Documentation

10.76.1.1 `bool gazebo::common::LogPlay::GetChunk (unsigned int _index, std::string & _data)`

Get data for a particular chunk index.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the chunk.
<code>out</code>	<code><i>_data</i></code>	Storage for the chunk's data.

Returns

True if the `_index` was valid.

10.76.1.2 `unsigned int gazebo::common::LogPlay::GetChunkCount () const`

Get the number of chunks (steps) in the open log file.

Returns

The number of recorded states in the log file.

10.76.1.3 `std::string gazebo::common::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. 439) has not been called at least once.

10.76.1.4 `std::string gazebo::common::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.

10.76.1.5 `std::string gazebo::common::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.

10.76.1.6 `uint32_t gazebo::common::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. 625).

10.76.1.7 `bool gazebo::common::LogPlay::IsOpen () const`

Return true if a file is open.

Returns

True if a log file is open.

10.76.1.8 `void gazebo::common::LogPlay::Open (const std::string & _logFile)`

Open a log file for reading.

Open a log file that was previously recorded.

Parameters

in	<code>_logFile</code>	The file to load
----	-----------------------	------------------

Exceptions

<i>Exception</i> (p. 314)

10.76.1.9 `bool gazebo::common::LogPlay::Step (std::string & _data)`

Step through the open log file.

Parameters

out	<code>_data</code>	Data from next entry in the log file.
-----	--------------------	---------------------------------------

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

- **LogPlay.hh**

10.77 Logplay Class Reference

Open and playback log files that were recorded using LogRecord.

10.77.1 Detailed Description

Open and playback log files that were recorded using LogRecord.

Use **Logplay** (p. 439) 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:

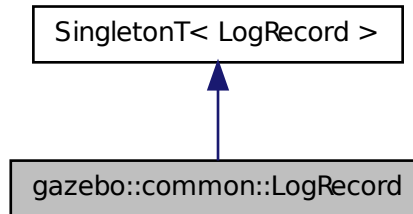
- **LogPlay.hh**

10.78 gazebo::common::LogRecord Class Reference

```
addtogroup gazebo_common
```

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

Inheritance diagram for gazebo::common::LogRecord:



Public Member Functions

- void **Add** (const std::string &_name, const std::string &_filename, boost::function< bool(std::ostringstream &)> _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.
- 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 **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="bz2")
Start the logger.
- void **Stop** ()
Stop the logger.

Additional Inherited Members

10.78.1 Detailed Description

addtogroup gazebo_common

Handles logging of data to disk

The **LogRecord** (p. 440) 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. 437) for playback functionality.

State information for an entity may be logged through the **LogRecord::Add** (p. 441) function, and stopped through the **LogRecord::Remove** (p. 444) function. Data may be logged into a single file, or split into many separate files by specifying different filenames for the **LogRecord::Add** (p. 441) function.

The **LogRecord** (p. 440) is updated at the start of each simulation step. This guarantees that all data is stored.

See Also

Logplay (p. 439), State

10.78.2 Member Function Documentation

10.78.2.1 void gazebo::common::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

<code>in</code>	<code>_name</code>	Name of the object to log.
<code>in</code>	<code>_filename</code>	Filename of the log file.
<code>in</code>	<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> (p. 314)

10.78.2.2 `void gazebo::common::LogRecord::Fini ()`

Finalize, and shutdown.

10.78.2.3 `std::string gazebo::common::LogRecord::GetBasePath () const`

Get the base path for a log recording.

Returns

Path for log recording.

10.78.2.4 `const std::string& gazebo::common::LogRecord::GetEncoding () const`

Get the encoding used.

Returns

Either [txt, or bz2], where txt is plain txt and bz2 is bzip2 compressed data with Base64 encoding.

10.78.2.5 `std::string gazebo::common::LogRecord::GetFilename (const std::string & _name) const`

Get the filename for a log object.

Parameters

<code>in</code>	<code>_name</code>	Name of the log object.
-----------------	--------------------	-------------------------

Returns

Filename, empty string if not found.

10.78.2.6 `unsigned int gazebo::common::LogRecord::GetFileSize (const std::string & _name) const`

Get the file size for a log object.

Parameters

in	<i>_name</i>	Name of the log object.
----	--------------	-------------------------

Returns

Size in bytes.

10.78.2.7 bool gazebo::common::LogRecord::GetFirstUpdate () const

Return true if an Update has not yet been completed.

Returns

True if an Update has not yet been completed.

10.78.2.8 bool gazebo::common::LogRecord::GetPaused () const

Get whether logging is paused.

Returns

True if logging is paused.

See Also

LogRecord::SetPaused (p. 444)

10.78.2.9 bool gazebo::common::LogRecord::GetRunning () const

Get whether logging is running.

Returns

True if logging has been started.

10.78.2.10 common::Time gazebo::common::LogRecord::GetRunTime () const

Get the run time in sim time.

Returns

Run sim time.

10.78.2.11 bool gazebo::common::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.

10.78.2.12 `bool gazebo::common::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.

10.78.2.13 `void gazebo::common::LogRecord::SetBasePath (const std::string & _path)`

Set the base path.

Parameters

<code>in</code>	<code>_path</code>	Path to the new logging location.
-----------------	--------------------	-----------------------------------

10.78.2.14 `void gazebo::common::LogRecord::SetPaused (bool _paused)`

Set whether logging should pause.

A (p. 109) 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. 443)

10.78.2.15 `bool gazebo::common::LogRecord::Start (const std::string & _encoding = "bz2")`

Start the logger.

Parameters

in	<code>_encoding</code>	The type of encoding (txt, or bz2).
----	------------------------	-------------------------------------

10.78.2.16 void gazebo::common::LogRecord::Stop ()

Stop the logger.

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

- **LogRecord.hh**

10.79 gazebo::Master Class Reference

A (p. 109) 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 \sim **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.

10.79.1 Detailed Description

A (p. 109) 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. 445), runs World update and sensor generation loops.

10.79.2 Constructor & Destructor Documentation

10.79.2.1 gazebo::Master::Master ()

Constructor.

10.79.2.2 virtual gazebo::Master::~~Master () [virtual]

Destructor.

10.79.3 Member Function Documentation

10.79.3.1 void gazebo::Master::Fini ()

Finalize the master.

10.79.3.2 void gazebo::Master::Init (uint16_t *_port*)

Initialize.

Parameters

<i>in</i>	<i>_port</i>	The master's port
-----------	--------------	-------------------

10.79.3.3 void gazebo::Master::Run ()

Run the master.

10.79.3.4 void gazebo::Master::RunOnce ()

Run the master one iteration.

10.79.3.5 void gazebo::Master::RunThread ()

Run the master in a new thread.

10.79.3.6 void gazebo::Master::Stop ()

Stop the master.

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

- **Master.hh**

10.80 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.

10.80.1 Detailed Description

Encapsulates description of a material.

10.80.2 Member Enumeration Documentation

10.80.2.1 enum gazebo::common::Material::BlendMode

Enumerator:

ADD

MODULATE

REPLACE

BLEND_COUNT

10.80.2.2 enum gazebo::common::Material::ShadeMode

Enumerator:

FLAT

GOURAUD

PHONG

BLINN

SHADE_COUNT

10.80.3 Constructor & Destructor Documentation

10.80.3.1 gazebo::common::Material::Material ()

Constructor.

10.80.3.2 `virtual gazebo::common::Material::~Material () [virtual]`

Destructor.

10.80.3.3 `gazebo::common::Material::Material (const Color & _clr)`

Create a material with a default color.

Parameters

<code>in</code>	<code>_clr</code>	Color (p. 205) of the material
-----------------	-------------------	---------------------------------------

10.80.4 Member Function Documentation

10.80.4.1 `Color gazebo::common::Material::GetAmbient () const`

Get the ambient color.

Returns

The ambient color

10.80.4.2 `void gazebo::common::Material::GetBlendFactors (double & _srcFactor, double & _dstFactor)`

Get the blend factors.

Parameters

<code>in</code>	<code>_srcFactor</code>	Source factor is returned in this variable
<code>in</code>	<code>_dstFactor</code>	Destination factor is returned in this variable

10.80.4.3 `BlendMode gazebo::common::Material::GetBlendMode () const`

Get the blending mode.

Returns

the blend mode

10.80.4.4 `bool gazebo::common::Material::GetDepthWrite () const`

Get depth write.

Returns

the depth write enabled state

10.80.4.5 `Color gazebo::common::Material::GetDiffuse () const`

Get the diffuse color.

Returns

The diffuse color

10.80.4.6 Color gazebo::common::Material::GetEmissive () const

Get the emissive color.

Returns

The emissive color

10.80.4.7 bool gazebo::common::Material::GetLighting () const

Get lighting enabled.

Returns

the lighting enabled state

10.80.4.8 std::string gazebo::common::Material::GetName () const

Get the name of the material.

Returns

The name of the material

10.80.4.9 double gazebo::common::Material::GetPointSize () const

Get the point size.

Returns

the point size

10.80.4.10 ShadeMode gazebo::common::Material::GetShadeMode () const

Get the shading mode.

Returns

the shading mode

10.80.4.11 double gazebo::common::Material::GetShininess () const

Get the shininess.

Returns

The shininess value

10.80.4.12 `Color gazebo::common::Material::GetSpecular () const`

Get the specular color.

Returns

The specular color

10.80.4.13 `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

10.80.4.14 `double gazebo::common::Material::GetTransparency () const`

Get the transparency percentage (0..1)

Returns

The transparency percentage

10.80.4.15 `void gazebo::common::Material::SetAmbient (const Color & _clr)`

Set the ambient color.

Parameters

<i>in</i>	<i>_clr</i>	The ambient color
-----------	-------------	-------------------

10.80.4.16 `void gazebo::common::Material::SetBlendFactors (double _srcFactor, double _dstFactor)`

Set the blende factors.

Will be interpreted as: (texture * *_srcFactor*) + (scene_pixel * *_dstFactor*)

Parameters

<i>in</i>	<i>_srcFactor</i>	The source factor
<i>in</i>	<i>_dstFactor</i>	The destination factor

10.80.4.17 `void gazebo::common::Material::SetBlendMode (BlendMode _b)`

Set the blending mode.

Parameters

in	<code>_b</code>	the blend mode
----	-----------------	----------------

10.80.4.18 void gazebo::common::Material::SetDepthWrite (bool *_value*)

Set depth write.

Parameters

in	<code>_value</code>	the depth write enabled state
----	---------------------	-------------------------------

10.80.4.19 void gazebo::common::Material::SetDiffuse (const Color & *_clr*)

Set the diffuse color.

Parameters

in	<code>_clr</code>	The diffuse color
----	-------------------	-------------------

10.80.4.20 void gazebo::common::Material::SetEmissive (const Color & *_clr*)

Set the emissive color.

Parameters

in	<code>_clr</code>	The emissive color
----	-------------------	--------------------

10.80.4.21 void gazebo::common::Material::SetLighting (bool *_value*)

Set lighting enabled.

Parameters

in	<code>_value</code>	the lighting enabled state
----	---------------------	----------------------------

10.80.4.22 void gazebo::common::Material::SetPointSize (double *_size*)

Set the point size.

Parameters

in	<code>_size</code>	the size
----	--------------------	----------

10.80.4.23 void gazebo::common::Material::SetShadeMode (ShadeMode *_b*)

Set the shading mode param[in] the shading mode.

10.80.4.24 void gazebo::common::Material::SetShininess (double *_t*)

Set the shininess.

Parameters

in	<i>_t</i>	The shininess value
----	-----------	---------------------

10.80.4.25 void gazebo::common::Material::SetSpecular (const Color & *_clr*)

Set the specular color.

Parameters

in	<i>_clr</i>	The specular color
----	-------------	--------------------

10.80.4.26 void gazebo::common::Material::SetTextureImage (const std::string & *_tex*)

Set a texture image.

Parameters

in	<i>_tex</i>	The name of the texture, which must be in Gazebo's resource path
----	-------------	--

10.80.4.27 void gazebo::common::Material::SetTextureImage (const std::string & *_tex*, const std::string & *_resourcePath*)

Set a texture image.

Parameters

in	<i>_tex</i>	The name of the texture
in	<i>_resourcePath</i>	Path which contains <i>_tex</i>

10.80.4.28 void gazebo::common::Material::SetTransparency (double *_t*)

Set the transparency percentage (0..1)

Parameters

in	<i>_t</i>	The amount of transparency (0..1)
----	-----------	-----------------------------------

10.80.5 Friends And Related Function Documentation

10.80.5.1 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.

10.80.6 Member Data Documentation

10.80.6.1 **Color** gazebo::common::Material::ambient [protected]

the ambient light color

10.80.6.2 **BlendMode** gazebo::common::Material::blendMode [protected]

blend mode

10.80.6.3 **std::string** gazebo::common::Material::BlendModeStr[**BLEND_COUNT**] [static]

10.80.6.4 **Color** gazebo::common::Material::diffuse [protected]

the diffuse lighth color

10.80.6.5 **Color** gazebo::common::Material::emissive [protected]

the emissive light color

10.80.6.6 **std::string** gazebo::common::Material::name [protected]

the name of the material

10.80.6.7 **double** gazebo::common::Material::pointSize [protected]

point size

10.80.6.8 **ShadeMode** gazebo::common::Material::shadeMode [protected]

the shade mode

10.80.6.9 **std::string** gazebo::common::Material::ShadeModeStr[**SHADE_COUNT**] [static]

10.80.6.10 **double** gazebo::common::Material::shininess [protected]

shininess value (0 to 1)

10.80.6.11 **Color** gazebo::common::Material::specular [protected]

the specular light color

10.80.6.12 **std::string** gazebo::common::Material::texImage [protected]

the texture image file name

10.80.6.13 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**

10.81 gazebo::math::Matrix3 Class Reference

A (p. 109) 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.
- 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

- std::ostream & **operator<<** (std::ostream &_out, const gazebo::math::**Matrix3** &_m)
Stream insertion operator.

10.81.1 Detailed Description

A (p. 109) 3x3 matrix class.

10.81.2 Constructor & Destructor Documentation

10.81.2.1 gazebo::math::Matrix3::Matrix3 ()

Constructor.

10.81.2.2 gazebo::math::Matrix3::Matrix3 (const Matrix3 & _m)

Copy constructor.

Parameters

_m	Matrix to copy
----	----------------

10.81.2.3 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

10.81.2.4 virtual gazebo::math::Matrix3::~~Matrix3 () [virtual]

Destructor.

10.81.3 Member Function Documentation

10.81.3.1 bool gazebo::math::Matrix3::operator==(const Matrix3 & _m) const

Equality test operator.

Parameters

in	_m	Matrix3 (p. 456) to test
----	----	---------------------------------

Returns

True if equal (using the default tolerance of 1e-6)

10.81.3.2 `const double* gazebo::math::Matrix3::operator[] (size_t _row) const` `[inline]`

Array subscript operator.

Parameters

<code>in</code>	<code>_row</code>	row index
-----------------	-------------------	-----------

Returns

a pointer to the row

References m.

10.81.3.3 `double* gazebo::math::Matrix3::operator[] (size_t _row)` `[inline]`

Array subscript operator.

Parameters

<code>in</code>	<code>_row</code>	row index
-----------------	-------------------	-----------

Returns

a pointer to the row

References m.

10.81.3.4 `void gazebo::math::Matrix3::SetCol (unsigned int _c, const Vector3 & _v)`

Set a column.

Parameters

<code>in</code>	<code>_c</code>	The column index (0, 1, 2)
<code>in</code>	<code>_v</code>	The value to set in each row of the column

10.81.3.5 `void gazebo::math::Matrix3::SetFromAxes (const Vector3 & _xAxis, const Vector3 & _yAxis, const Vector3 & _zAxis)`

Set the matrix from three axis (1 per column)

Parameters

<code>in</code>	<code>_xAxis</code>	The x axis
<code>in</code>	<code>_yAxis</code>	The y axis
<code>in</code>	<code>_zAxis</code>	The z axis

10.81.3.6 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

10.81.4 Friends And Related Function Documentation

10.81.4.1 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

10.81.5 Member Data Documentation

10.81.5.1 double gazebo::math::Matrix3::m[3][3] [protected]

the 3x3 matrix

Referenced by operator[[]].

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

- **Matrix3.hh**

10.82 gazebo::math::Matrix4 Class Reference

A (p. 109) 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 \sim **Matrix4** ()
- Destructor.
- **math::Pose GetAsPose** () const
 - Get the transformation as **math::Pose** (p. 584).
- **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. 839).
- **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.

10.82.1 Detailed Description

A (p. 109) 3x3 matrix class.

10.82.2 Constructor & Destructor Documentation

10.82.2.1 gazebo::math::Matrix4::Matrix4 ()

Constructor.

10.82.2.2 gazebo::math::Matrix4::Matrix4 (const Matrix4 & _m)

Copy constructor.

Parameters

<code>_m</code>	Matrix to copy
-----------------	----------------

10.82.2.3 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	<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

10.82.2.4 `virtual gazebo::math::Matrix4::~~Matrix4 () [virtual]`

Destructor.

10.82.3 Member Function Documentation

10.82.3.1 `math::Pose gazebo::math::Matrix4::GetAsPose () const`

Get the transformation as **math::Pose** (p. 584).

Returns

the pose

10.82.3.2 `Vector3 gazebo::math::Matrix4::GetEulerRotation (unsigned int solution_number = 1) const`

Get the rotation as a Euler angles.

Returns

the rotation

10.82.3.3 `Quaternion gazebo::math::Matrix4::GetRotation () const`

Get the rotation as a quaternion.

Returns

the rotation

10.82.3.4 `Vector3 gazebo::math::Matrix4::GetTranslation () const`

Get the translational values as a **Vector3** (p. 839).

Returns

x,y,z

10.82.3.5 `Matrix4 gazebo::math::Matrix4::Inverse () const`

Return the inverse matrix.

10.82.3.6 bool gazebo::math::Matrix4::IsAffine () const

Return true if the matrix is affine.

Returns

true if the matrix is affine, false otherwise

10.82.3.7 Matrix4 gazebo::math::Matrix4::operator* (const Matrix4 & _mat) const

Multiplication operator.

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

Returns

This matrix * `_mat`

10.82.3.8 Matrix4 gazebo::math::Matrix4::operator* (const Matrix3 & _mat) const

Multiplication operator.

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

Returns

This matrix * `_mat`

10.82.3.9 Vector3 gazebo::math::Matrix4::operator* (const Vector3 & _vec) const

Multiplication operator.

Parameters

<code>_vec</code>	Vector3 (p. 839)
-------------------	-------------------------

Returns

Resulting vector from multiplication

10.82.3.10 Matrix4& gazebo::math::Matrix4::operator= (const Matrix4 & _mat)

Equal operator.

this = `_mat`

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

Returns

itself

10.82.3.11 `const Matrix4& gazebo::math::Matrix4::operator= (const Matrix3 & _mat)`

Equal operator for 3x3 matrix.

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

Returns

itself

10.82.3.12 `bool gazebo::math::Matrix4::operator==(const Matrix4 & _m) const`

Equality operator.

Parameters

<code>in</code>	<code>_m</code>	Matrix3 (p. 456) to test
-----------------	-----------------	---------------------------------

Returns

true if the 2 matrices are equal (using the tolerance 1e-6), false otherwise

10.82.3.13 `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.

10.82.3.14 `const double* gazebo::math::Matrix4::operator[](size_t _row) const [inline]`

Parameters

in	<code>_row</code>	the row index
----	-------------------	---------------

Returns

the row

References m.

10.82.3.15 `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

10.82.3.16 `void gazebo::math::Matrix4::SetScale (const Vector3 & _s)`

Set the scale.

Parameters

in	<code>_s</code>	scale
----	-----------------	-------

10.82.3.17 `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
----	-----------------	---------------

10.82.3.18 Vector3 gazebo::math::Matrix4::TransformAffine (const Vector3 & _v) const

Perform an affine transformation.

Parameters

<code>_v</code>	Vector3 (p. 839) value for the transformation
-----------------	--

Returns

The result of the transformation

10.82.4 Friends And Related Function Documentation

10.82.4.1 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

10.82.5 Member Data Documentation

10.82.5.1 const Matrix4 gazebo::math::Matrix4::IDENTITY [static]

Identity matrix.

10.82.5.2 double gazebo::math::Matrix4::m[4][4] [protected]

The 4x4 matrix.

Referenced by operator[[]].

10.82.5.3 const Matrix4 gazebo::math::Matrix4::ZERO [static]

Zero matrix.

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

- **Matrix4.hh**

10.83 gazebo::common::Mesh Class Reference

A (p. 109) 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 **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.
- 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.

10.83.1 Detailed Description

A (p. 109) 3D mesh.

10.83.2 Constructor & Destructor Documentation

10.83.2.1 gazebo::common::Mesh::Mesh ()

Constructor.

10.83.2.2 virtual gazebo::common::Mesh::~~Mesh () [virtual]

Destructor.

10.83.3 Member Function Documentation

10.83.3.1 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

10.83.3.2 void gazebo::common::Mesh::AddSubMesh (**SubMesh** * `_child`)

Add a submesh mesh.

The **Mesh** (p. 466) object takes ownership of the submesh.

Parameters

<code>in</code>	<code>_child</code>	the submesh
-----------------	---------------------	-------------

10.83.3.3 void gazebo::common::Mesh::FillArrays (float ** *_vertArr*, int ** *_indArr*) const

Put all the data into flat arrays.

Parameters

out	<i>_vertArr</i>	the vertex array
out	<i>_indArr</i>	the index array

10.83.3.4 void gazebo::common::Mesh::GenSphericalTexCoord (const math::Vector3 & *_center*)

Generate texture coordinates using spherical projection from center.

Parameters

in	<i>_center</i>	the center of the projection
----	----------------	------------------------------

10.83.3.5 void gazebo::common::Mesh::GetAABB (math::Vector3 & *_center*, math::Vector3 & *_min_xyz*, math::Vector3 & *_max_xyz*) const

Get AABB coordinate.

Parameters

out	<i>_center</i>	of the bounding box
out	<i>_min_xyz</i>	bounding box minimum values
out	<i>_max_xyz</i>	bounding box maximum values

10.83.3.6 unsigned int gazebo::common::Mesh::GetIndexCount () const

Return the number of indices.

Returns

the count

10.83.3.7 const Material* gazebo::common::Mesh::GetMaterial (int *_index*) const

Get a material.

Parameters

in	<i>_index</i>	the index
----	---------------	-----------

Returns

the material or NULL if the index is out of bounds

10.83.3.8 `unsigned int gazebo::common::Mesh::GetMaterialCount () const`

Get the number of materials.

Returns

the count

10.83.3.9 `math::Vector3 gazebo::common::Mesh::GetMax () const`

Get the maximum X, Y, Z values.

Returns

the upper bounds of the bounding box

10.83.3.10 `math::Vector3 gazebo::common::Mesh::GetMin () const`

Get the minimum X, Y, Z values.

Returns

the lower bounds of the bounding box

10.83.3.11 `std::string gazebo::common::Mesh::GetName () const`

Get the name of this mesh.

Returns

the name

10.83.3.12 `unsigned int gazebo::common::Mesh::GetNormalCount () const`

Return the number of normals.

Returns

the count

10.83.3.13 `std::string gazebo::common::Mesh::GetPath () const`

Get the path which contains the mesh resource.

Returns

the path to the mesh resource

10.83.3.14 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.

10.83.3.15 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

10.83.3.16 unsigned int gazebo::common::Mesh::GetSubMeshCount () const

Get the number of children.

Returns

the count

10.83.3.17 unsigned int gazebo::common::Mesh::GetTexCoordCount () const

Return the number of texture coordinates.

Returns

the count

10.83.3.18 unsigned int gazebo::common::Mesh::GetVertexCount () const

Return the number of vertices.

Returns

the count

10.83.3.19 bool gazebo::common::Mesh::HasSkeleton () const

Return true if mesh is attached to a skeleton.

10.83.3.20 void gazebo::common::Mesh::RecalculateNormals ()

Recalculate all the normals of each face defined by three indices.

10.83.3.21 void gazebo::common::Mesh::Scale (double *_factor*)

Scale all vertices by *_factor*.

Parameters

<i>_factor</i>	Scaling factor
----------------	----------------

10.83.3.22 void gazebo::common::Mesh::SetName (const std::string & *_n*)

Set the name of this mesh.

Parameters

in	<i>_n</i>	the name to set
----	-----------	-----------------

10.83.3.23 void gazebo::common::Mesh::SetPath (const std::string & *_path*)

Set the path which contains the mesh resource.

Parameters

in	<i>_path</i>	the file path
----	--------------	---------------

10.83.3.24 void gazebo::common::Mesh::SetScale (const math::Vector3 & *_factor*)

Scale all vertices by the *_factor* vector.

Parameters

in	<i>_factor</i>	Scaling vector
----	----------------	----------------

10.83.3.25 void gazebo::common::Mesh::SetSkeleton (Skeleton * *_skel*)

Set the mesh skeleton.

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

- **Mesh.hh**

10.84 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.

10.84.1 Detailed Description

Creates CSG meshes.

10.84.2 Member Enumeration Documentation

10.84.2.1 enum gazebo::common::MeshCSG::BooleanOperation

An enumeration of the boolean operations.

Enumerator:

UNION
INTERSECTION
DIFFERENCE

10.84.3 Constructor & Destructor Documentation

10.84.3.1 gazebo::common::MeshCSG::MeshCSG ()

Constructor.

10.84.3.2 virtual gazebo::common::MeshCSG::~~MeshCSG () [virtual]

Destructor.

10.84.4 Member Function Documentation

10.84.4.1 **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	<code>_m1</code>	the parent mesh in the boolean operation
in	<code>_m2</code>	the child mesh in the boolean operation
in	<code>_operation</code>	the boolean operation applied to the two meshes
in	<code>_offset</code>	<code>_m2</code> 's pose offset from <code>_m1</code>

Returns

a pointer to the created mesh

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

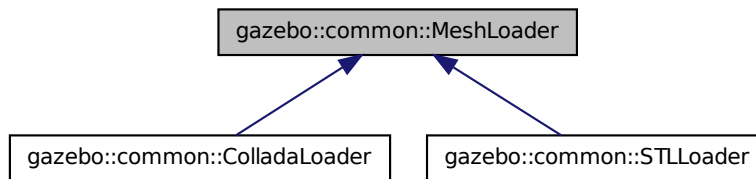
- **MeshCSG.hh**

10.85 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.

10.85.1 Detailed Description

Base class for loading meshes.

10.85.2 Constructor & Destructor Documentation

10.85.2.1 gazebo::common::MeshLoader::MeshLoader ()

Constructor.

10.85.2.2 virtual gazebo::common::MeshLoader::~~MeshLoader () [virtual]

Destructor.

10.85.3 Member Function Documentation

10.85.3.1 virtual Mesh* gazebo::common::MeshLoader::Load (const std::string & *filename*) [pure virtual]

Load a 3D mesh.

Parameters

in	<i>_filename</i>	the path to the mesh
----	------------------	----------------------

Returns

a pointer to the created mesh

Implemented in [gazebo::common::ColladaLoader](#) (p. 191), and [gazebo::common::STLLoader](#) (p. 749).

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

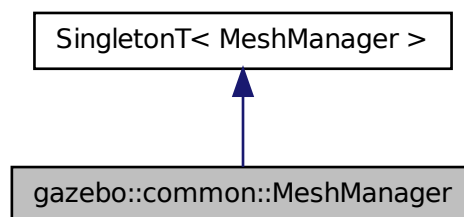
- [MeshLoader.hh](#)

10.86 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

10.86.1 Detailed Description

Maintains and manages all meshes.

10.86.2 Member Function Documentation

10.86.2.1 void gazebo::common::MeshManager::AddMesh (Mesh * _mesh)

Add a mesh to the manager.

This **MeshManager** (p. 475) takes ownership of the mesh and will destroy it. See ~MeshManager.

Parameters

in	<i>the</i>	mesh to add.
----	------------	--------------

10.86.2.2 void gazebo::common::MeshManager::CreateBox (const std::string & _name, const math::Vector3 & _sides, const math::Vector2d & _uvCoords)

Create a Box mesh.

Parameters

in	<i>_name</i>	the name of the new mesh
in	<i>_sides</i>	the x y x dimentions of eah side in meter
in	<i>_uvCoords</i>	the texture coordinates

10.86.2.3 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

10.86.2.4 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

10.86.2.5 void gazebo::common::MeshManager::CreateCylinder (const std::string & _name, float _radius, float _height, int _rings, int _segments)

Create a cylinder mesh.

Parameters

in	<code>_name</code>	the name of the new mesh
in	<code>_radius</code>	the radius of the cylinder in the x y plane
in	<code>_height</code>	the height along z
in	<code>_rings</code>	the number of circles along the height
in	<code>_segments</code>	the number of segment per circle

10.86.2.6 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	<code>_name</code>	
in	<code>_plane</code>	plane parameters
in	<code>_segments</code>	number of segments in x and y
in	<code>_uvTile</code>	the texture tile size in x and y

10.86.2.7 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	<code>_name</code>	the name of the new mesh
in	<code>_normal</code>	the normal to the plane
in	<code>_d</code>	distance from the origin along normal
in	<code>_size</code>	the size of the plane in x and y
in	<code>_segments</code>	the number of segments in x and y
in	<code>_uvTile</code>	the texture tile size in x and y

10.86.2.8 void gazebo::common::MeshManager::CreateSphere (const std::string & *_name*, float *_radius*, int *_rings*, int *_segments*)

Create a sphere mesh.

Parameters

in	<code>_name</code>	the name of the mesh
in	<code>_radius</code>	radius of the sphere in meter
in	<code>_rings</code>	number of circles on th y axis
in	<code>_segments</code>	number of segment per circle

10.86.2.9 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>_outerRadius</i>	the outer radius of the tube in the x y plane
in	<i>_height</i>	the height along z
in	<i>_rings</i>	the number of circles along the height
in	<i>_segments</i>	the number of segment per circle

10.86.2.10 void gazebo::common::MeshManager::GenSphericalTexCoord (const Mesh * *_mesh*, math::Vector3 *_center*)

generate spherical texture coordinates

10.86.2.11 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

10.86.2.12 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

10.86.2.13 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
----	--------------	----------------------

10.86.2.14 `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

10.86.2.15 `const Mesh* gazebo::common::MeshManager::Load (const std::string & _filename)`

Load a mesh from a file.

Parameters

<code>in</code>	<code>_filename</code>	the path to the mesh
-----------------	------------------------	----------------------

Returns

a pointer to the created mesh

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

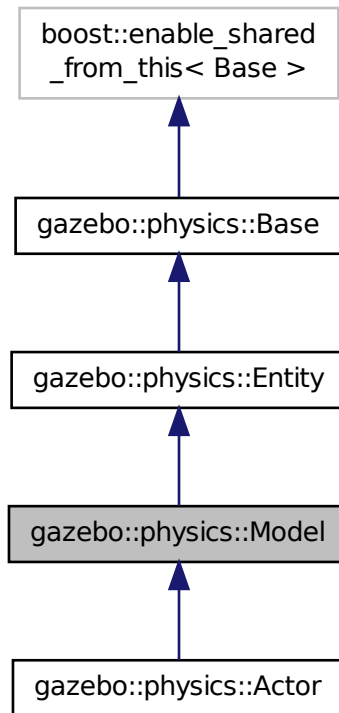
- **MeshManager.hh**

10.87 gazebo::physics::Model Class Reference

A (p. 109) 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.
- 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.

- **JointPtr GetJoint** (const std::string &name)
Get a joint.
- **JointControllerPtr GetJointController** ()
Get a handle to the Controller for the joints in this model.
- unsigned int **GetJointCount** () const
Get the number of joints.
- const **Joint_V & GetJoints** () const
Get the joints.
- **LinkPtr GetLink** (const std::string &_name="canonical") const
Get a link by name.
- **LinkPtr GetLinkById** (unsigned int _id) const
This is an internal function
- **Link_V GetLinks** () const
*Construct and return a vector of **Link** (p. 412)'s in this model Note this constructs the vector of **Link** (p. 412)'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. 412), 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. 375) Animation.*
- void **SetJointPosition** (const std::string &_jointName, double _position)
 - Set the positions of a **Joint** (p. 375) 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. 480) by specifying desired Pose of a **Link** (p. 412) within the **Model** (p. 480).*
- void **SetLinkWorldPose** (const **math::Pose** &_pose, const **LinkPtr** &_link)
 - Set the Pose of the entire **Model** (p. 480) by specifying desired Pose of a **Link** (p. 412) within the **Model** (p. 480).*
- 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. 484)
- `std::vector< math::Pose > attachedModelsOffset`
used by `Model::AttachStaticModel` (p. 484)

Additional Inherited Members

10.87.1 Detailed Description

A (p. 109) model is a collection of links, joints, and plugins.

10.87.2 Constructor & Destructor Documentation

10.87.2.1 `gazebo::physics::Model::Model (BasePtr _parent) [explicit]`

Constructor.

Parameters

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

10.87.2.2 `virtual gazebo::physics::Model::~~Model () [virtual]`

Destructor.

10.87.3 Member Function Documentation

10.87.3.1 `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. 480), which is a **Model** (p. 480) that has been marked as static (no physics simulation), and attaches it to this **Model** (p. 480) 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

<code>in</code>	<code><i>_model</i></code>	Pointer to the static model.
<code>in</code>	<code><i>_offset</i></code>	Offset, relative to this Model (p. 480), to place <i>_model</i> .

10.87.3.2 `void gazebo::physics::Model::DetachStaticModel (const std::string & _model)`

Detach a static model from this model.

Parameters

in	<i>_model</i>	Name of an attached static model to remove.
----	---------------	---

See Also

Model::AttachStaticModel (p. 484).

10.87.3.3 void gazebo::physics::Model::FillMsg (msgs::Model & *_msg*)

Fill a model message.

Parameters

in	<i>_msg</i>	Message to fill using this model's data.
----	-------------	--

10.87.3.4 virtual void gazebo::physics::Model::Fini () [virtual]

Finalize the model.

Reimplemented from **gazebo::physics::Entity** (p. 280).

Reimplemented in **gazebo::physics::Actor** (p. 112).

10.87.3.5 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.

10.87.3.6 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. 280).

10.87.3.7 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

10.87.3.8 `JointControllerPtr gazebo::physics::Model::GetJointController () [inline]`

Get a handle to the Controller for the joints in this model.

Returns

A (p. 109) handle to the Controller for the joints in this model.

10.87.3.9 `unsigned int gazebo::physics::Model::GetJointCount () const`

Get the number of joints.

Returns

Get the number of joints.

10.87.3.10 `const Joint_V& gazebo::physics::Model::GetJoints () const`

Get the joints.

Returns

Vector of joints.

10.87.3.11 `LinkPtr gazebo::physics::Model::GetLink (const std::string & _name = "canonical") const`

Get a link by name.

Parameters

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

Returns

Pointer to the link, NULL if the name is invalid.

10.87.3.12 `LinkPtr gazebo::physics::Model::GetLinkById (unsigned int _id) const`

This is an internal function

Get a link by id.

Returns

Pointer to the link, NULL if the id is invalid.

10.87.3.13 `Link_V gazebo::physics::Model::GetLinks () const`

Construct and return a vector of **Link** (p. 412)'s in this model Note this constructs the vector of **Link** (p. 412)'s on the fly, could be costly.

Returns

a vector of **Link** (p. 412)'s in this model

10.87.3.14 `unsigned int gazebo::physics::Model::GetPluginCount () const`

Get the number of plugins this model has.

Returns

Number of plugins associated with this model.

10.87.3.15 `virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularAccel () const [virtual]`

Get the angular acceleration of the entity.

Returns

math::Vector3 (p. 839), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 282).

10.87.3.16 `virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularVel () const [virtual]`

Get the angular velocity of the entity.

Returns

math::Vector3 (p. 839), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 282).

10.87.3.17 `virtual math::Vector3 gazebo::physics::Model::GetRelativeLinearAccel () const [virtual]`

Get the linear acceleration of the entity.

Returns

math::Vector3 (p. 839), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 282).

10.87.3.18 virtual **math::Vector3** gazebo::physics::Model::GetRelativeLinearVel () const [virtual]

Get the linear velocity of the entity.

Returns

math::Vector3 (p. 839), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 283).

10.87.3.19 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. 142).

Reimplemented in **gazebo::physics::Actor** (p. 112).

10.87.3.20 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.

10.87.3.21 virtual **math::Vector3** gazebo::physics::Model::GetWorldAngularAccel () const [virtual]

Get the angular acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 839), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 283).

10.87.3.22 virtual **math::Vector3** gazebo::physics::Model::GetWorldAngularVel () const [virtual]

Get the angular velocity of the entity in the world frame.

Returns

math::Vector3 (p. 839), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 283).

10.87.3.23 virtual **math::Vector3** gazebo::physics::Model::GetWorldLinearAccel () const [virtual]

Get the linear acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 839), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 283).

10.87.3.24 virtual **math::Vector3** gazebo::physics::Model::GetWorldLinearVel () const [virtual]

Get the linear velocity of the entity in the world frame.

Returns

math::Vector3 (p. 839), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 284).

10.87.3.25 virtual void gazebo::physics::Model::Init () [virtual]

Initialize the model.

Reimplemented from **gazebo::physics::Base** (p. 143).

Reimplemented in **gazebo::physics::Actor** (p. 113).

10.87.3.26 void gazebo::physics::Model::Load (sdf::ElementPtr _sdf) [virtual]

Load the model.

Parameters

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

Reimplemented from **gazebo::physics::Entity** (p. 284).

10.87.3.27 void gazebo::physics::Model::LoadJoints ()

Load all the joints.

10.87.3.28 void gazebo::physics::Model::LoadPlugins ()

Load all plugins.

Load all plugins specified in the SDF for the model.

10.87.3.29 virtual void gazebo::physics::Model::OnPoseChange () [protected],[virtual]

Callback when the pose of the model has been changed.

Implements **gazebo::physics::Entity** (p. 285).

10.87.3.30 `void gazebo::physics::Model::ProcessMsg (const msgs::Model & _msg)`

Update parameters from a model message.

Parameters

in	<code>_msg</code>	Message to process.
----	-------------------	---------------------

10.87.3.31 `virtual void gazebo::physics::Model::RemoveChild (EntityPtr _child) [virtual]`

Remove a child.

Parameters

in	<code>_child</code>	Remove a child entity.
----	---------------------	------------------------

10.87.3.32 `void gazebo::physics::Model::Reset () [virtual]`

Reset the model.

Reimplemented from **gazebo::physics::Entity** (p. 285).

10.87.3.33 `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
----	-------------------	------------------------------

10.87.3.34 `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.
----	-------------------	---------------------------

10.87.3.35 `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.
----	-----------------------	--

10.87.3.36 void gazebo::physics::Model::SetCollideMode (const std::string & *_mode*)

This is not implemented in **Link** (p. 412), which means this function doesn't do anything.

Set the collide mode of the model.

Parameters

in	<i>_mode</i>	The collision mode
----	--------------	--------------------

10.87.3.37 void gazebo::physics::Model::SetEnabled (bool *_enabled*)

Enable all the links in all the models.

Parameters

in	<i>_enabled</i>	True to enable all the links.
----	-----------------	-------------------------------

10.87.3.38 void gazebo::physics::Model::SetGravityMode (const bool & *_value*)

Set the gravity mode of the model.

Parameters

in	<i>_value</i>	False to turn gravity on for the model.
----	---------------	---

10.87.3.39 void gazebo::physics::Model::SetJointAnimation (const std::map< std::string, common::NumericAnimationPtr > *_anim*, boost::function< void()> *_onComplete* = NULL)

Joint (p. 375) 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.

10.87.3.40 void gazebo::physics::Model::SetJointPosition (const std::string & *_jointName*, double *_position*)

Set the positions of a **Joint** (p. 375) by name.

See Also

JointController::SetJointPosition (p. 394)

Parameters

in	<i>_jointName</i>	Name of the joint to set.
in	<i>_position</i>	Position to set the joint to.

10.87.3.41 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. 394).

Parameters

in	<i>_jointPositions</i>	Map of joint names to their positions.
----	------------------------	--

10.87.3.42 void gazebo::physics::Model::SetLaserRetro (const float *_retro*)

Set the laser retro reflectiveness of the model.

Parameters

in	<i>_retro</i>	Retro reflectance value.
----	---------------	--------------------------

10.87.3.43 void gazebo::physics::Model::SetLinearAccel (const math::Vector3 & *_vel*)

Set the linear acceleration of the model, and all its links.

Parameters

in	<i>_vel</i>	The new linear acceleration.
----	-------------	------------------------------

10.87.3.44 void gazebo::physics::Model::SetLinearVel (const math::Vector3 & *_vel*)

Set the linear velocity of the model, and all its links.

Parameters

in	<i>_vel</i>	The new linear velocity.
----	-------------	--------------------------

10.87.3.45 void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & *_pose*, std::string *_linkName*)

Set the Pose of the entire **Model** (p. 480) by specifying desired Pose of a **Link** (p. 412) within the **Model** (p. 480).

Doing so, keeps the configuration of the **Model** (p. 480) unchanged, i.e. all **Joint** (p. 375) angles are unchanged.

Parameters

in	<i>_pose</i>	Pose to set the link to.
in	<i>_linkName</i>	Name of the link to set.

10.87.3.46 void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & *_pose*, const LinkPtr & *_link*)

Set the Pose of the entire **Model** (p. 480) by specifying desired Pose of a **Link** (p. 412) within the **Model** (p. 480). Doing so, keeps the configuration of the **Model** (p. 480) unchanged, i.e. all **Joint** (p. 375) angles are unchanged.

Parameters

in	<i>_pose</i>	Pose to set the link to.
in	<i>_link</i>	Pointer to the link to set.

10.87.3.47 void gazebo::physics::Model::SetState (const ModelState & *_state*)

Set the current model state.

Parameters

in	<i>_state</i>	State (p. 744) to set the model to.
----	---------------	--

10.87.3.48 virtual void gazebo::physics::Model::StopAnimation () [virtual]

Stop the current animations.

Reimplemented from **gazebo::physics::Entity** (p. 287).

10.87.3.49 void gazebo::physics::Model::Update () [virtual]

Update the model.

Reimplemented from **gazebo::physics::Base** (p. 146).

10.87.3.50 virtual void gazebo::physics::Model::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. 287).

Reimplemented in **gazebo::physics::Actor** (p. 113).

10.87.4 Member Data Documentation

10.87.4.1 std::vector<ModelPtr> gazebo::physics::Model::attachedModels [protected]

used by **Model::AttachStaticModel** (p. 484)

10.87.4.2 `std::vector<math::Pose> gazebo::physics::Model::attachedModelsOffset` [protected]

used by `Model::AttachStaticModel` (p. 484)

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

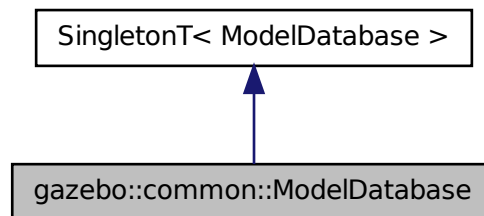
- `Model.hh`

10.88 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.
- std::string **GetDBConfig** (const std::string &_uri)
Return the database.config file as a string.
- std::string **GetManifest** (const std::string &_uri) **GAZEBO_DEPRECATED**
Deprecated.
- 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.

Additional Inherited Members

10.88.1 Detailed Description

Connects to model database, and has utility functions to find models.

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

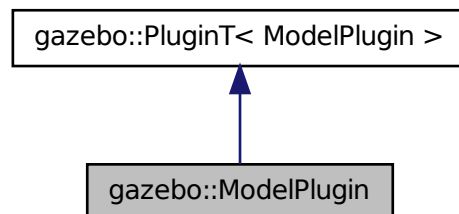
- **ModelDatabase.hh**

10.89 gazebo::ModelPlugin Class Reference

A (p. 109) plugin with access to **physics::Model** (p. 480).

```
#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

10.89.1 Detailed Description

A (p. 109) plugin with access to **physics::Model** (p. 480).

See reference.

10.89.2 Constructor & Destructor Documentation

10.89.2.1 gazebo::ModelPlugin::ModelPlugin () [inline]

Constructor.

References gazebo::MODEL_PLUGIN, and gazebo::PluginT< ModelPlugin >::type.

10.89.2.2 virtual gazebo::ModelPlugin::~~ModelPlugin () [inline],[virtual]

Destructor.

10.89.3 Member Function Documentation

10.89.3.1 virtual void gazebo::ModelPlugin::Init () [inline],[virtual]

Override this method for custom plugin initialization behavior.

10.89.3.2 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.

10.89.3.3 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:

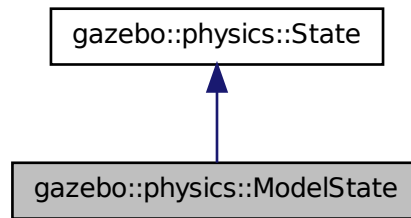
- **common/Plugin.hh**

10.90 gazebo::physics::ModelState Class Reference

Store state information of a **physics::Model** (p. 480) object.

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

Inheritance diagram for gazebo::physics::ModelState:



Public Member Functions

- **ModelState** ()
Default 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. 375) state.*
- **JointState GetJointState** (const std::string &_jointName) const
*Get a **Joint** (p. 375) state by **Joint** (p. 375) name.*
- unsigned int **GetJointStateCount** () const
Get the number of joint states.
- const std::vector< **JointState** > & **GetJointStates** () const
Get the joint states.
- **LinkState GetLinkState** (unsigned int _index) const
Get a link state.
- **LinkState GetLinkState** (const std::string &_linkName) const
*Get a link state by **Link** (p. 412) name.*
- unsigned int **GetLinkStateCount** () const
Get the number of link states.
- const std::vector< **LinkState** > & **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.
- 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.

Friends

- std::ostream & **operator<<** (std::ostream &_out, const **gazebo::physics::ModelState** &_state)

Stream insertion operator.

Additional Inherited Members

10.90.1 Detailed Description

Store state information of a **physics::Model** (p. 480) object.

This class captures the entire state of a **Model** (p. 480) at one specific time during a simulation run.

State (p. 744) of a **Model** (p. 480) includes the state of all its child Links and Joints.

10.90.2 Constructor & Destructor Documentation

10.90.2.1 gazebo::physics::ModelState::ModelState ()

Default constructor.

10.90.2.2 gazebo::physics::ModelState::ModelState (const ModelPtr _model) [explicit]

Constructor.

Build a **ModelState** (p. 497) from an existing **Model** (p. 480).

Parameters

in	_model	Pointer to the model from which to gather state info.
----	--------	---

10.90.2.3 gazebo::physics::ModelState::ModelState (const sdf::ElementPtr *_sdf*) [explicit]

Constructor.

Build a **ModelState** (p. 497) from SDF data

Parameters

in	<i>_sdf</i>	SDF data to load a model state from.
----	-------------	--------------------------------------

10.90.2.4 virtual gazebo::physics::ModelState::~~ModelState () [virtual]

Destructor.

10.90.3 Member Function Documentation

10.90.3.1 void gazebo::physics::ModelState::FillSDF (sdf::ElementPtr *_sdf*)

Populate a state SDF element with data from the object.

Parameters

out	<i>_sdf</i>	SDF element to populate.
-----	-------------	--------------------------

10.90.3.2 JointState gazebo::physics::ModelState::GetJointState (unsigned int *_index*) const

Get a **Joint** (p. 375) state.

Return a **JointState** (p. 395) based on a index, where index is between 0...**ModelState::GetJointStateCount()** (p. 500).

Parameters

in	<i>_index</i>	Index of a JointState (p. 395).
----	---------------	--

Returns

State (p. 744) of a **Joint** (p. 375).

Exceptions

common::Exception (p. 314)	When <i>_index</i> is out of range.
--------------------------------------	-------------------------------------

10.90.3.3 JointState gazebo::physics::ModelState::GetJointState (const std::string & *_jointName*) const

Get a **Joint** (p. 375) state by **Joint** (p. 375) name.

Searches through all JointStates. Returns the **JointState** (p. 395) with the matching name, if any.

Parameters

in	<code>_jointName</code>	Name of the JointState (p. 395).
----	-------------------------	---

Returns

State (p. 744) of the **Joint** (p. 375).

Exceptions

<i>common::Exception</i> (p. 314)	When <code>_jointName</code> is invalid.
---	--

10.90.3.4 unsigned int gazebo::physics::ModelState::GetJointStateCount () const

Get the number of joint states.

Returns the number of JointStates recorded.

Returns

Number of JointStates.

10.90.3.5 const std::vector<JointState>& gazebo::physics::ModelState::GetJointStates () const

Get the joint states.

Returns

A (p. 109) vector of joint states.

10.90.3.6 LinkState gazebo::physics::ModelState::GetLinkState (unsigned int _index) const

Get a link state.

Get a **Link** (p. 412) **State** (p. 744) based on an index, where index is in the range of 0...**ModelState::GetLinkStateCount** (p. 501)

Parameters

in	<code>_index</code>	Index of the LinkState (p. 431)
----	---------------------	--

Returns

State (p. 744) of the **Link** (p. 412).

Exceptions

<i>common::Exception</i> (p. 314)	When <code>_index</code> is out of range.
---	---

10.90.3.7 LinkState gazebo::physics::ModelState::GetLinkState (const std::string & *_linkName*) const

Get a link state by **Link** (p. 412) name.

Searches through all LinkStates. Returns the **LinkState** (p. 431) with the matching name, if any.

Parameters

<i>in</i>	<i>_linkName</i>	Name of the LinkState (p. 431)
-----------	------------------	---------------------------------------

Returns

State (p. 744) of the **Link** (p. 412).

Exceptions

<i>common::Exception</i> (p. 314)	When <i>_linkName</i> is invalid.
---	-----------------------------------

10.90.3.8 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. 431) recorded.

10.90.3.9 const std::vector<LinkState>& gazebo::physics::ModelState::GetLinkStates () const

Get the link states.

Returns

A (p. 109) vector of link states.

10.90.3.10 const math::Pose& gazebo::physics::ModelState::GetPose () const

Get the stored model pose.

Returns

The **math::Pose** (p. 584) of the **Model** (p. 480).

10.90.3.11 bool gazebo::physics::ModelState::HasJointState (const std::string & *_jointName*) const

Return true if there is a joint with the specified name.

Parameters

in	<i>_jointName</i>	Name of the Jointtate.
----	-------------------	------------------------

Returns

True if the joint exists in the model.

10.90.3.12 `bool gazebo::physics::ModelState::HasLinkState (const std::string & _linkName) const`

Return true if there is a link with the specified name.

Parameters

in	<i>_linkName</i>	Name of the LinkState (p. 431).
----	------------------	--

Returns

True if the link exists in the model.

10.90.3.13 `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.

10.90.3.14 `virtual void gazebo::physics::ModelState::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Load **ModelState** (p. 497) 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. 747).

10.90.3.15 `ModelState gazebo::physics::ModelState::operator+ (const ModelState & _state) const`

Addition operator.

Parameters

in	<i>_pt</i>	A (p. 109) state to subtract.
----	------------	--------------------------------------

Returns

The resulting state.

10.90.3.16 ModelState gazebo::physics::ModelState::operator- (const ModelState & _state) const

Subtraction operator.

Parameters

<i>in</i>	<i>_pt</i>	A (p. 109) state to subtract.
-----------	------------	--------------------------------------

Returns

The resulting state.

10.90.3.17 ModelState& gazebo::physics::ModelState::operator= (const ModelState & _state)

Assignment operator.

Parameters

<i>in</i>	<i>_state</i>	State (p. 744) value
-----------	---------------	-----------------------------

Returns

this

10.90.4 Friends And Related Function Documentation**10.90.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::physics::ModelState & _state) [friend]**

Stream insertion operator.

Parameters

<i>in</i>	<i>_out</i>	output stream.
<i>in</i>	<i>_state</i>	Model (p. 480) state to output.

Returns

The stream.

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

- **ModelState.hh**

10.91 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.

10.91.1 Detailed Description

Generic description of a mouse event.

10.91.2 Member Enumeration Documentation

10.91.2.1 enum gazebo::common::MouseEvent::Buttons

Standard mouse buttons enumeration.

Enumerator:

NO_BUTTON
LEFT
MIDDLE
RIGHT

10.91.2.2 enum gazebo::common::MouseEvent::EventType

Mouse event types enumeration.

Enumerator:

NO_EVENT
MOVE
PRESS
RELEASE
SCROLL

10.91.3 Constructor & Destructor Documentation

10.91.3.1 gazebo::common::MouseEvent::MouseEvent () [inline]

Constructor.

10.91.4 Member Data Documentation

10.91.4.1 bool gazebo::common::MouseEvent::alt

Alt key press flag.

10.91.4.2 unsigned int gazebo::common::MouseEvent::button

The button which caused the event.

10.91.4.3 unsigned int gazebo::common::MouseEvent::buttons

State of the buttons when the event was generated.

10.91.4.4 bool gazebo::common::MouseEvent::control

Control key press flag.

10.91.4.5 `bool gazebo::common::MouseEvent::dragging`

Flag for mouse drag motion.

10.91.4.6 `float gazebo::common::MouseEvent::moveScale`

Scaling factor.

10.91.4.7 `math::Vector2i gazebo::common::MouseEvent::pos`

Mouse pointer position on the screen.

10.91.4.8 `math::Vector2i gazebo::common::MouseEvent::pressPos`

Position of button press.

10.91.4.9 `math::Vector2i gazebo::common::MouseEvent::prevPos`

Previous position.

10.91.4.10 `math::Vector2i gazebo::common::MouseEvent::scroll`

Scroll position.

10.91.4.11 `bool gazebo::common::MouseEvent::shift`

Shift key press flag.

10.91.4.12 `EventType gazebo::common::MouseEvent::type`

Event type.

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

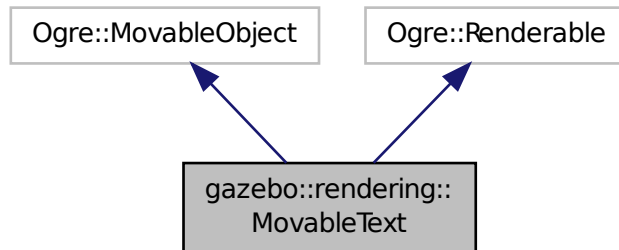
- **MouseEvent.hh**

10.92 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 ontop.*
- 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

10.92.1 Detailed Description

Movable text.

10.92.2 Member Enumeration Documentation

10.92.2.1 enum gazebo::rendering::MovableText::HorizAlign

Horizontal alignment.

Enumerator:

H_LEFT Left alignment.

H_CENTER Center alignment.

10.92.2.2 enum gazebo::rendering::MovableText::VertAlign

vertical alignment

Enumerator:

V_BELOW Align below.

V_ABOVE Align above.

10.92.3 Constructor & Destructor Documentation

10.92.3.1 gazebo::rendering::MovableText::MovableText ()

Constructor.

10.92.3.2 virtual gazebo::rendering::MovableText::~~MovableText () [virtual]

Destructor.

10.92.4 Member Function Documentation

10.92.4.1 void gazebo::rendering::MovableText::setupGeometry () [protected]

10.92.4.2 void gazebo::rendering::MovableText::updateColors () [protected]

10.92.4.3 math::Box gazebo::rendering::MovableText::GetAABB ()

Get the axis aligned bounding box of the text.

Returns

The axis aligned bounding box.

10.92.4.4 float gazebo::rendering::MovableText::GetBaseline () const

Get the baseline height.

Returns

Baseline height

10.92.4.5 float gazebo::rendering::MovableText::getBoundingRadius () const [protected]

10.92.4.6 float gazebo::rendering::MovableText::GetCharHeight () const

Set the height of a characters return Height of the characters.

10.92.4.7 `const common::Color& gazebo::rendering::MovableText::GetColor () const`

Get the text color.

Returns

Texture color.

10.92.4.8 `const std::string& gazebo::rendering::MovableText::GetFont () const`

Get the font.

Returns

The font name

10.92.4.9 `const Ogre::LightList& gazebo::rendering::MovableText::getLights (void) const` [protected]

10.92.4.10 `const Ogre::MaterialPtr& gazebo::rendering::MovableText::getMaterial (void) const` [protected]

10.92.4.11 `void gazebo::rendering::MovableText::getRenderOperation (Ogre::RenderOperation & op)` [protected]

10.92.4.12 `bool gazebo::rendering::MovableText::GetShowOnTop () const`

True = text is displayed on top.

Returns

True if `MovableText::SetShownOnTop(true)` was called.

10.92.4.13 `float gazebo::rendering::MovableText::GetSpaceWidth () const`

Get the width of a space.

Returns

Space width

10.92.4.14 `float gazebo::rendering::MovableText::getSquaredViewDepth (const Ogre::Camera * cam) const` [protected]

10.92.4.15 `const std::string& gazebo::rendering::MovableText::GetText () const`

Get the displayed text.

Returns

The displayed text.

10.92.4.16 void gazebo::rendering::MovableText::getWorldTransforms (Ogre::Matrix4 * *xform*) const [protected]

10.92.4.17 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

10.92.4.18 void gazebo::rendering::MovableText::SetBaseline (float *_height*)

Set the baseline height of the text.

Parameters

in	<i>_height</i>	Baseline height
----	----------------	-----------------

10.92.4.19 void gazebo::rendering::MovableText::SetCharHeight (float *_height*)

Set the height of a character.

Parameters

in	<i>_height</i>	Height of the characters.
----	----------------	---------------------------

10.92.4.20 void gazebo::rendering::MovableText::SetColor (const common::Color & *_color*)

Set the text color.

Parameters

in	<i>_color</i>	Text color.
----	---------------	-------------

10.92.4.21 void gazebo::rendering::MovableText::SetFontName (const std::string & *_font*)

Set the font.

Parameters

in	<i>_font</i>	Name of the font
----	--------------	------------------

10.92.4.22 `void gazebo::rendering::MovableText::SetShowOnTop (bool _show)`

True = text always is displayed on top.

Parameters

<code>in</code>	<code><i>_show</i></code>	Set to true to render the text on top of all other drawables.
-----------------	---------------------------	---

10.92.4.23 `void gazebo::rendering::MovableText::SetSpaceWidth (float _width)`

Set the width of a space.

Parameters

<code>in</code>	<code><i>_width</i></code>	space width
-----------------	----------------------------	-------------

10.92.4.24 `void gazebo::rendering::MovableText::SetText (const std::string & _text)`

Set the text to display.

Parameters

<code>in</code>	<code><i>_text</i></code>	The text to display.
-----------------	---------------------------	----------------------

10.92.4.25 `void gazebo::rendering::MovableText::SetTextAlignment (const HorizAlign & _hAlign, const VertAlign & _vAlign)`

Set the alignment of the text.

Parameters

<code>in</code>	<code><i>_hAlign</i></code>	Horizontal alignment
<code>in</code>	<code><i>_vAlign</i></code>	Vertical alignment

10.92.4.26 `void gazebo::rendering::MovableText::Update ()`

Update the text.

10.92.4.27 `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**

10.93 gazebo::msgs::MsgFactory Class Reference

A (p. 109) 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 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.

10.93.1 Detailed Description

A (p. 109) factory that generates protobuf message based on a string type.

10.93.2 Member Function Documentation

10.93.2.1 static void gazebo::msgs::MsgFactory::GetMsgTypes (std::vector< std::string > &_types) [static]

Get all the message types.

Parameters

out	<code>_types</code>	Vector of strings of the message types.
-----	---------------------	---

10.93.2.2 static google::protobuf::Message* gazebo::msgs::MsgFactory::NewMsg (const std::string &_msgType) [static]

Create a new instance of a message.

Parameters

in	<code>_msgType</code>	Type of message to create.
----	-----------------------	----------------------------

Returns

Pointer to a google protobuf message. Null if the message type could not be handled.

10.93.2.3 static void gazebo::msgs::MsgFactory::RegisterMsg (const std::string &_msgType, MsgFactoryFn _factoryfn) [static]

Register a message.

Parameters

in	<code>_msgType</code>	Type of message to register.
in	<code>_factoryfn</code>	Function that generates the message.

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

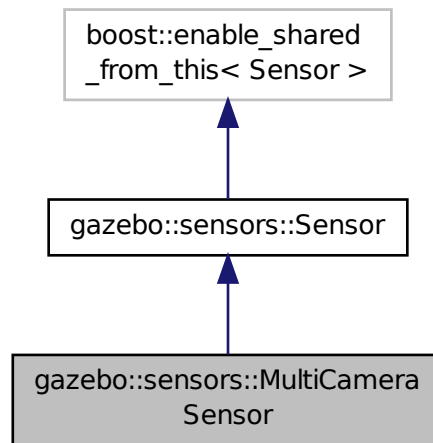
- [MsgFactory.hh](#)

10.94 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. 159).*
- 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

10.94.1 Detailed Description

Multiple camera sensor.

This sensor type can create one or more synchronized cameras.

10.94.2 Constructor & Destructor Documentation

10.94.2.1 gazebo::sensors::MultiCameraSensor::MultiCameraSensor ()

Constructor.

10.94.2.2 virtual gazebo::sensors::MultiCameraSensor::~~MultiCameraSensor () [virtual]

Destructor.

10.94.3 Member Function Documentation

10.94.3.1 virtual void gazebo::sensors::MultiCameraSensor::Fini () [protected],[virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 687).

10.94.3.2 rendering::CameraPtr gazebo::sensors::MultiCameraSensor::GetCamera (unsigned int _index) const

Returns a pointer to a **rendering::Camera** (p. 159).

Parameters

in	<i>_index</i>	Index of the camera to get
----	---------------	----------------------------

Returns

The Pointer to the camera sensor.

See Also

MultiCameraSensor::GetCameraCount (p. 516)

10.94.3.3 `unsigned int gazebo::sensors::MultiCameraSensor::GetCameraCount () const`

Get the number of cameras.

Returns

The number of cameras.

10.94.3.4 `const unsigned char* gazebo::sensors::MultiCameraSensor::GetImageData (unsigned int _index)`

Gets the raw image data from the sensor.

Parameters

in	<i>_index</i>	Index of the camera
----	---------------	---------------------

Returns

The pointer to the image data array.

See Also

MultiCameraSensor::GetCameraCount (p. 516)

10.94.3.5 `unsigned int gazebo::sensors::MultiCameraSensor::GetImageHeight (unsigned int _index) const`

Gets the height of the image in pixels.

Parameters

in	<i>_index</i>	Index of the camera
----	---------------	---------------------

Returns

The image height in pixels.

See Also

MultiCameraSensor::GetCameraCount (p. 516)

10.94.3.6 unsigned int gazebo::sensors::MultiCameraSensor::GetImageWidth (unsigned int *_index*) const

Gets the width of the image in pixels.

Parameters

<i>in</i>	<i>_index</i>	Index of the camera
-----------	---------------	---------------------

Returns

The image width in pixels.

See Also

MultiCameraSensor::GetCameraCount (p. 516)

10.94.3.7 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. 689).

10.94.3.8 virtual void gazebo::sensors::MultiCameraSensor::Init () [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 690).

10.94.3.9 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. 690).

10.94.3.10 virtual void gazebo::sensors::MultiCameraSensor::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. 690).

10.94.3.11 `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. 516)

10.94.3.12 `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. 691).

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

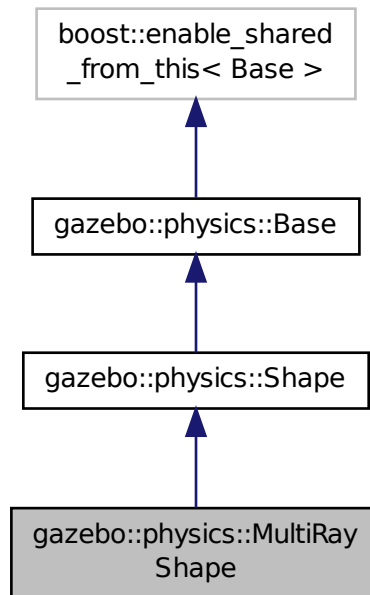
- **MultiCameraSensor.hh**

10.95 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.
- 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

10.95.1 Detailed Description

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

10.95.2 Constructor & Destructor Documentation

10.95.2.1 `gazebo::physics::MultiRayShape::MultiRayShape (CollisionPtr _parent) [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Parent collision shape.
----	----------------------	-------------------------

10.95.2.2 `virtual gazebo::physics::MultiRayShape::~~MultiRayShape () [virtual]`

Destructor.

10.95.3 Member Function Documentation

10.95.3.1 `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.

10.95.3.2 `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.

10.95.3.3 void gazebo::physics::MultiRayShape::DisconnectNewLaserScans (event::ConnectionPtr & _conn) [inline]

Disconnect from the new laser scans signal.

Parameters

in	_conn	Connection to remove.
----	-------	-----------------------

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

10.95.3.4 void gazebo::physics::MultiRayShape::FillMsg (msgs::Geometry & _msg) [virtual]

This function is not implemented.

Fill a message with this shape's values.

Parameters

out	_msg	Message that contains the shape's values.
-----	------	---

Implements gazebo::physics::Shape (p. 708).

10.95.3.5 int gazebo::physics::MultiRayShape::GetFiducial (int _index)

Get detected fiducial value for a ray.

Parameters

in	_index	Index of the ray.
----	--------	-------------------

Returns

Fiducial value for the ray.

10.95.3.6 math::Angle gazebo::physics::MultiRayShape::GetMaxAngle () const

Get the maximum angle.

Returns

Maximum angle of ray scan.

10.95.3.7 double gazebo::physics::MultiRayShape::GetMaxRange () const

Get the maximum range.

Returns

Maximum range of all the rays.

10.95.3.8 math::Angle gazebo::physics::MultiRayShape::GetMinAngle () const

Get the minimum angle.

Returns

Minimum angle of ray scan.

10.95.3.9 double gazebo::physics::MultiRayShape::GetMinRange () const

Get the minimum range.

Returns

Minimum range of all the rays.

10.95.3.10 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.

10.95.3.11 double gazebo::physics::MultiRayShape::GetResRange () const

Get the range resolution.

Returns

Range resolution of all the rays.

10.95.3.12 double gazebo::physics::MultiRayShape::GetRetro (int *_index*)

Get detected retro (intensity) value for a ray.

Parameters

<i>in</i>	<i>_index</i>	Index of the ray.
-----------	---------------	-------------------

Returns

Retro value for the ray.

10.95.3.13 `int gazebo::physics::MultiRayShape::GetSampleCount () const`

Get the horizontal sample count.

Returns

Horizontal sample count.

10.95.3.14 `double gazebo::physics::MultiRayShape::GetScanResolution () const`

Get the horizontal resolution.

Returns

Horizontal resolution.

10.95.3.15 `math::Angle gazebo::physics::MultiRayShape::GetVerticalMaxAngle () const`

Get the vertical max angle.

Returns

Vertical max angle.

10.95.3.16 `math::Angle gazebo::physics::MultiRayShape::GetVerticalMinAngle () const`

Get the vertical min angle.

Returns

Vertical min angle.

10.95.3.17 `int gazebo::physics::MultiRayShape::GetVerticalSampleCount () const`

Get the vertical sample count.

Returns

Vertical sample count.

10.95.3.18 `double gazebo::physics::MultiRayShape::GetVerticalScanResolution () const`

Get the vertical range resolution.

Returns

Vertical range resolution.

10.95.3.19 virtual void gazebo::physics::MultiRayShape::Init () [virtual]

Init the shape.

Implements **gazebo::physics::Shape** (p. 708).

10.95.3.20 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. 708).

10.95.3.21 void gazebo::physics::MultiRayShape::Update () [virtual]

Update the ray collisions.

Reimplemented from **gazebo::physics::Base** (p. 146).

10.95.3.22 virtual void gazebo::physics::MultiRayShape::UpdateRays () [protected],[pure virtual]

Physics engine specific method for updating the rays.

10.95.4 Member Data Documentation

10.95.4.1 sdf::ElementPtr gazebo::physics::MultiRayShape::horzElem [protected]

Horizontal SDF element pointer.

10.95.4.2 event::EventT<void()> gazebo::physics::MultiRayShape::newLaserScans [protected]

New laser scans event.

Referenced by ConnectNewLaserScans(), and DisconnectNewLaserScans().

10.95.4.3 math::Pose gazebo::physics::MultiRayShape::offset [protected]

Pose offset of all the rays.

10.95.4.4 sdf::ElementPtr gazebo::physics::MultiRayShape::rangeElem [protected]

Range SDF element pointer.

10.95.4.5 `sdf::ElementPtr gazebo::physics::MultiRayShape::rayElem` [protected]

Ray SDF element pointer.

10.95.4.6 `std::vector<RayShapePtr> gazebo::physics::MultiRayShape::rays` [protected]

Ray data.

10.95.4.7 `sdf::ElementPtr gazebo::physics::MultiRayShape::scanElem` [protected]

Scan SDF element pointer.

10.95.4.8 `sdf::ElementPtr gazebo::physics::MultiRayShape::vertElem` [protected]

Vertical SDF element pointer.

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

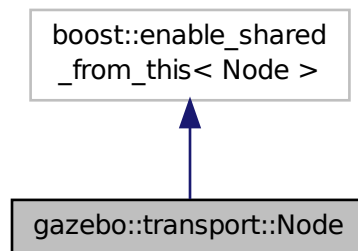
- **MultiRayShape.hh**

10.96 gazebo::transport::Node Class Reference

A (p. 109) 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 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 ProcessIncoming` ()
Process incoming messages.
- `void ProcessPublishers` ()
Process all publishers, which has each publisher send it's most recent message over the wire.
- `void RemoveCallback` (const std::string &_topic, unsigned int _id)
- `template<typename M , typename T >`
SubscriberPtr Subscribe (const std::string &_topic, void(T::*_fp)(const 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 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.

10.96.1 Detailed Description

A (p. 109) node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

10.96.2 Constructor & Destructor Documentation

10.96.2.1 gazebo::transport::Node::Node ()

Constructor.

10.96.2.2 virtual gazebo::transport::Node::~~Node () [virtual]

Destructor.

10.96.3 Member Function Documentation

10.96.3.1 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().

10.96.3.2 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().

10.96.3.3 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

10.96.3.4 void gazebo::transport::Node::Fini ()

Finalize the node.

10.96.3.5 unsigned int gazebo::transport::Node::GetId () const

Get the unique ID of the node.

Returns

The unique ID of the node

Referenced by Subscribe().

10.96.3.6 std::string gazebo::transport::Node::GetMsgType (const std::string & *_topic*) const

Get the message type for a topic.

Parameters

<i>in</i>	<i>_topic</i>	The topic
-----------	---------------	-----------

Returns

The message type

10.96.3.7 std::string gazebo::transport::Node::GetTopicNamespace () const

Get the topic namespace for this node.

Returns

The namespace

10.96.3.8 bool gazebo::transport::Node::HandleData (const std::string & *_topic*, const std::string & *_msg*)

Handle incoming data.

Parameters

<i>in</i>	<i>_topic</i>	Topic for which the data was received
<i>in</i>	<i>_msg</i>	The message that was received

Returns

true if the message was handled successfully, false otherwise

10.96.3.9 `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.

10.96.3.10 `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. 445)
-----------	---------------	---

10.96.3.11 `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.

10.96.3.12 `void gazebo::transport::Node::ProcessIncoming ()`

Process incoming messages.

10.96.3.13 `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

10.96.3.14 `void gazebo::transport::Node::RemoveCallback (const std::string & _topic, unsigned int _id)`

10.96.3.15 `template<typename M, typename T > SubscriberPtr gazebo::transport::Node::Subscribe (const std::string & _topic, void(T::*)(const 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. 761) object

References `DecodeTopicName()`, `GetId()`, `SingletonT< T >::Instance()`, and `gazebo::transport::Subscriber::Set-CallbackId()`.

10.96.3.16 `template<typename M > SubscriberPtr gazebo::transport::Node::Subscribe (const std::string & _topic, void(*)(const 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. 761) object

References `DecodeTopicName()`, `GetId()`, `SingletonT< T >::Instance()`, and `gazebo::transport::Subscriber::Set-CallbackId()`.

10.96.3.17 `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. 761) object

References DecodeTopicName(), GetId(), gazebo::transport::SubscribeOptions::Init(), SingletonT< T >::Instance(), and gazebo::transport::Subscriber::SetCallbackId().

10.96.3.18 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	<i>_topic</i>	The topic to subscribe to
in	<i>_fp</i>	Function to be called on receipt of new message
in	<i>_latching</i>	If true, latch latest incoming message; otherwise don't latch

Returns

Pointer to new **Subscriber** (p. 761) object

References DecodeTopicName(), GetId(), gazebo::transport::SubscribeOptions::Init(), SingletonT< T >::Instance(), and gazebo::transport::Subscriber::SetCallbackId().

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

- **Node.hh**

10.97 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

10.97.1 Detailed Description

Node animation.

10.97.2 Constructor & Destructor Documentation

10.97.2.1 gazebo::common::NodeAnimation::NodeAnimation (const std::string & *_name*)

constructor

Parameters

<i>in</i>	<i>_name</i>	the name of the node
-----------	--------------	----------------------

10.97.2.2 gazebo::common::NodeAnimation::~~NodeAnimation ()

Destructor. It empties the key frames list.

10.97.3 Member Function Documentation

10.97.3.1 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

10.97.3.2 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

10.97.3.3 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)

10.97.3.4 unsigned int gazebo::common::NodeAnimation::GetFrameCount () const

Returns the number of key frames.

Returns

the count

10.97.3.5 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

10.97.3.6 `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

<code>in</code>	<code>_i</code>	the index
-----------------	-----------------	-----------

Returns

a pair that contains the time and transformation. **Time** (p. 776) is -1 if the index is out of bounds

10.97.3.7 `double gazebo::common::NodeAnimation::GetLength () const`

Returns the duration of the animations.

Returns

the time of the last animation

10.97.3.8 `std::string gazebo::common::NodeAnimation::GetName () const`

Returns the name.

Returns

the name

10.97.3.9 `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

<code>in</code>	<code>_x</code>	the value along x. You must ensure that <code>_x</code> is within a valid range.
-----------------	-----------------	--

10.97.3.10 `void gazebo::common::NodeAnimation::Scale (const double _scale)`

Scales each transformation in the key frames.

This only affects the translational values.

Parameters

<code>in</code>	<code>_scale</code>	the scaling factor
-----------------	---------------------	--------------------

10.97.3.11 void gazebo::common::NodeAnimation::SetName (const std::string & *_name*)

Changes the name of the animation.

Parameters

<i>in</i>	<i>the</i>	new name
-----------	------------	----------

10.97.4 Member Data Documentation

10.97.4.1 std::map<double, math::Matrix4> gazebo::common::NodeAnimation::keyFrames [protected]

the dictionary of key frames, indexed by time

10.97.4.2 double gazebo::common::NodeAnimation::length [protected]

the duration of the animations (time of last key frame)

10.97.4.3 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**

10.98 gazebo::common::NodeAssignment Struct Reference

Vertex to node weighted assignement 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)

10.98.1 Detailed Description

Vertex to node weighted assignement for skeleton animation visualization.

10.98.2 Member Data Documentation

10.98.2.1 unsigned int gazebo::common::NodeAssignment::nodeIndex

node (or bone) index

10.98.2.2 unsigned int gazebo::common::NodeAssignment::vertexIndex

index of the vertex

10.98.2.3 float gazebo::common::NodeAssignment::weight

the weight (between 0 and 1)

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

- **Mesh.hh**

10.99 gazebo::common::NodeTransform Class Reference

NodeTransform (p. 537) **Skeleton.hh** (p. 1070) common/common.hh

```
#include <Skeleton.hh>
```

Public Types

- enum **TransformType** { **TRANSLATE**, **ROTATE**, **SCALE**, **MATRIX** }
Enumeration of the transform types.

Public Member Functions

- **NodeTransform** (**TransformType** _type=**MATRIX**)
Constructor.
- **NodeTransform** (**math::Matrix4** _mat, std::string _sid="_default_", **TransformType** _type=**MATRIX**)
Constructor.
- **~NodeTransform** ()
Destructor. It does nothing.
- **math::Matrix4** **Get** ()
Returns the transformation matrix.
- std::string **GetSID** ()
Returns thr SID.
- **TransformType** **GetType** ()
Returns the transformation type.
- **math::Matrix4** **operator**() ()
Matrix cast operator.
- **math::Matrix4** **operator*** (**NodeTransform** _t)
Node transform multiplication operator.

- **math::Matrix4 operator*** (**math::Matrix4** _m)
Matrix multiplication operator.
- void **PrintSource** ()
Prints the transform matrix to std::err stream.
- void **RecalculateMatrix** ()
Sets the transform matrix from the source according to the type.
- void **Set** (**math::Matrix4** _mat)
Assign a transformation.
- void **SetComponent** (unsigned int _idx, double _value)
Set a transformation matrix component value.
- void **SetSID** (std::string _sid)
Set the SID.
- void **SetSourceValues** (**math::Matrix4** _mat)
Set source data values _param[in] _mat the values.
- void **SetSourceValues** (**math::Vector3** _vec)
Set source data values.
- void **SetSourceValues** (**math::Vector3** _axis, double _angle)
Sets source matrix values from 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

10.99.1 Detailed Description

NodeTransform (p. 537) **Skeleton.hh** (p. 1070) common/common.hh

A (p. 109) transformation node

10.99.2 Member Enumeration Documentation

10.99.2.1 enum gazebo::common::NodeTransform::TransformType

Enumeration of the transform types.

Enumerator:

TRANSLATE
ROTATE
SCALE
MATRIX

10.99.3 Constructor & Destructor Documentation

10.99.3.1 gazebo::common::NodeTransform::NodeTransform (TransformType *_type* = MATRIX)

Constructor.

Parameters

in	<i>_type</i>	the type of transform
----	--------------	-----------------------

10.99.3.2 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

10.99.3.3 gazebo::common::NodeTransform::~~NodeTransform ()

Destructor. It does nothing.

10.99.4 Member Function Documentation

10.99.4.1 math::Matrix4 gazebo::common::NodeTransform::Get ()

Returns the transformation matrix.

Returns

the matrix

10.99.4.2 std::string gazebo::common::NodeTransform::GetSID ()

Returns thr SID.

Returns

the SID

10.99.4.3 TransformType gazebo::common::NodeTransform::GetType ()

Returns the transformation type.

Returns

the type

10.99.4.4 `math::Matrix4 gazebo::common::NodeTransform::operator() ()`

Matrix cast operator.

Returns

the transform

10.99.4.5 `math::Matrix4 gazebo::common::NodeTransform::operator* (NodeTransform _t)`

Node transform multiplication operator.

Parameters

in	_t	a transform
----	----	-------------

Returns

transform matrix multiplied by _t's transform

10.99.4.6 `math::Matrix4 gazebo::common::NodeTransform::operator* (math::Matrix4 _m)`

Matrix multiplication operator.

Parameters

in	_m	a matrix
----	----	----------

Returns

transform matrix multiplied by _m

10.99.4.7 `void gazebo::common::NodeTransform::PrintSource ()`

Prints the transform matrix to std::err stream.

10.99.4.8 `void gazebo::common::NodeTransform::RecalculateMatrix ()`

Sets the transform matrix from the source according to the type.

10.99.4.9 `void gazebo::common::NodeTransform::Set (math::Matrix4 _mat)`

Assign a transformation.

Parameters

in	_mat	the transform
----	------	---------------

10.99.4.10 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

10.99.4.11 void gazebo::common::NodeTransform::SetSID (std::string *_sid*)

Set the SID.

Parameters

in	<i>_sid</i>	the sid
----	-------------	---------

10.99.4.12 void gazebo::common::NodeTransform::SetSourceValues (math::Matrix4 *_mat*)

Set source data values *_param[in]* *_mat* the values.

10.99.4.13 void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 *_vec*)

Set source data values.

10.99.4.14 void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 *_axis*, double *_angle*)

Sets source matrix values from roation.

Parameters

in	<i>_axis</i>	of rotation
in	<i>_angle</i>	of rotation

10.99.4.15 void gazebo::common::NodeTransform::SetType (TransformType *_type*)

Set transform type.

Parameters

in	<i>_type</i>	the type
----	--------------	----------

10.99.5 Member Data Documentation

10.99.5.1 std::string gazebo::common::NodeTransform::sid [protected]

the sid

10.99.5.2 `std::vector<double>` gazebo::common::NodeTransform::source [protected]

source data values (can be a matrix, a position or rotation)

10.99.5.3 `math::Matrix4` gazebo::common::NodeTransform::transform [protected]

transform

10.99.5.4 `TransformType` gazebo::common::NodeTransform::type [protected]

transform type

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

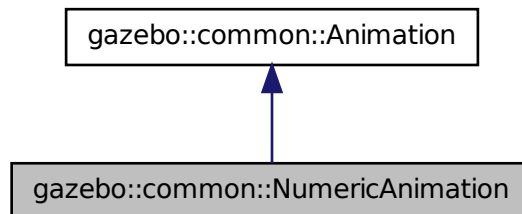
- **Skeleton.hh**

10.100 gazebo::common::NumericAnimation Class Reference

A (p. 109) 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

10.100.1 Detailed Description

A (p. 109) numeric animation.

10.100.2 Constructor & Destructor Documentation

10.100.2.1 gazebo::common::NumericAnimation::NumericAnimation (const std::string & *_name*, double *_length*, bool *_loop*)

Constructor.

Parameters

in	<i>_name</i>	String name of the animation. This should be unique.
in	<i>_length</i>	Length of the animation in seconds
in	<i>_loop</i>	True == loop the animation

10.100.2.2 virtual gazebo::common::NumericAnimation::~~NumericAnimation () [virtual]

Destructor.

10.100.3 Member Function Documentation

10.100.3.1 NumericKeyFrame* gazebo::common::NumericAnimation::CreateKeyFrame (double *_time*)

Create a numeric keyframe at the given time.

Parameters

in	<i>_time</i>	Time (p. 776) at which to create the keyframe
----	--------------	--

Returns

Pointer to the new keyframe

10.100.3.2 void gazebo::common::NumericAnimation::GetInterpolatedKeyFrame (NumericKeyFrame & *_kf*) const

Get a keyframe using the animation's current time.

Parameters

out	<i>_kf</i>	NumericKeyFrame (p. 544) reference to hold the interpolated result
-----	------------	---

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

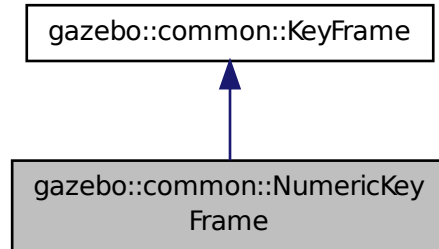
- **Animation.hh**

10.101 gazebo::common::NumericKeyFrame Class Reference

A (p. 109) keyframe for a **NumericAnimation** (p. 542).

```
#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

10.101.1 Detailed Description

A (p. 109) keyframe for a **NumericAnimation** (p. 542).

10.101.2 Constructor & Destructor Documentation

10.101.2.1 gazebo::common::NumericKeyFrame::NumericKeyFrame (double _time)

Constructor.

Parameters

in	<code>_time</code>	Time (p. 776) of the keyframe
----	--------------------	--------------------------------------

10.101.2.2 virtual gazebo::common::NumericKeyFrame::~~NumericKeyFrame () [virtual]

Destructor.

10.101.3 Member Function Documentation

10.101.3.1 const double& gazebo::common::NumericKeyFrame::GetValue () const

Get the value of the keyframe.

Returns

the value of the keyframe

10.101.3.2 void gazebo::common::NumericKeyFrame::SetValue (const double & *_value*)

Set the value of the keyframe.

Parameters

in	<code>_value</code>	The new value
----	---------------------	---------------

10.101.4 Member Data Documentation

10.101.4.1 double gazebo::common::NumericKeyFrame::value [protected]

numeric value

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

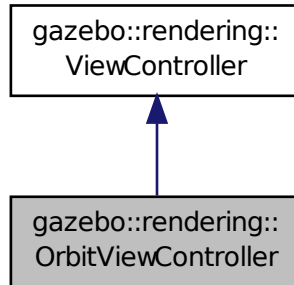
- **KeyFrame.hh**

10.102 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

10.102.1 Detailed Description

Orbit view controller.

10.102.2 Constructor & Destructor Documentation

10.102.2.1 gazebo::rendering::OrbitViewController::OrbitViewController (`UserCameraPtr _camera`)

Constructor.

Parameters

in	<code>_camera</code>	Pointer to the camera to control.
----	----------------------	-----------------------------------

10.102.2.2 virtual gazebo::rendering::OrbitViewController::~~OrbitViewController () [virtual]

Destructor.

10.102.3 Member Function Documentation

10.102.3.1 `math::Vector3` gazebo::rendering::OrbitViewController::GetFocalPoint () const

Get the focal point.

Returns

The focal point

10.102.3.2 `static std::string` gazebo::rendering::OrbitViewController::GetTypeString () [static]

Get the type name of this view controller.

Returns

The view controller name: "orbit".

10.102.3.3 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. 867).

10.102.3.4 `void gazebo::rendering::OrbitViewController::HandleKeyReleaseEvent (const std::string & _key) [virtual]`

Handle a key release event.

Parameters

<code>in</code>	<code><i>_key</i></code>	The key that was released.
-----------------	--------------------------	----------------------------

Implements **`gazebo::rendering::ViewController`** (p. 867).

10.102.3.5 `virtual void gazebo::rendering::OrbitViewController::HandleMouseEvent (const common::MouseEvent & _event) [virtual]`

Handle a mouse event.

Parameters

<code>in</code>	<code><i>_event</i></code>	The mouse event.
-----------------	----------------------------	------------------

Implements **`gazebo::rendering::ViewController`** (p. 867).

10.102.3.6 `virtual void gazebo::rendering::OrbitViewController::Init () [virtual]`

Initialize the controller.

Implements **`gazebo::rendering::ViewController`** (p. 867).

10.102.3.7 `virtual void gazebo::rendering::OrbitViewController::Init (const math::Vector3 & _focalPoint) [virtual]`

Initialize the controller with a focal point.

Parameters

<code>in</code>	<code><i>_focalPoint</i></code>	Point to look at.
-----------------	---------------------------------	-------------------

Reimplemented from **`gazebo::rendering::ViewController`** (p. 868).

10.102.3.8 `void gazebo::rendering::OrbitViewController::SetDistance (float _d)`

Set the distance to the focal point.

Parameters

<code>in</code>	<code><i>_d</i></code>	The distance from the focal point.
-----------------	------------------------	------------------------------------

10.102.3.9 `void gazebo::rendering::OrbitViewController::SetFocalPoint (const math::Vector3 & _fp)`

Set the focal point.

Parameters

in	<code>_fp</code>	The focal point
----	------------------	-----------------

10.102.3.10 `virtual void gazebo::rendering::OrbitViewController::Update () [virtual]`

Update.

Implements `gazebo::rendering::ViewController` (p. 868).

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

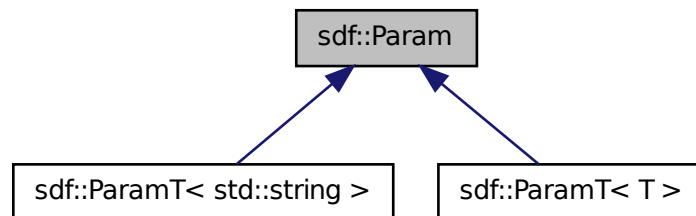
- `OrbitViewController.hh`

10.103 sdf::Param Class Reference

A (p. 109) parameter class.

```
#include <Param.hh>
```

Inheritance diagram for `sdf::Param`:



Public Member Functions

- **Param** (**Param** *`_newParam`)
Constructor.
- virtual **~Param** ()
Destructor.
- virtual **Param** * **Clone** () const =0
- bool **Get** (bool &`_value`)
- bool **Get** (int &`_value`)
- bool **Get** (unsigned int &`_value`)
- bool **Get** (float &`_value`)
- bool **Get** (double &`_value`)
- bool **Get** (char &`_value`)
- bool **Get** (std::string &`_value`)

- bool **Get** (gazebo::math::Vector3 &_value)
- bool **Get** (gazebo::math::Vector2i &_value)
- bool **Get** (gazebo::math::Vector2d &_value)
- bool **Get** (gazebo::math::Quaternion &_value)
- bool **Get** (gazebo::math::Pose &_value)
- bool **Get** (gazebo::common::Color &_value)
- bool **Get** (gazebo::common::Time &_value)
- virtual std::string **GetAsString** () const
Get the type.
- virtual std::string **GetDefaultAsString** () const
- std::string **GetDescription** () const
Get the description of the parameter.
- const std::string & **GetKey** () const
- bool **GetRequired** () const
- bool **GetSet** () const
Return true if the parameter has been set.
- std::string **GetType** () const
- bool **IsBool** () const
- bool **IsChar** () const
- bool **IsColor** () const
- bool **IsDouble** () const
- bool **IsFloat** () const
- bool **IsInt** () const
- bool **IsPose** () const
- bool **IsQuaternion** () const
- bool **IsStr** () const
- bool **IsTime** () const
- bool **IsUInt** () const
- bool **IsVector2d** () const
- bool **IsVector2i** () const
- bool **IsVector3** () const
- virtual void **Reset** ()=0
Reset the parameter.
- bool **Set** (const bool &_value)
- bool **Set** (const int &_value)
- bool **Set** (const unsigned int &_value)
- bool **Set** (const float &_value)
- bool **Set** (const double &_value)
- bool **Set** (const char &_value)
- bool **Set** (const std::string &_value)
- bool **Set** (const char *_value)
- bool **Set** (const gazebo::math::Vector3 &_value)
- bool **Set** (const gazebo::math::Vector2i &_value)
- bool **Set** (const gazebo::math::Vector2d &_value)
- bool **Set** (const gazebo::math::Quaternion &_value)
- bool **Set** (const gazebo::math::Pose &_value)
- bool **Set** (const gazebo::common::Color &_value)
- bool **Set** (const gazebo::common::Time &_value)
- void **SetDescription** (const std::string &_desc)
Set the description of the parameter.

- virtual bool **SetFromString** (const std::string &)
Set the parameter value from a string.
- template<typename T >
void **SetUpdateFunc** (T _updateFunc)
Update function.
- virtual void **Update** ()=0

Protected Attributes

- std::string **description**
- std::string **key**
- bool **required**
- bool **set**
- std::string **typeName**
- boost::function< boost::any()> **updateFunc**

10.103.1 Detailed Description

A (p. 109) parameter class.

10.103.2 Constructor & Destructor Documentation

10.103.2.1 `sdf::Param::Param (Param * _newParam)`

Constructor.

10.103.2.2 `virtual sdf::Param::~~Param () [virtual]`

Destructor.

10.103.3 Member Function Documentation

10.103.3.1 `virtual Param* sdf::Param::Clone () const [pure virtual]`

Implemented in `sdf::ParamT< T >` (p. 557), and `sdf::ParamT< std::string >` (p. 557).

10.103.3.2 `bool sdf::Param::Get (bool & _value)`

10.103.3.3 `bool sdf::Param::Get (int & _value)`

10.103.3.4 `bool sdf::Param::Get (unsigned int & _value)`

10.103.3.5 `bool sdf::Param::Get (float & _value)`

10.103.3.6 `bool sdf::Param::Get (double & _value)`

10.103.3.7 `bool sdf::Param::Get (char & _value)`

10.103.3.8 `bool sdf::Param::Get (std::string & _value)`

10.103.3.9 `bool sdf::Param::Get (gazebo::math::Vector3 & _value)`

10.103.3.10 `bool sdf::Param::Get (gazebo::math::Vector2i & _value)`

10.103.3.11 `bool sdf::Param::Get (gazebo::math::Vector2d & _value)`

10.103.3.12 `bool sdf::Param::Get (gazebo::math::Quaternion & _value)`

10.103.3.13 `bool sdf::Param::Get (gazebo::math::Pose & _value)`

10.103.3.14 `bool sdf::Param::Get (gazebo::common::Color & _value)`

10.103.3.15 `bool sdf::Param::Get (gazebo::common::Time & _value)`

10.103.3.16 `virtual std::string sdf::Param::GetAsString () const [inline],[virtual]`

Get the type.

Reimplemented in `sdf::ParamT< T >` (p. 557), and `sdf::ParamT< std::string >` (p. 557).

10.103.3.17 `virtual std::string sdf::Param::GetDefaultAsString () const [inline],[virtual]`

Reimplemented in `sdf::ParamT< T >` (p. 557), and `sdf::ParamT< std::string >` (p. 557).

10.103.3.18 `std::string sdf::Param::GetDescription () const`

Get the description of the parameter.

10.103.3.19 `const std::string& sdf::Param::GetKey () const [inline]`

References key.

Referenced by `sdf::ParamT< std::string >::Clone()`, and `sdf::ParamT< std::string >::Set()`.

10.103.3.20 `bool sdf::Param::GetRequired () const [inline]`

References required.

10.103.3.21 `bool sdf::Param::GetSet () const [inline]`

Return true if the parameter has been set.

10.103.3.22 `std::string sdf::Param::GetTypeNames () const`

10.103.3.23 `bool sdf::Param::IsBool () const`

10.103.3.24 `bool sdf::Param::IsChar () const`

- 10.103.3.25 `bool sdf::Param::IsColor () const`
- 10.103.3.26 `bool sdf::Param::IsDouble () const`
- 10.103.3.27 `bool sdf::Param::IsFloat () const`
- 10.103.3.28 `bool sdf::Param::IsInt () const`
- 10.103.3.29 `bool sdf::Param::IsPose () const`
- 10.103.3.30 `bool sdf::Param::IsQuaternion () const`
- 10.103.3.31 `bool sdf::Param::IsStr () const`
- 10.103.3.32 `bool sdf::Param::IsTime () const`
- 10.103.3.33 `bool sdf::Param::IsUInt () const`
- 10.103.3.34 `bool sdf::Param::IsVector2d () const`
- 10.103.3.35 `bool sdf::Param::IsVector2i () const`
- 10.103.3.36 `bool sdf::Param::IsVector3 () const`
- 10.103.3.37 `virtual void sdf::Param::Reset () [pure virtual]`

Reset the parameter.

Implemented in `sdf::ParamT< T >` (p. 557), and `sdf::ParamT< std::string >` (p. 557).

- 10.103.3.38 `bool sdf::Param::Set (const bool & _value)`

Referenced by `sdf::ParamT< std::string >::Update()`.

- 10.103.3.39 `bool sdf::Param::Set (const int & _value)`
- 10.103.3.40 `bool sdf::Param::Set (const unsigned int & _value)`
- 10.103.3.41 `bool sdf::Param::Set (const float & _value)`
- 10.103.3.42 `bool sdf::Param::Set (const double & _value)`
- 10.103.3.43 `bool sdf::Param::Set (const char & _value)`
- 10.103.3.44 `bool sdf::Param::Set (const std::string & _value)`
- 10.103.3.45 `bool sdf::Param::Set (const char * _value)`
- 10.103.3.46 `bool sdf::Param::Set (const gazebo::math::Vector3 & _value)`
- 10.103.3.47 `bool sdf::Param::Set (const gazebo::math::Vector2i & _value)`

10.103.3.48 `bool sdf::Param::Set (const gazebo::math::Vector2d & _value)`

10.103.3.49 `bool sdf::Param::Set (const gazebo::math::Quaternion & _value)`

10.103.3.50 `bool sdf::Param::Set (const gazebo::math::Pose & _value)`

10.103.3.51 `bool sdf::Param::Set (const gazebo::common::Color & _value)`

10.103.3.52 `bool sdf::Param::Set (const gazebo::common::Time & _value)`

10.103.3.53 `void sdf::Param::SetDescription (const std::string & _desc)`

Set the description of the parameter.

10.103.3.54 `virtual bool sdf::Param::SetFromString (const std::string &) [inline],[virtual]`

Set the parameter value from a string.

Reimplemented in `sdf::ParamT< T >` (p. 557), and `sdf::ParamT< std::string >` (p. 557).

10.103.3.55 `template<typename T> void sdf::Param::SetUpdateFunc (T _updateFunc) [inline]`

Update function.

References updateFunc.

10.103.3.56 `virtual void sdf::Param::Update () [pure virtual]`

Implemented in `sdf::ParamT< T >` (p. 558), and `sdf::ParamT< std::string >` (p. 558).

10.103.4 Member Data Documentation

10.103.4.1 `std::string sdf::Param::description [protected]`

Referenced by `sdf::ParamT< std::string >::Clone()`, and `sdf::ParamT< std::string >::ParamT()`.

10.103.4.2 `std::string sdf::Param::key [protected]`

Referenced by `GetKey()`, `sdf::ParamT< std::string >::ParamT()`, and `sdf::ParamT< std::string >::Set()`.

10.103.4.3 `bool sdf::Param::required [protected]`

Referenced by `sdf::ParamT< std::string >::Clone()`, `GetRequired()`, `sdf::ParamT< std::string >::ParamT()`, and `sdf::ParamT< std::string >::Set()`.

10.103.4.4 `bool sdf::Param::set [protected]`

10.103.4.5 `std::string sdf::Param::typeName` [protected]

Referenced by `sdf::ParamT< std::string >::Clone()`, and `sdf::ParamT< std::string >::ParamT()`.

10.103.4.6 `boost::function<boost::any ()> sdf::Param::updateFunc` [protected]

Referenced by `SetUpUpdateFunc()`, and `sdf::ParamT< std::string >::Update()`.

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

- **Param.hh**

10.104 ParamT< T > Class Template Reference

```
#include <CommonTypes.hh>
```

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

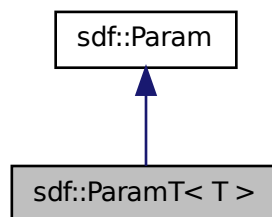
- **CommonTypes.hh**

10.105 sdf::ParamT< T > Class Template Reference

Templatized parameter class.

```
#include <Param.hh>
```

Inheritance diagram for `sdf::ParamT< T >`:



Public Member Functions

- **ParamT** (`const std::string &_key`, `const std::string &_default`, `bool _required`, `const std::string &_typeName=""`, `const std::string &_description=""`)

Constructor.

- `virtual ~ParamT ()`

Destructor.

- virtual **Param** * **Clone** () const
- virtual std::string **GetAsString** () const
Get the parameter value as a string.
- virtual std::string **GetDefaultAsString** () const
- T **GetDefaultValue** () const
Get the value.
- T **GetValue** () const
Get the value.
- T **operator*** () const
- virtual void **Reset** ()
Reset to default value.
- virtual bool **Set** (const std::string &_str)
Set the parameter value from a string.
- virtual bool **SetFromString** (const std::string &_value)
Set the parameter value from a string.
- void **SetValue** (const T &_value)
Set the value of the parameter.
- virtual void **Update** ()
Update param value.

Protected Attributes

- T **defaultValue**
- T **value**

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **ParamT**< T > &_p)

10.105.1 Detailed Description

template<typename T>class sdf::ParamT< T >

Templatized parameter class.

10.105.2 Constructor & Destructor Documentation

10.105.2.1 template<typename T> sdf::ParamT< T >::ParamT (const std::string & _key, const std::string & _default, bool _required, const std::string & _typeName = "", const std::string & _description = "") [inline]

Constructor.

10.105.2.2 template<typename T> virtual sdf::ParamT< T >::~~ParamT () [inline],[virtual]

Destructor.

10.105.3 Member Function Documentation

10.105.3.1 `template<typename T> virtual Param* sdf::ParamT< T >::Clone () const` [inline],[virtual]

Implements **sdf::Param** (p. 551).

10.105.3.2 `template<typename T> virtual std::string sdf::ParamT< T >::GetAsString () const` [inline],[virtual]

Get the parameter value as a string.

Reimplemented from **sdf::Param** (p. 552).

Referenced by `sdf::ParamT< std::string >::Clone()`.

10.105.3.3 `template<typename T> virtual std::string sdf::ParamT< T >::GetDefaultAsString () const` [inline],[virtual]

Reimplemented from **sdf::Param** (p. 552).

10.105.3.4 `template<typename T> T sdf::ParamT< T >::GetDefaultValue () const` [inline]

Get the value.

10.105.3.5 `template<typename T> T sdf::ParamT< T >::GetValue () const` [inline]

Get the value.

10.105.3.6 `template<typename T> T sdf::ParamT< T >::operator*() const` [inline]

10.105.3.7 `template<typename T> virtual void sdf::ParamT< T >::Reset ()` [inline],[virtual]

Reset to default value.

Implements **sdf::Param** (p. 553).

10.105.3.8 `template<typename T> virtual bool sdf::ParamT< T >::Set (const std::string & _str)` [inline],[virtual]

Set the parameter value from a string.

Referenced by `sdf::ParamT< std::string >::ParamT()`, and `sdf::ParamT< std::string >::SetFromString()`.

10.105.3.9 `template<typename T> virtual bool sdf::ParamT< T >::SetFromString (const std::string & _value)` [inline],[virtual]

Set the parameter value from a string.

Reimplemented from **sdf::Param** (p. 554).

10.105.3.10 `template<typename T> void sdf::ParamT< T >::SetValue (const T & _value) [inline]`

Set the value of the parameter.

10.105.3.11 `template<typename T> virtual void sdf::ParamT< T >::Update () [inline],[virtual]`

Update param value.

Implements `sdf::Param` (p. 554).

10.105.4 Friends And Related Function Documentation

10.105.4.1 `template<typename T> std::ostream& operator<< (std::ostream & _out, const ParamT< T > & _p) [friend]`

10.105.5 Member Data Documentation

10.105.5.1 `template<typename T> T sdf::ParamT< T >::defaultValue [protected]`

Referenced by `sdf::ParamT< std::string >::GetDefaultAsString()`, `sdf::ParamT< std::string >::GetDefaultValue()`, `sdf::ParamT< std::string >::ParamT()`, `sdf::ParamT< std::string >::Reset()`, and `sdf::ParamT< std::string >::Set()`.

10.105.5.2 `template<typename T> T sdf::ParamT< T >::value [protected]`

Referenced by `sdf::ParamT< std::string >::GetAsString()`, `sdf::ParamT< std::string >::GetValue()`, `sdf::ParamT< std::string >::operator*()`, `sdf::ParamT< std::string >::ParamT()`, `sdf::ParamT< std::string >::Reset()`, `sdf::ParamT< std::string >::Set()`, and `sdf::ParamT< std::string >::SetValue()`.

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

- **Param.hh**

10.106 gazebo::physics::PhysicsEngine Class Reference

Base (p. 135) class for a physics engine.

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

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)=0
Create a new joint.

- virtual **LinkPtr CreateLink** (**ModelPtr** _parent)=0
Create a new body.
- virtual **ShapePtr CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)=0
*Create a **physics::Shape** (p. 706) 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.
- boost::recursive_mutex * **GetPhysicsUpdateMutex** () const
*returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 569).*
- 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.
- virtual double **GetStepTime** ()=0
Get the simulation step time.
- virtual std::string **GetType** () const =0
Return the type of the physics engine (ode|bullet).
- double **GetUpdatePeriod** ()
Get the simulation update period.
- double **GetUpdateRate** ()
Get the simulation update rate.
- 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 gravity vector.
 - virtual void **SetMaxContacts** (double _maxContacts)
 - : Remove this function, and replace it with a more generic property map
 - 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
 - virtual void **SetStepTime** (double _value)=0
 - Set the simulation step time.
 - void **SetUpdateRate** (double _value)
 - Set the simulation update rate.
 - 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.
- **transport::NodePtr node**
 - Node for communication.
- **transport::SubscriberPtr physicsSub**

Subscribe to the physics topic.

- boost::recursive_mutex * **physicsUpdateMutex**

Mutex to protect the update cycle.

- transport::SubscriberPtr **requestSub**

Subscribe to the request topic.

- transport::PublisherPtr **responsePub**

Response publisher.

- sdf::ElementPtr **sdf**

Our SDF values.

- WorldPtr **world**

Pointer to the world.

10.106.1 Detailed Description

Base (p. 135) class for a physics engine.

10.106.2 Constructor & Destructor Documentation

10.106.2.1 gazebo::physics::PhysicsEngine::PhysicsEngine (WorldPtr *world*) [explicit]

Default constructor.

Parameters

in	<i>world</i>	Pointer to the world.
----	--------------	-----------------------

10.106.2.2 virtual gazebo::physics::PhysicsEngine::~PhysicsEngine () [virtual]

Destructor.

10.106.3 Member Function Documentation

10.106.3.1 virtual CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision (const std::string & *shapeType*, LinkPtr *link*) [pure virtual]

Create a collision.

Parameters

in	<i>shapeType</i>	Type of collision to create.
in	<i>link</i>	Parent link.

10.106.3.2 CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision (const std::string & *shapeType*, const std::string & *linkName*)

Create a collision.

Parameters

in	<code>_shapeType</code>	Type of collision to create.
in	<code>_linkName</code>	Name of the parent link.

10.106.3.3 `virtual JointPtr gazebo::physics::PhysicsEngine::CreateJoint (const std::string & _type, ModelPtr _parent)`
`[pure virtual]`

Create a new joint.

Parameters

in	<code>_type</code>	Type of joint to create.
in	<code>_parent</code>	Model (p. 480) parent.

10.106.3.4 `virtual LinkPtr gazebo::physics::PhysicsEngine::CreateLink (ModelPtr _parent)` `[pure virtual]`

Create a new body.

Parameters

in	<code>_parent</code>	Parent model for the link.
----	----------------------	----------------------------

10.106.3.5 `virtual ShapePtr gazebo::physics::PhysicsEngine::CreateShape (const std::string & _shapeType, CollisionPtr`
`_collision)` `[pure virtual]`

Create a **physics::Shape** (p. 706) object.

Parameters

in	<code>_shapeType</code>	Type of shape to create.
in	<code>_collision</code>	Collision (p. 192) parent.

10.106.3.6 `virtual void gazebo::physics::PhysicsEngine::DebugPrint () const` `[pure virtual]`

Debug print out of the physic engine state.

10.106.3.7 `virtual void gazebo::physics::PhysicsEngine::Fini ()` `[virtual]`

Finilize the physics engine.

10.106.3.8 `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.

10.106.3.9 ContactManager* gazebo::physics::PhysicsEngine::GetContactManager () const

Get a pointer to the contact manger.

Returns

Pointer to the contact manager.

10.106.3.10 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.

10.106.3.11 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. 232) suerface layer depth.

10.106.3.12 virtual math::Vector3 gazebo::physics::PhysicsEngine::GetGravity () const [virtual]

Return the gavity vector.

Returns

The gavity vector.

10.106.3.13 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.

10.106.3.14 `boost::recursive_mutex* gazebo::physics::PhysicsEngine::GetPhysicsUpdateMutex () const [inline]`

returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 569).

Returns

Pointer to the physics mutex.

References physicsUpdateMutex.

10.106.3.15 `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.

10.106.3.16 `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.

10.106.3.17 `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.

10.106.3.18 `virtual double gazebo::physics::PhysicsEngine::GetStepTime () [pure virtual]`

Get the simulation step time.

Returns

Simulation step time.

10.106.3.19 `virtual std::string gazebo::physics::PhysicsEngine::GetType () const [pure virtual]`

Return the type of the physics engine (ode|bullet).

Returns

Type of the physics engine.

10.106.3.20 `double gazebo::physics::PhysicsEngine::GetUpdatePeriod ()`

Get the simulation update period.

Returns

Simulation update period.

10.106.3.21 `double gazebo::physics::PhysicsEngine::GetUpdateRate ()`

Get the simulation update rate.

Returns

Update rate.

10.106.3.22 `virtual double gazebo::physics::PhysicsEngine::GetWorldCFM () [inline],[virtual]`

: Remove this function, and replace it with a more generic property map

Get **World** (p. 893) CFM.

Returns

World (p. 893) CFM.

10.106.3.23 `virtual double gazebo::physics::PhysicsEngine::GetWorldERP () [inline],[virtual]`

: Remove this function, and replace it with a more generic property map

Get **World** (p. 893) ERP.

Returns

World (p. 893) ERP.

10.106.3.24 `virtual void gazebo::physics::PhysicsEngine::Init () [pure virtual]`

Initialize the physics engine.

10.106.3.25 `virtual void gazebo::physics::PhysicsEngine::InitForThread () [pure virtual]`

Init the engine for threads.

10.106.3.26 `virtual void gazebo::physics::PhysicsEngine::Load (sdf::ElementPtr _sdf) [virtual]`

Load the physics engine.

Parameters

in	<code>_sdf</code>	Pointer to the SDF parameters.
----	-------------------	--------------------------------

10.106.3.27 `virtual void gazebo::physics::PhysicsEngine::OnPhysicsMsg (ConstPhysicsPtr & _msg) [protected], [virtual]`

virtual callback for gztopic "~/physics".

Parameters

in	<code>_msg</code>	Physics message.
----	-------------------	------------------

10.106.3.28 `virtual void gazebo::physics::PhysicsEngine::OnRequest (ConstRequestPtr & _msg) [protected], [virtual]`

virtual callback for gztopic "~/request".

Parameters

in	<code>_msg</code>	Request message.
----	-------------------	------------------

10.106.3.29 `virtual void gazebo::physics::PhysicsEngine::Reset () [inline], [virtual]`

Rest the physics engine.

10.106.3.30 `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.
----	---------------------------	--

10.106.3.31 `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.
----	-------------------	--------------------------

10.106.3.32 `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	<code>_layerDepth</code>	Surface layer depth
----	--------------------------	---------------------

10.106.3.33 `virtual void gazebo::physics::PhysicsEngine::SetGravity (const gazebo::math::Vector3 & _gravity) [pure virtual]`

Set the gravity vector.

Parameters

in	<code>_gravity</code>	New gravity vector.
----	-----------------------	---------------------

10.106.3.34 `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	<code>_maxContacts</code>	Maximum number of contacts.
----	---------------------------	-----------------------------

10.106.3.35 `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.
----	--------------------	-------------------------

10.106.3.36 `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.
----	--------------------	-----------------------

10.106.3.37 `virtual void gazebo::physics::PhysicsEngine::SetSORPGSPreconlters (unsigned int _iters) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

<code>in</code>	<code><i>_iter</i></code>	Number of iterations.
-----------------	---------------------------	-----------------------

10.106.3.38 `virtual void gazebo::physics::PhysicsEngine::SetSORPGSW (double _w) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

<code>in</code>	<code><i>_w</i></code>	SORPGSW value.
-----------------	------------------------	----------------

10.106.3.39 `virtual void gazebo::physics::PhysicsEngine::SetStepTime (double _value) [pure virtual]`

Set the simulation step time.

Parameters

<code>in</code>	<code><i>_value</i></code>	Value of the step time.
-----------------	----------------------------	-------------------------

10.106.3.40 `void gazebo::physics::PhysicsEngine::SetUpdateRate (double _value)`

Set the simulation update rate.

Parameters

<code>in</code>	<code><i>_value</i></code>	Value of the update rate.
-----------------	----------------------------	---------------------------

10.106.3.41 `virtual void gazebo::physics::PhysicsEngine::SetWorldCFM (double _cfm) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

<code>in</code>	<code><i>_cfm</i></code>	Constraint force mixing.
-----------------	--------------------------	--------------------------

10.106.3.42 `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

<code>in</code>	<code>_erp</code>	Error reduction parameter.
-----------------	-------------------	----------------------------

10.106.3.43 `virtual void gazebo::physics::PhysicsEngine::UpdateCollision ()` [pure virtual]

Update the physics engine collision.

10.106.3.44 `virtual void gazebo::physics::PhysicsEngine::UpdatePhysics ()` [inline],[virtual]

Update the physics engine.

10.106.4 Member Data Documentation

10.106.4.1 `ContactManager* gazebo::physics::PhysicsEngine::contactManager` [protected]

Class that handles all contacts generated by the physics engine.

10.106.4.2 `transport::NodePtr gazebo::physics::PhysicsEngine::node` [protected]

Node for communication.

10.106.4.3 `transport::SubscriberPtr gazebo::physics::PhysicsEngine::physicsSub` [protected]

Subscribe to the physics topic.

10.106.4.4 `boost::recursive_mutex* gazebo::physics::PhysicsEngine::physicsUpdateMutex` [protected]

Mutex to protect the update cycle.

Referenced by `GetPhysicsUpdateMutex()`.

10.106.4.5 `transport::SubscriberPtr gazebo::physics::PhysicsEngine::requestSub` [protected]

Subscribe to the request topic.

10.106.4.6 `transport::PublisherPtr gazebo::physics::PhysicsEngine::responsePub` [protected]

Response publisher.

10.106.4.7 `sdf::ElementPtr gazebo::physics::PhysicsEngine::sdf` [protected]

Our SDF values.

10.106.4.8 **WorldPtr** gazebo::physics::PhysicsEngine::world [protected]

Pointer to the world.

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

- **PhysicsEngine.hh**

10.107 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.

10.107.1 Detailed Description

The physics factory instantiates different physics engines.

10.107.2 Member Function Documentation

10.107.2.1 static bool gazebo::physics::PhysicsFactory::IsRegistered (const std::string *_name*) [static]

Check if a physics engine is registered.

Parameters

in	<i>_name</i>	Name of the physics engine.
----	--------------	-----------------------------

Returns

True if physics engine is registered, false otherwise.

10.107.2.2 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. 893) to pass to the created physics engine.

10.107.2.3 `static void gazebo::physics::PhysicsFactory::RegisterAll () [static]`

Register everything.

10.107.2.4 `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**

10.108 gazebo::common::PID Class Reference

Generic **PID** (p. 571) 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. 571) controller.*
- void **GetErrors** (double &_pe, double &_ie, double &_de)
*Return **PID** (p. 571) 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. 571) 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 derivtive 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.*

10.108.1 Detailed Description

Generic **PID** (p. 571) 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.

10.108.2 Constructor & Destructor Documentation

10.108.2.1 `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.

10.108.2.2 `virtual gazebo::common::PID::~~PID () [virtual]`

Destructor.

10.108.3 Member Function Documentation

10.108.3.1 double gazebo::common::PID::GetCmd ()

Return current command for this **PID** (p. 571) controller.

Returns

the command value

10.108.3.2 void gazebo::common::PID::GetErrors (double & _pe, double & _ie, double & _de)

Return **PID** (p. 571) 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.

10.108.3.3 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.

10.108.3.4 PID& gazebo::common::PID::operator= (const PID & _p) [inline]

Assignment operator.

Parameters

in	<code>_p</code>	a reference to a PID (p. 571) to assign values from
----	-----------------	--

Returns

reference to this instance

References Reset().

10.108.3.5 void gazebo::common::PID::Reset ()

Reset the errors and command.

Referenced by operator=().

10.108.3.6 void gazebo::common::PID::SetCmd (double *_cmd*)

Set current target command for this **PID** (p. 571) controller.

Parameters

in	<i>_cmd</i>	New command
----	-------------	-------------

10.108.3.7 void gazebo::common::PID::SetCmdMax (double *_c*)

Set the maximum value for the command.

Parameters

in	<i>_c</i>	The maximum value
----	-----------	-------------------

10.108.3.8 void gazebo::common::PID::SetCmdMin (double *_c*)

Set the maximum value for the command.

Parameters

in	<i>_c</i>	The maximum value
----	-----------	-------------------

10.108.3.9 void gazebo::common::PID::SetDGain (double *_d*)

Set the derivative Gain.

Parameters

in	<i>_p</i>	derivative gain value
----	-----------	-----------------------

10.108.3.10 void gazebo::common::PID::SetIGain (double *_i*)

Set the integral Gain.

Parameters

in	<i>_p</i>	integral gain value
----	-----------	---------------------

10.108.3.11 void gazebo::common::PID::SetIMax (double *_i*)

Set the integral upper limit.

Parameters

in	<i>_p</i>	integral upper limit value
----	-----------	----------------------------

10.108.3.12 `void gazebo::common::PID::SetIMin (double i)`

Set the integral lower limit.

Parameters

<code>in</code>	<code>_p</code>	integral lower limit value
-----------------	-----------------	----------------------------

10.108.3.13 `void gazebo::common::PID::SetPGain (double p)`

Set the proportional Gain.

Parameters

<code>in</code>	<code>_p</code>	proportional gain value
-----------------	-----------------	-------------------------

10.108.3.14 `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 (<code>p_state - p_target</code>).
<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**

10.109 gazebo::math::Plane Class Reference

A (p. 109) 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. 575) offset.

- **Vector3** **normal**

Plane (p. 575) normal.

- **Vector2d** **size**

Plane (p. 575) size.

10.109.1 Detailed Description

A (p. 109) plane and related functions.

10.109.2 Constructor & Destructor Documentation

10.109.2.1 gazebo::math::Plane::Plane ()

Constructor.

10.109.2.2 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

10.109.2.3 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

10.109.2.4 `virtual gazebo::math::Plane::~~Plane () [virtual]`

Destructor.

10.109.3 Member Function Documentation

10.109.3.1 `double gazebo::math::Plane::Distance (const Vector3 & _origin, const Vector3 & _dir) const`

Get distance to the plane give an origin and direction.

Parameters

<code>in</code>	<code>_origin</code>	the origin
<code>in</code>	<code>_dir</code>	a direction

Returns

the shortest distance

10.109.3.2 `Plane& gazebo::math::Plane::operator= (const Plane & _p)`

Equal operator.

Parameters

<code>_p</code>	another plane
-----------------	---------------

Returns

itself

10.109.3.3 `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

10.109.4 Member Data Documentation

10.109.4.1 `double gazebo::math::Plane::d`

Plane (p. 575) offset.

10.109.4.2 Vector3 gazebo::math::Plane::normal

Plane (p. 575) normal.

10.109.4.3 Vector2d gazebo::math::Plane::size

Plane (p. 575) size.

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

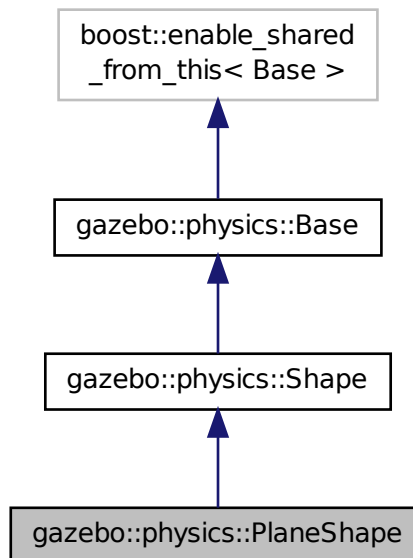
- **Plane.hh**

10.110 gazebo::physics::PlaneShape Class Reference

Collision (p. 192) 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.
- void **SetSize** (const **math::Vector2d** &_size)
Set the size.

Additional Inherited Members

10.110.1 Detailed Description

Collision (p. 192) 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.

10.110.2 Constructor & Destructor Documentation

10.110.2.1 gazebo::physics::PlaneShape::PlaneShape (**CollisionPtr** *_parent*) [explicit]

Constructor.

Parameters

in	_parent	Link (p. 412) to which we are attached.
----	---------	---

10.110.2.2 virtual gazebo::physics::PlaneShape::~~PlaneShape () [virtual]

Destructor.

10.110.3 Member Function Documentation

10.110.3.1 virtual void gazebo::physics::PlaneShape::CreatePlane () [virtual]

Create the plane.

10.110.3.2 `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. 708).

10.110.3.3 `math::Vector3 gazebo::physics::PlaneShape::GetNormal () const`

Get the plane normal.

Returns

The plane normal.

10.110.3.4 `math::Vector2d gazebo::physics::PlaneShape::GetSize () const`

Get the size.

Returns

Size of the plane.

10.110.3.5 `virtual void gazebo::physics::PlaneShape::Init () [virtual]`

Initialize the plane.

Implements `gazebo::physics::Shape` (p. 708).

10.110.3.6 `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	<code>_msg</code>	Message to update from.
----	-------------------	-------------------------

Implements `gazebo::physics::Shape` (p. 708).

10.110.3.7 `virtual void gazebo::physics::PlaneShape::SetAltitude (const math::Vector3 & _pos) [virtual]`

Set the altitude of the plane.

Parameters

in	<code>_pos</code>	Position of the plane.
----	-------------------	------------------------

10.110.3.8 void gazebo::physics::PlaneShape::SetNormal (const math::Vector3 & *_norm*)

Set the normal.

Parameters

in	<i>_norm</i>	Plane normal.
----	--------------	---------------

10.110.3.9 void gazebo::physics::PlaneShape::SetSize (const math::Vector2d & *_size*)

Set the size.

Parameters

in	<i>_size</i>	2D size of the plane.
----	--------------	-----------------------

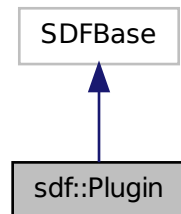
The documentation for this class was generated from the following file:

- **PlaneShape.hh**

10.111 sdf::Plugin Class Reference

```
#include <Plugin.hh>
```

Inheritance diagram for sdf::Plugin:



Public Member Functions

- **Plugin** ()
- void **Clear** ()
- void **Print** (const std::string &prefix)

Public Attributes

- std::vector< **ParamT**
< std::string > > **data**

- **ParamT**< std::string > **filename**
- **ParamT**< std::string > **name**

10.111.1 Constructor & Destructor Documentation

10.111.1.1 `sdf::Plugin::Plugin ()` [inline]

10.111.2 Member Function Documentation

10.111.2.1 `void sdf::Plugin::Clear ()` [inline]

References data.

10.111.2.2 `void sdf::Plugin::Print (const std::string & prefix)` [inline]

References filename, and name.

10.111.3 Member Data Documentation

10.111.3.1 `std::vector<ParamT<std::string>> sdf::Plugin::data`

Referenced by Clear().

10.111.3.2 `ParamT<std::string> sdf::Plugin::filename`

Referenced by Print().

10.111.3.3 `ParamT<std::string> sdf::Plugin::name`

Referenced by Print().

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

- **sdf/interface/Plugin.hh**

10.112 gazebo::PluginT< T > Class Template Reference

A (p. 109) class which all plugins must inherit from.

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

Public Types

- typedef T * **TPtr**
plugin pointer type definition

Public Member Functions

- `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.

10.112.1 Detailed Description

`template<class T>class gazebo::PluginT< T >`

A (p. 109) class which all plugins must inherit from.

10.112.2 Member Typedef Documentation

10.112.2.1 `template<class T> typedef T* gazebo::PluginT< T >::TPtr`

plugin pointer type definition

10.112.3 Member Function Documentation

10.112.3.1 `template<class T> static TPtr gazebo::PluginT< T >::Create (const std::string & _filename, const std::string & _handle) [inline],[static]`

a class method that creates a plugin from a file name.

It locates the shared library and loads it dynamically.

Parameters

in	<code>_filename</code>	the path to the shared library.
in	<code>_handle</code>	short name of the handler

Returns

Shared Pointer to this class type

10.112.3.2 `template<class T> std::string gazebo::PluginT< T >::GetFilename () const` [inline]

Get the name of the handler.

10.112.3.3 `template<class T> std::string gazebo::PluginT< T >::GetHandle () const` [inline]

Get the short name of the handler.

10.112.3.4 `template<class T> PluginType gazebo::PluginT< T >::GetType () const` [inline]

Returns the type of the plugin.

Returns

type of the plugin

10.112.4 Member Data Documentation

10.112.4.1 `template<class T> std::string gazebo::PluginT< T >::filename` [protected]

Path to the shared library file.

Referenced by `gazebo::PluginT< ModelPlugin >::Create()`, and `gazebo::PluginT< ModelPlugin >::GetFilename()`.

10.112.4.2 `template<class T> std::string gazebo::PluginT< T >::handle` [protected]

Short name.

Referenced by `gazebo::PluginT< ModelPlugin >::Create()`, and `gazebo::PluginT< ModelPlugin >::GetHandle()`.

10.112.4.3 `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:

- **common/Plugin.hh**

10.113 gazebo::math::Pose Class Reference

Encapsulates a position and rotation in three space.

```
#include <math/gzmath.hh>
```


Public Member Functions

- **Pose** ()
Default constructors.
- **Pose** (const **Vector3** &_pos, const **Quaternion** &_rot)
Constructor.
- **Pose** (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)
Constructor.
- **Pose** (const **Pose** &_pose)
Copy constructor.
- virtual ~**Pose** ()
Destructor.
- **Pose CoordPoseSolve** (const **Pose** &_b) const
Find the inverse of a pose; i.e., if $b = this + a$, given b and $this$, find a .
- **Vector3 CoordPositionAdd** (const **Vector3** &_pos) const
Add one point to a vector: result = this + pos.
- **Vector3 CoordPositionAdd** (const **Pose** &_pose) const
Add one point to another: result = this + pose.
- **Vector3 CoordPositionSub** (const **Pose** &_pose) const
Subtract one position from another: result = this - pose.
- **Quaternion CoordRotationAdd** (const **Quaternion** &_rot) const
Add one rotation to another: result = this->rot + rot.
- **Quaternion CoordRotationSub** (const **Quaternion** &_rot) const
Subtract one rotation from another: result = this->rot - rot.
- void **Correct** ()
Fix any nan values.
- **Pose GetInverse** () const
Get the inverse of this pose.
- bool **IsFinite** () const
See if a pose is finite (e.g., not nan)
- bool **operator!=** (const **Pose** &_pose) const
Inequality operator.
- **Pose operator*** (const **Pose** &_pose)
Multiplication operator.
- **Pose operator+** (const **Pose** &_pose) const
Addition operator.
- const **Pose** & **operator+=** (const **Pose** &_pose)
Add-Equals operator.
- **Pose operator-** () const
Negation operator.
- **Pose operator-** (const **Pose** &_pose) const
Subtraction operator.
- const **Pose** & **operator-=** (const **Pose** &_pose)
Subtraction 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. 839) and a **Quaternion** (p. 609).*
- 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.

10.113.1 Detailed Description

Encapsulates a position and rotation in three space.

10.113.2 Constructor & Destructor Documentation

10.113.2.1 gazebo::math::Pose::Pose ()

Default constructors.

Referenced by operator-().

10.113.2.2 gazebo::math::Pose::Pose (const **Vector3** & _pos, const **Quaternion** & _rot)

Constructor.

Parameters

in	<i>_pos</i>	A (p. 109) position
in	<i>_rot</i>	A (p. 109) rotation

10.113.2.3 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.

10.113.2.4 gazebo::math::Pose::Pose (const Pose & *_pose*)

Copy constructor.

Parameters

in	<i>_pose</i>	Pose (p. 584) to copy
----	--------------	------------------------------

10.113.2.5 virtual gazebo::math::Pose::~~Pose () [virtual]

Destructor.

10.113.3 Member Function Documentation

10.113.3.1 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
----	-----------	----------------

10.113.3.2 Vector3 gazebo::math::Pose::CoordPositionAdd (const Vector3 & *_pos*) const

Add one point to a vector: $\text{result} = \text{this} + \text{pos}$.

Parameters

in	<i>_pos</i>	Position to add to this pose
----	-------------	------------------------------

Returns

the resulting position

10.113.3.3 Vector3 gazebo::math::Pose::CoordPositionAdd (const Pose & *_pose*) const

Add one point to another: result = this + pose.

Parameters

in	<i>_pose</i>	The Pose (p. 584) to add
----	--------------	---------------------------------

Returns

The resulting position

10.113.3.4 Vector3 gazebo::math::Pose::CoordPositionSub (const Pose & *_pose*) const [inline]

Subtract one position from another: result = this - pose.

Parameters

in	<i>_pose</i>	Pose (p. 584) 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-().

10.113.3.5 Quaternion gazebo::math::Pose::CoordRotationAdd (const Quaternion & *_rot*) const

Add one rotation to another: result = this->rot + rot.

Parameters

in	<i>_rot</i>	Rotation to add
----	-------------	-----------------

Returns

The resulting rotation

10.113.3.6 Quaternion gazebo::math::Pose::CoordRotationSub (const Quaternion & *_rot*) const [inline]

Subtract one rotation from another: result = this->rot - rot.

Parameters

in	<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-().

10.113.3.7 void gazebo::math::Pose::Correct () [inline]

Fix any nan values.

References gazebo::math::Vector3::Correct(), gazebo::math::Quaternion::Correct(), pos, and rot.

10.113.3.8 Pose gazebo::math::Pose::GetInverse () const

Get the inverse of this pose.

Returns

the inverse pose

10.113.3.9 bool gazebo::math::Pose::IsFinite () const

See if a pose is finite (e.g., not nan)

10.113.3.10 bool gazebo::math::Pose::operator!=(const Pose & _pose) const

Inequality operator.

Parameters

<i>in</i>	<i>_pose</i>	Pose (p. 584) for comparison
-----------	--------------	-------------------------------------

Returns

True if not equal

10.113.3.11 Pose gazebo::math::Pose::operator* (const Pose & _pose)

Multiplication operator.

Parameters

<i>in</i>	<i>_pose</i>	the other pose
-----------	--------------	----------------

Returns

itself

10.113.3.12 **Pose** gazebo::math::Pose::operator+ (const Pose & *_pose*) const

Addition operator.

Parameters

in	<i>_pose</i>	Pose (p. 584) to add to this pose
----	--------------	--

Returns

The resulting pose

10.113.3.13 **const Pose&** gazebo::math::Pose::operator+= (const Pose & *_pose*)

Add-Equals operator.

Parameters

in	<i>_pose</i>	Pose (p. 584) to add to this pose
----	--------------	--

Returns

The resulting pose

10.113.3.14 **Pose** gazebo::math::Pose::operator- () const [inline]

Negation operator.

Returns

The resulting pose

References Pose().

10.113.3.15 **Pose** gazebo::math::Pose::operator- (const Pose & *_pose*) const [inline]

Subtraction operator.

Parameters

in	<i>_pose</i>	Pose (p. 584) to subtract from this one
----	--------------	--

Returns

The resulting pose

References CoordPositionSub(), CoordRotationSub(), Pose(), and rot.

10.113.3.16 **const Pose&** gazebo::math::Pose::operator-= (const Pose & *_pose*)

Subtraction operator.

Parameters

in	<i>_pose</i>	Pose (p. 584) to subtract from this one
----	--------------	--

Returns

The resulting pose

10.113.3.17 `bool gazebo::math::Pose::operator==(const Pose & _pose) const`

Equality operator.

Parameters

in	<i>_pose</i>	Pose (p. 584) for comparison
----	--------------	-------------------------------------

Returns

True if equal

10.113.3.18 `void gazebo::math::Pose::Reset ()`

Reset the pose.

10.113.3.19 `Pose gazebo::math::Pose::RotatePositionAboutOrigin (const Quaternion & _rot) const`

Rotate vector part of a pose about the origin.

Parameters

in	<i>_rot</i>	rotation
----	-------------	----------

Returns

the rotated pose

10.113.3.20 `void gazebo::math::Pose::Round (int _precision)`

Round all values to *_precision* decimal places.

Parameters

in	<i>_precision</i>	
----	-------------------	--

10.113.3.21 `void gazebo::math::Pose::Set (const Vector3 & _pos, const Quaternion & _rot)`

Set the pose from a **Vector3** (p. 839) and a **Quaternion** (p. 609).

Parameters

in	<code>_pos</code>	The position.
in	<code>_rot</code>	The rotation.

10.113.3.22 `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

in	<code>_x</code>	x position in meters.
in	<code>_y</code>	y position in meters.
in	<code>_z</code>	z position in meters.
in	<code>_roll</code>	Roll (rotation about X-axis) in radians.
in	<code>_pitch</code>	Pitch (rotation about y-axis) in radians.
in	<code>_yaw</code>	Pitch (rotation about z-axis) in radians.

10.113.4 Friends And Related Function Documentation

10.113.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Pose & _pose)` [`friend`]

Stream insertion operator.

Parameters

in	<code>_out</code>	output stream
in	<code>_pose</code>	pose to output

Returns

the stream

10.113.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Pose & _pose)` [`friend`]

Stream extraction operator.

Parameters

in	<code>_in</code>	the input stream
in	<code>_pose</code>	the pose

Returns

the stream

10.113.5 Member Data Documentation

10.113.5.1 `Vector3 gazebo::math::Pose::pos`

The position.

Referenced by CoordPositionSub(), and Correct().

10.113.5.2 Quaternion gazebo::math::Pose::rot

The rotation.

Referenced by CoordPositionSub(), CoordRotationSub(), Correct(), and operator-().

10.113.5.3 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:

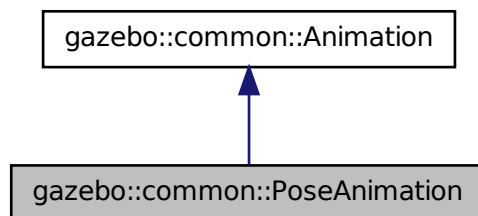
- **Pose.hh**

10.114 gazebo::common::PoseAnimation Class Reference

A (p. 109) 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

10.114.1 Detailed Description

A (p. 109) pose animation.

10.114.2 Constructor & Destructor Documentation

10.114.2.1 gazebo::common::PoseAnimation::PoseAnimation (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

10.114.2.2 virtual gazebo::common::PoseAnimation::~~PoseAnimation () [virtual]

Destructor.

10.114.3 Member Function Documentation

10.114.3.1 void gazebo::common::PoseAnimation::BuildInterpolationSplines () const [protected]

Update the pose splines.

10.114.3.2 **PoseKeyFrame*** gazebo::common::PoseAnimation::CreateKeyFrame (double _time)

Create a pose keyframe at the given time.

Parameters

in	<code>_time</code>	Time (p. 776) at which to create the keyframe
----	--------------------	--

Returns

Pointer to the new keyframe

10.114.3.3 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (PoseKeyFrame & _kf) const

Get a keyframe using the animation's current time.

Parameters

out	_kf	PoseKeyFrame (p. 595) reference to hold the interpolated result
-----	-----	--

10.114.3.4 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (double _time, PoseKeyFrame & _kf) const
[protected]

Get a keyframe using a passed in time.

Parameters

in	_time	Time (p. 776) in seconds
out	_kf	PoseKeyFrame (p. 595) reference to hold the interpolated result

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

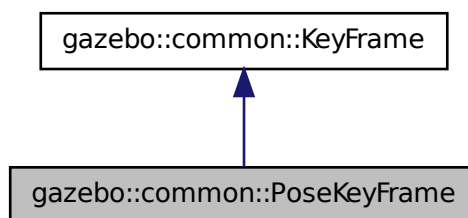
- **Animation.hh**

10.115 gazebo::common::PoseKeyFrame Class Reference

A (p. 109) keyframe for a **PoseAnimation** (p. 593).

```
#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

10.115.1 Detailed Description

A (p. 109) keyframe for a **PoseAnimation** (p. 593).

10.115.2 Constructor & Destructor Documentation

10.115.2.1 gazebo::common::PoseKeyFrame::PoseKeyFrame (double *_time*)

Constructor.

Parameters

<code>in</code>	<code>_time</code>	of the keyframe
-----------------	--------------------	-----------------

10.115.2.2 virtual gazebo::common::PoseKeyFrame::~~PoseKeyFrame () [virtual]

Destructor.

10.115.3 Member Function Documentation

10.115.3.1 const **math::Quaternion**& gazebo::common::PoseKeyFrame::GetRotation () const

Get the rotation of the keyframe.

Returns

The rotation amount

10.115.3.2 const **math::Vector3**& gazebo::common::PoseKeyFrame::GetTranslation () const

Get the translation of the keyframe.

Returns

The translation amount

10.115.3.3 void gazebo::common::PoseKeyFrame::SetRotation (const math::Quaternion & _rot)

Set the rotation for the keyframe.

Parameters

in	_rot	Rotation amount
----	------	-----------------

10.115.3.4 void gazebo::common::PoseKeyFrame::SetTranslation (const math::Vector3 & _trans)

Set the translation for the keyframe.

Parameters

in	_trans	Translation amount
----	--------	--------------------

10.115.4 Member Data Documentation

10.115.4.1 math::Quaternion gazebo::common::PoseKeyFrame::rotate [protected]

the rotation quaternion

10.115.4.2 math::Vector3 gazebo::common::PoseKeyFrame::translate [protected]

the translation vector

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

- **KeyFrame.hh**

10.116 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.

10.116.1 Detailed Description

Projects a material onto surface, light a light projector.

10.116.2 Constructor & Destructor Documentation

10.116.2.1 gazebo::rendering::Projector::Projector (**VisualPtr** _parent)

Constructor.

Parameters

in	<i>_parent</i>	Name of the parent visual.
----	----------------	----------------------------

10.116.2.2 virtual gazebo::rendering::Projector::~~Projector () [virtual]

Destructor.

10.116.3 Member Function Documentation

10.116.3.1 **VisualPtr** gazebo::rendering::Projector::GetParent ()

Get the parent visual.

Returns

Pointer of the parent visual.

10.116.3.2 void gazebo::rendering::Projector::Load (**sdf::ElementPtr** _sdf)

Load from an sdf pointer.

Parameters

in	<code>_sdf</code>	Pointer to the SDF element.
----	-------------------	-----------------------------

10.116.3.3 `void gazebo::rendering::Projector::Load (const msgs::Projector & _msg)`

Load from a message.

Parameters

in	<code>_msg</code>	Load from a message.
----	-------------------	----------------------

10.116.3.4 `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.

10.116.3.5 `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.
----	-----------------------	-------------------------------

10.116.3.6 `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.
----	---------------------------	---------------------------------

10.116.3.7 `void gazebo::rendering::Projector::Toggle ()`

Toggle the activation of the projector.

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

- **Projector.hh**

10.117 gazebo::transport::Publication Class Reference

A (p. 109) 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** (const google::protobuf::Message &_msg, const boost::function< void()> &_cb=**NULL**)
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.

10.117.1 Detailed Description

A (p. 109) publication for a topic.

This facilitates transport of messages

10.117.2 Constructor & Destructor Documentation

10.117.2.1 gazebo::transport::Publication::Publication (const std::string & *_topic*, const std::string & *_msgType*)

Constructor.

Parameters

in	<i>_topic</i>	The topic we're publishing
in	<i>_msgType</i>	The type of the topic we're publishing

10.117.2.2 virtual gazebo::transport::Publication::~~Publication () [virtual]

Destructor.

10.117.3 Member Function Documentation

10.117.3.1 void gazebo::transport::Publication::AddPublisher (PublisherPtr *_pub*)

Add a publisher.

Parameters

in, out	<i>_pub</i>	Pointer to publisher object to be added
---------	-------------	---

Referenced by gazebo::transport::TopicManager::Advertise().

10.117.3.2 void gazebo::transport::Publication::AddSubscription (const CallbackHelperPtr *_callback*)

Subscribe a callback to our topic.

Parameters

in	<i>_callback</i>	The callback
----	------------------	--------------

Referenced by gazebo::transport::TopicManager::Advertise().

10.117.3.3 void gazebo::transport::Publication::AddSubscription (const NodePtr & *_node*)

Subscribe a node to our topic.

Parameters

in	<i>_node</i>	The node
----	--------------	----------

10.117.3.4 `void gazebo::transport::Publication::AddTransport (const PublicationTransportPtr & _publink)`

Add a transport.

Parameters

<code>in</code>	<code>_publink</code>	Pointer to publication transport object to be added
-----------------	-----------------------	---

10.117.3.5 `unsigned int gazebo::transport::Publication::GetCallbackCount () const`

Get the number of callbacks.

Returns

The number of callbacks

10.117.3.6 `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

Referenced by `gazebo::transport::TopicManager::Advertise()`.

10.117.3.7 `std::string gazebo::transport::Publication::GetMsgType () const`

Get the type of message.

Returns

The type of message

10.117.3.8 `unsigned int gazebo::transport::Publication::GetNodeCount () const`

Get the number of nodes.

Returns

The number of nodes

10.117.3.9 `unsigned int gazebo::transport::Publication::GetRemoteSubscriptionCount ()`

Get the number of remote subscriptions.

Returns

The number of remote subscriptions

10.117.3.10 `unsigned int gazebo::transport::Publication::GetTransportCount () const`

Get the number of transports.

Returns

The number of transports

10.117.3.11 `bool gazebo::transport::Publication::HasTransport (const std::string & _host, unsigned int _port)`

Does a given transport exist?

Parameters

<code>in</code>	<code><i>_host</i></code>	Hostname of the transport
<code>in</code>	<code><i>_port</i></code>	Port of the transport

Returns

true if the transport exists, false otherwise

10.117.3.12 `void gazebo::transport::Publication::LocalPublish (const std::string & _data)`

Publish data to local subscribers (skip serialization)

Parameters

<code>in</code>	<code><i>_data</i></code>	The data to be published
-----------------	---------------------------	--------------------------

10.117.3.13 `void gazebo::transport::Publication::Publish (const google::protobuf::Message & _msg, const boost::function< void()> & _cb = NULL)`

Publish data to remote subscribers.

Parameters

<code>in</code>	<code><i>_msg</i></code>	Message to be published
<code>in</code>	<code><i>_cb</i></code>	If non-null, callback to be invoked after publishing is completed

10.117.3.14 `void gazebo::transport::Publication::RemoveSubscription (const NodePtr & _node)`

Unsubscribe a node from our topic.

Parameters

<code>in</code>	<code><i>_node</i></code>	The node
-----------------	---------------------------	----------

10.117.3.15 `void gazebo::transport::Publication::RemoveSubscription (const std::string & _host, unsigned int _port)`

Unsubscribe a a node by host/port from our topic.

Parameters

<code>in</code>	<code>_host</code>	The node's hostname
<code>in</code>	<code>_port</code>	The node's port

10.117.3.16 `void gazebo::transport::Publication::RemoveTransport (const std::string & _host, unsigned int _port)`

Remove a transport.

Parameters

<code>in</code>	<code>_host</code>	The transport's hostname
<code>in</code>	<code>_port</code>	The transport's port

10.117.3.17 `void gazebo::transport::Publication::SetLocallyAdvertised (bool _value)`

Set whether this topic has been advertised from this process.

Parameters

<code>in</code>	<code>_value</code>	If true, the topic was locally advertise, otherwise it was not
-----------------	---------------------	--

Referenced by `gazebo::transport::TopicManager::Advertise()`.

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

- **Publication.hh**

10.118 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.

10.118.1 Detailed Description

transport/transport.hh

Reads data from a remote advertiser, and passes the data along to local subscribers

10.118.2 Constructor & Destructor Documentation

10.118.2.1 `gazebo::transport::PublicationTransport::PublicationTransport (const std::string & _topic, const std::string & _msgType)`

Constructor.

Parameters

<code>in</code>	<code>_topic</code>	Topic that we're publishing
<code>in</code>	<code>_topic</code>	Type of the topic that we're publishing

10.118.2.2 `virtual gazebo::transport::PublicationTransport::~~PublicationTransport () [virtual]`

Destructor.

10.118.3 Member Function Documentation

10.118.3.1 `void gazebo::transport::PublicationTransport::AddCallback (const boost::function< void(const std::string &)> & _cb)`

Add a callback to the transport.

Parameters

<code>in</code>	<code>_cb</code>	The callback to be added
-----------------	------------------	--------------------------

10.118.3.2 `void gazebo::transport::PublicationTransport::Fini ()`

Finalize the transport.

10.118.3.3 `const ConnectionPtr gazebo::transport::PublicationTransport::GetConnection () const`

Get the underlying connection.

Returns

Pointer to the underlying connection

10.118.3.4 `std::string gazebo::transport::PublicationTransport::GetMsgType () const`

Get the topic type.

Returns

The topic type

10.118.3.5 `std::string gazebo::transport::PublicationTransport::GetTopic () const`

Get the topic name.

Returns

The topic name

10.118.3.6 `void gazebo::transport::PublicationTransport::Init (const ConnectionPtr & _conn, bool _latched)`

Initialize the transport.

Parameters

<code>in</code>	<code>_conn</code>	The underlying connection.
<code>in</code>	<code>_latched</code>	True to grab the last message sent on the topic.

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

- **PublicationTransport.hh**

10.119 gazebo::transport::Publisher Class Reference

A (p. 109) 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, bool _latch) **GAZEBO_DEPRECATED**
Deprecated.
- **Publisher** (const std::string &_topic, const std::string &_msgType, unsigned int _limit, double _hzRate)
Constructor.
- virtual **~Publisher** ()
Destructor.

- bool **GetLatching** () const **GAZEBO_DEPRECATED**
Deprecated.
- 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.
- 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 **SetPublication** (**PublicationPtr** &_publication, int _i)
Set the publication object for a particular publication.
- void **WaitForConnection** () const
Block until a connection has been established with this publisher.

10.119.1 Detailed Description

A (p. 109) publisher of messages on a topic.

10.119.2 Constructor & Destructor Documentation

10.119.2.1 gazebo::transport::Publisher::Publisher (const std::string & *_topic*, const std::string & *_msgType*, unsigned int *_limit*, bool *_latch*)

Deprecated.

10.119.2.2 gazebo::transport::Publisher::Publisher (const std::string & *_topic*, const std::string & *_msgType*, unsigned int *_limit*, double *_hzRate*)

Constructor.

Parameters

in	<i>_topic</i>	Name of topic to be published
in	<i>_msgType</i>	Type of the message to be published
in	<i>_limit</i>	Maximum number of outgoing messages to queue
in	<i>_hz</i>	Update rate for the publisher. Units are 1.0/seconds.

10.119.2.3 `virtual gazebo::transport::Publisher::~~Publisher () [virtual]`

Destructor.

10.119.3 Member Function Documentation

10.119.3.1 `bool gazebo::transport::Publisher::GetLatching () const`

Deprecated.

10.119.3.2 `std::string gazebo::transport::Publisher::GetMsgType () const`

Get the message type.

Returns

The message type

10.119.3.3 `unsigned int gazebo::transport::Publisher::GetOutgoingCount () const`

Get the number of outgoing messages.

Returns

The number of outgoing messages

10.119.3.4 `std::string gazebo::transport::Publisher::GetPrevMsg () const`

Get the previously published message.

Returns

The previously published message, if any

10.119.3.5 `std::string gazebo::transport::Publisher::GetTopic () const`

Get the topic name.

Returns

The topic name

10.119.3.6 `bool gazebo::transport::Publisher::HasConnections () const`

Are there any connections?

Returns

true if there are any connections, false otherwise

10.119.3.7 `void gazebo::transport::Publisher::Publish (const google::protobuf::Message & _message, bool _block = false)`
`[inline]`

Publish a protobuf message on the topic.

Parameters

<code>in</code>	<code><i>_message</i></code>	Message to be published
<code>in</code>	<code><i>_block</i></code>	Whether to block until the message is actually written out

10.119.3.8 `template<typename M > void gazebo::transport::Publisher::Publish (M _message, bool _block = false)`
`[inline]`

Publish an arbitrary message on the topic.

Parameters

<code>in</code>	<code><i>_message</i></code>	Message to be published
<code>in</code>	<code><i>_block</i></code>	Whether to block until the message is actually written out

10.119.3.9 `void gazebo::transport::Publisher::SendMessage ()`

Send latest message over the wire. For internal use only.

10.119.3.10 `void gazebo::transport::Publisher::SetPublication (PublicationPtr & _publication, int _i)`

Set the publication object for a particular publication.

Parameters

<code>in</code>	<code><i>_publication</i></code>	Pointer to the publication object to be set
<code>in</code>	<code><i>_i</i></code>	Index into publications vector that will be set

10.119.3.11 `void gazebo::transport::Publisher::WaitForConnection () const`

Block until a connection has been established with this publisher.

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

- **Publisher.hh**

10.120 gazebo::math::Quaternion Class Reference

A (p. 109) 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. 839) 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.

10.120.1 Detailed Description

A (p. 109) quaternion class.

10.120.2 Constructor & Destructor Documentation

10.120.2.1 gazebo::math::Quaternion::Quaternion ()

Default Constructor.

Referenced by operator*().

10.120.2.2 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

10.120.2.3 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

10.120.2.4 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

10.120.2.5 gazebo::math::Quaternion::Quaternion (const Vector3 & *_rpy*)

Constructor.

Parameters

in	<i>_rpy</i>	euler angles
----	-------------	--------------

10.120.2.6 gazebo::math::Quaternion::Quaternion (const Quaternion & *_qt*)

Copy constructor.

Parameters

<i>qt</i>	Quaternion (p. 609) to copy
-----------	------------------------------------

10.120.2.7 gazebo::math::Quaternion::~~Quaternion ()

Destructor.

10.120.3 Member Function Documentation

10.120.3.1 void gazebo::math::Quaternion::Correct () [inline]

Correct any nan.

References gazebo::math::equal(), w, x, y, and z.

Referenced by gazebo::math::Pose::Correct().

10.120.3.2 double gazebo::math::Quaternion::Dot (const Quaternion & _q) const

Dot product.

Parameters

<code>in</code>	<code>_q</code>	the other quaternion
-----------------	-----------------	----------------------

Returns

the product

10.120.3.3 static Quaternion gazebo::math::Quaternion::EulerToQuaternion (const Vector3 & _vec) [static]

Convert euler angles to quatern.

Parameters

<code>in</code>		
-----------------	--	--

10.120.3.4 static Quaternion gazebo::math::Quaternion::EulerToQuaternion (double _x, double _y, double _z) [static]

Convert euler angles to quatern.

Parameters

<code>in</code>	<code>_x</code>	rotation along x
<code>in</code>	<code>_y</code>	rotation along y
<code>in</code>	<code>_z</code>	rotation along z

10.120.3.5 void gazebo::math::Quaternion::GetAsAxis (Vector3 & _axis, double & _angle) const

Return rotation as axis and angle.

Parameters

<code>in</code>	<code>_axis</code>	rotation axis
<code>in</code>	<code>_angle</code>	ccw angle in radians

10.120.3.6 **Vector3** gazebo::math::Quaternion::GetAsEuler () const

Return the rotation in Euler angles.

Returns

This quaternion as an Euler vector

10.120.3.7 **Matrix3** gazebo::math::Quaternion::GetAsMatrix3 () const

Get the quaternion as a 3x3 matrix.

10.120.3.8 **Matrix4** gazebo::math::Quaternion::GetAsMatrix4 () const

Get the quaternion as a 4x4 matrix.

Returns

a 4x4 matrix

10.120.3.9 **Quaternion** gazebo::math::Quaternion::GetExp () const

Return the exponent.

Returns

the exp

10.120.3.10 **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 Rotate-Vector().

10.120.3.11 **Quaternion** gazebo::math::Quaternion::GetLog () const

Return the logarithm.

Returns

the log

10.120.3.12 `double gazebo::math::Quaternion::GetPitch ()`

Get the Euler pitch angle in radians.

Returns

the pitch

10.120.3.13 `double gazebo::math::Quaternion::GetRoll ()`

Get the Euler roll angle in radians.

Returns

the roll

10.120.3.14 `Vector3 gazebo::math::Quaternion::GetXAxis () const`

Return the X axis.

Returns

the vector

10.120.3.15 `double gazebo::math::Quaternion::GetYaw ()`

Get the Euler yaw angle in radians.

Returns

the yaw

10.120.3.16 `Vector3 gazebo::math::Quaternion::GetYAxis () const`

Return the Y axis.

Returns

the vector

10.120.3.17 `Vector3 gazebo::math::Quaternion::GetZAxis () const`

Return the Z axis.

Returns

the vector

10.120.3.18 void gazebo::math::Quaternion::Invert ()

Invert the quaternion.

10.120.3.19 bool gazebo::math::Quaternion::IsFinite () const

See if a quatern is finite (e.g., not nan)

Returns

True if quatern is finite

10.120.3.20 void gazebo::math::Quaternion::Normalize ()

Normalize the quaternion.

Referenced by gazebo::math::Pose::CoordRotationSub().

10.120.3.21 bool gazebo::math::Quaternion::operator!=(const Quaternion & _qt) const

Not equal to operator.

Parameters

in	_qt	Quaternion (p. 609) for comparison
----	-----	---

Returns

True if not equal

10.120.3.22 Quaternion gazebo::math::Quaternion::operator* (const Quaternion & _q) const [inline]

Multiplication operator.

Parameters

in	_qt	Quaternion (p. 609) for multiplication
----	-----	---

Returns

This quaternion multiplied by the parameter

References Quaternion(), w, x, y, and z.

10.120.3.23 Quaternion gazebo::math::Quaternion::operator* (const double & _f) const

Multiplication operator.

Parameters

<code>in</code>	<code>_f</code>	factor
-----------------	-----------------	--------

Returns

quaternion multiplied by `_f`

10.120.3.24 **Vector3** gazebo::math::Quaternion::operator* (const Vector3 & `_v`) const

Vector3 (p. 839) multiplication operator.

Parameters

<code>in</code>	<code>_v</code>	vector to multiply
-----------------	-----------------	--------------------

10.120.3.25 **Quaternion** gazebo::math::Quaternion::operator*= (const Quaternion & `qt`)

Multiplication operator.

Parameters

<code>in</code>	<code>_qt</code>	Quaternion (p. 609) for multiplication
-----------------	------------------	---

Returns

This quatern multiplied by the parameter

10.120.3.26 **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`

10.120.3.27 **Quaternion** gazebo::math::Quaternion::operator+= (const Quaternion & `_qt`)

Addition operator.

Parameters

<code>in</code>	<code>_qt</code>	quaternion for addition
-----------------	------------------	-------------------------

Returns

this quaternion + qt

10.120.3.28 Quaternion gazebo::math::Quaternion::operator- (const Quaternion & _qt) const

Substraction operator.

Parameters

in	_qt	quaternion to subtract
----	-----	------------------------

Returns

this quaternion - _qt

10.120.3.29 Quaternion gazebo::math::Quaternion::operator- () const

Unary minus operator.

Returns

negates each component of the quaternion

10.120.3.30 Quaternion gazebo::math::Quaternion::operator-= (const Quaternion & _qt)

Substraction operator.

Parameters

in	_qt	Quaternion (p. 609) for subtraction
----	-----	--

Returns

This quatern - qt

10.120.3.31 Quaternion& gazebo::math::Quaternion::operator= (const Quaternion & _qt)

Equal operator.

Parameters

in	_qt	Quaternion (p. 609) to copy
----	-----	------------------------------------

10.120.3.32 bool gazebo::math::Quaternion::operator==(const Quaternion & _qt) const

Equal to operator.

Parameters

in	<code>_qt</code>	Quaternion (p. 609) for comparison
----	------------------	---

Returns

True if equal

10.120.3.33 **Vector3** gazebo::math::Quaternion::RotateVector (const **Vector3** & `_vec`) const [inline]

Rotate a vector using the quaternion.

Parameters

in	<code>_vec</code>	vector to rotate
----	-------------------	------------------

Returns

the rotated vector

References GetInverse(), gazebo::math::Vector3::x, gazebo::math::Vector3::y, gazebo::math::Vector3::z, and z.

10.120.3.34 **Vector3** gazebo::math::Quaternion::RotateVectorReverse (**Vector3** `_vec`) const

Do the reverse rotation of a vector by this quaternion.

Parameters

in	<code>_vec</code>	the vector
----	-------------------	------------

Returns

the

10.120.3.35 void gazebo::math::Quaternion::Round (int `_precision`)

Round all values to `_precision` decimal places.

Parameters

in	<code>_precision</code>	the precision
----	-------------------------	---------------

10.120.3.36 void gazebo::math::Quaternion::Scale (double `_scale`)

Scale a Quaternionion.

Parameters

in	<code>_scale</code>	Amount to scale this rotation
----	---------------------	-------------------------------

10.120.3.37 `void gazebo::math::Quaternion::Set (double _u, double _x, double _y, double _z)`

Set this quaternion from 4 floating numbers.

Parameters

in	<code>_u</code>	u
in	<code>_x</code>	x
in	<code>_y</code>	y
in	<code>_z</code>	z

10.120.3.38 `void gazebo::math::Quaternion::SetFromAxis (double _x, double _y, double _z, double _a)`

Set the quaternion from an axis and angle.

Parameters

in	<code>_x</code>	X axis
in	<code>_y</code>	Y axis
in	<code>_z</code>	Z axis
in	<code>_a</code>	Angle (p. 116) in radians

10.120.3.39 `void gazebo::math::Quaternion::SetFromAxis (const Vector3 & _axis, double _a)`

Set the quaternion from an axis and angle.

Parameters

in	<code>_axis</code>	Axis
in	<code>_a</code>	Angle (p. 116) in radians

10.120.3.40 `void gazebo::math::Quaternion::SetFromEuler (const Vector3 & _vec)`

Set the quaternion from Euler angles.

Parameters

in	<code>vec</code>	Euler angle
----	------------------	-------------

10.120.3.41 `void gazebo::math::Quaternion::SetFromEuler (double _roll, double _pitch, double _yaw)`

Set the quaternion from Euler angles.

Parameters

in	<code>_roll</code>	Roll angle (radians).
in	<code>_pitch</code>	Roll angle (radians).
in	<code>_yaw</code>	Roll angle (radians).

10.120.3.42 `void gazebo::math::Quaternion::SetToIdentity ()`

Set the quatern to the identity.

10.120.3.43 `static Quaternion gazebo::math::Quaternion::Slerp (double _ft, const Quaternion & _rkP, const Quaternion & _rkQ, bool _shortestPath = false) [static]`

Spherical linear interpolation between 2 quaternions, given the ends and an interpolation parameter between 0 and 1.

Parameters

in	<code><i>_ft</i></code>	the interpolation parameter
in	<code><i>_rkP</i></code>	the beginning quaternion
in	<code><i>_rkQ</i></code>	the end quaternion
in	<code><i>_shortestPath</i></code>	when true, the rotation may be inverted to get to minimize rotation

10.120.3.44 `static Quaternion gazebo::math::Quaternion::Squad (double _ft, const Quaternion & _rkP, const Quaternion & _rkA, const Quaternion & _rkB, const Quaternion & _rkQ, bool _shortestPath = false) [static]`

Spherical quadratic interpolation given the ends and an interpolation parameter between 0 and 1.

Parameters

in	<code><i>_ft</i></code>	the interpolation parameter
in	<code><i>_rkP</i></code>	the beginning quaternion
in	<code><i>_rkA</i></code>	first intermediate quaternion
in	<code><i>_rkB</i></code>	second intermediate quaternion
in	<code><i>_rkQ</i></code>	the end quaternion
in	<code><i>_shortestPath</i></code>	when true, the rotation may be inverted to get to minimize rotation

10.120.4 Friends And Related Function Documentation

10.120.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Quaternion & _q) [friend]`

Stream insertion operator.

Parameters

in	<code><i>_out</i></code>	output stream
in	<code><i>_q</i></code>	quaternion to output

Returns

the stream

10.120.4.2 `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. 609) to read values into

Returns

The istream

10.120.5 Member Data Documentation

10.120.5.1 double gazebo::math::Quaternion::w

Attributes of the quaternion.

Referenced by `Correct()`, `GetInverse()`, and `operator*()`.

10.120.5.2 double gazebo::math::Quaternion::x

Attributes of the quaternion.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `GetInverse()`, `operator*()`, and `RotateVector()`.

10.120.5.3 double gazebo::math::Quaternion::y

Attributes of the quaternion.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `GetInverse()`, `operator*()`, and `RotateVector()`.

10.120.5.4 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**

10.121 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.

10.121.1 Detailed Description

Random number generator class.

10.121.2 Member Function Documentation

10.121.2.1 static double gazebo::math::Rand::GetDbNormal (double _mean = 0, double _sigma = 1) [static]

Get a double from a normal distribution.

Parameters

in	<code>_mean</code>	Mean value for the distribution
in	<code>_sigma</code>	Sigma value for the distribution

10.121.2.2 static double gazebo::math::Rand::GetDbUniform (double _min = 0, double _max = 1) [static]

Get a double from a uniform distribution.

Parameters

in	<code>_min</code>	Minimum bound for the random number
in	<code>_max</code>	Maximum bound for the random number

10.121.2.3 static int gazebo::math::Rand::GetIntNormal (int _mean, int _sigma) [static]

Get a double from a normal distribution.

Parameters

in	<code>_mean</code>	Mean value for the distribution
in	<code>_sigma</code>	Sigma value for the distribution

10.121.2.4 static int gazebo::math::Rand::GetIntUniform (int _min, int _max) [static]

Get a integer from a uniform distribution.

Parameters

in	<code>_min</code>	Minimum bound for the random number
in	<code>_max</code>	Maximum bound for the random number

10.121.2.5 `static uint32_t gazebo::math::Rand::GetSeed () [static]`

Get the seed value.

Returns

The seed value used to initialize the random number generator.

10.121.2.6 `static void gazebo::math::Rand::SetSeed (uint32_t _seed) [static]`

Set the seed value.

Parameters

in	<code>_seed</code>	The seed used to initialize the random number generator.
----	--------------------	--

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

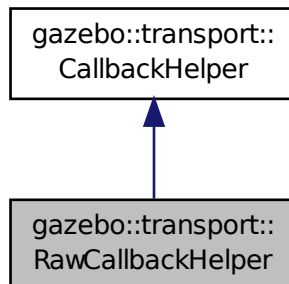
- **Rand.hh**

10.122 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)
Process new incoming data.
- virtual bool **IsLocal** () const
Is the callback local?

Additional Inherited Members

10.122.1 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.

10.122.2 Constructor & Destructor Documentation

10.122.2.1 gazebo::transport::RawCallbackHelper::RawCallbackHelper (const boost::function< void(const std::string &);> &_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.

10.122.3 Member Function Documentation

10.122.3.1 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. 157).

10.122.3.2 virtual bool gazebo::transport::RawCallbackHelper::HandleData (const std::string &_newdata) [inline],[virtual]

Process new incoming data.

Parameters

in	<code>_newdata</code>	Incoming data to be processed
----	-----------------------	-------------------------------

Returns

true if successfully processed; false otherwise

Implements `gazebo::transport::CallbackHelper` (p. 157).

10.122.3.3 `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. 157).

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

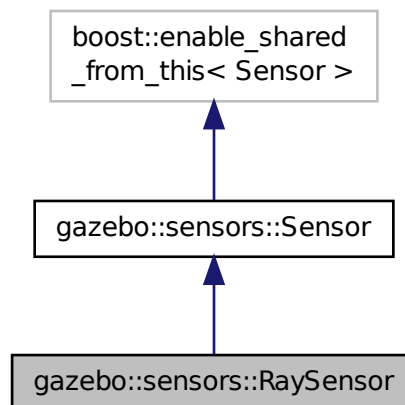
- `CallbackHelper.hh`

10.123 gazebo::sensors::RaySensor Class Reference

Sensor (p. 683) with one or more rays.

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

Inheritance diagram for `gazebo::sensors::RaySensor`:



Public Member Functions

- **RaySensor** ()
Constructor.
- virtual \sim **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. 518).*
- 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

10.123.1 Detailed Description

Sensor (p. 683) 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).

10.123.2 Constructor & Destructor Documentation

10.123.2.1 gazebo::sensors::RaySensor::RaySensor ()

Constructor.

10.123.2.2 virtual gazebo::sensors::RaySensor::~~RaySensor () [virtual]

Destructor.

10.123.3 Member Function Documentation

10.123.3.1 virtual void gazebo::sensors::RaySensor::Fini () [protected], [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 687).

10.123.3.2 math::Angle gazebo::sensors::RaySensor::GetAngleMax () const

Get the maximum angle.

Returns

the maximum angle object

10.123.3.3 math::Angle gazebo::sensors::RaySensor::GetAngleMin () const

Get the minimum angle.

Returns

The minimum angle object

10.123.3.4 `double gazebo::sensors::RaySensor::GetAngleResolution () const`

Get the angle in radians between each range.

Returns

Resolution of the angle

10.123.3.5 `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

<code>in</code>	<code><i>_index</i></code>	Index value of specific ray
-----------------	----------------------------	-----------------------------

Returns

Fiducial value

10.123.3.6 `physics::MultiRayShapePtr gazebo::sensors::RaySensor::GetLaserShape () const` `[inline]`

Returns a pointer to the internal `physics::MultiRayShape` (p. 518).

Returns

Pointer to ray shape

10.123.3.7 `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

<code>in</code>	<code><i>_index</i></code>	Index of specific ray
-----------------	----------------------------	-----------------------

Returns

Returns DBL_MAX for no detection.

10.123.3.8 `int gazebo::sensors::RaySensor::GetRangeCount () const`

Get the range count.

Returns

The number of ranges

10.123.3.9 `double gazebo::sensors::RaySensor::GetRangeMax () const`

Get the maximum range.

Returns

The maximum range

10.123.3.10 `double gazebo::sensors::RaySensor::GetRangeMin () const`

Get the minimum range.

Returns

The minimum range

10.123.3.11 `double gazebo::sensors::RaySensor::GetRangeResolution () const`

Get the range resolution.

Returns

Resolution of the range

10.123.3.12 `void gazebo::sensors::RaySensor::GetRanges (std::vector< double > & _ranges)`

Get all the ranges.

Parameters

<code>_ranges</code>	A (p. 109) vector that will contain all the range data
----------------------	---

10.123.3.13 `int gazebo::sensors::RaySensor::GetRayCount () const`

Get the ray count.

Returns

The number of rays

10.123.3.14 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

in	<i>_index</i>	Index of specific ray
----	---------------	-----------------------

Returns

Retro (intensity) value for ray

10.123.3.15 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. 689).

10.123.3.16 math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMax () const

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.123.3.17 math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMin () const

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.123.3.18 `int gazebo::sensors::RaySensor::GetVerticalRangeCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.123.3.19 `int gazebo::sensors::RaySensor::GetVerticalRayCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.123.3.20 `virtual void gazebo::sensors::RaySensor::Init () [virtual]`

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 690).

10.123.3.21 `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. 690).

10.123.3.22 `virtual void gazebo::sensors::RaySensor::Load (const std::string & _worldName) [virtual]`

Load the sensor with default parameters.

Parameters

<code>in</code>	<code>_worldName</code>	Name of world to load from.
-----------------	-------------------------	-----------------------------

Reimplemented from **gazebo::sensors::Sensor** (p. 690).

10.123.3.23 `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. 691).

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

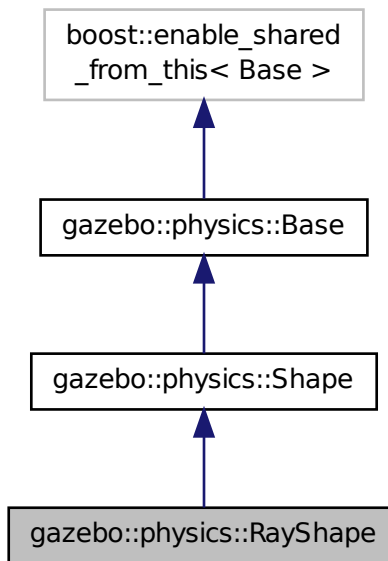
- `RaySensor.hh`

10.124 gazebo::physics::RayShape Class Reference

Base (p. 135) 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 **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

10.124.1 Detailed Description

Base (p. 135) class for Ray collision geometry.

10.124.2 Constructor & Destructor Documentation

10.124.2.1 gazebo::physics::RayShape::RayShape (PhysicsEnginePtr *_physicsEngine*) [explicit]

Constructor for a global ray.

Parameters

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

10.124.2.2 gazebo::physics::RayShape::RayShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Collision (p. 192) parent of the shape.
----	----------------	--

10.124.2.3 virtual gazebo::physics::RayShape::~~RayShape () [virtual]

Destructor.

10.124.3 Member Function Documentation

10.124.3.1 void gazebo::physics::RayShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Fill a message with data from this object.

Parameters

out	<i>_msg</i>	Message to fill. Implement this function.
-----	-------------	---

Implements **gazebo::physics::Shape** (p. 708).

10.124.3.2 int gazebo::physics::RayShape::GetFiducial () const

Get the fiducial id detected by this ray.

Returns

Fiducial id detected.

10.124.3.3 `virtual void gazebo::physics::RayShape::GetGlobalPoints (math::Vector3 & _posA, math::Vector3 & _posB)`
`[virtual]`

Get the global starting and ending points.

Parameters

out	<code><i>_posA</i></code>	Returns the starting point.
out	<code><i>_posB</i></code>	Returns the ending point.

10.124.3.4 `virtual void gazebo::physics::RayShape::GetIntersection (double & _dist, std::string & _entity)` `[pure virtual]`

Get the nearest intersection.

Parameters

out	<code><i>_dist</i></code>	Distance to the intersection.
out	<code><i>_entity</i></code>	Name of the entity the ray intersected with.

10.124.3.5 `double gazebo::physics::RayShape::GetLength () const`

Get the length of the ray.

Returns

The ray length.

10.124.3.6 `virtual void gazebo::physics::RayShape::GetRelativePoints (math::Vector3 & _posA, math::Vector3 & _posB)`
`[virtual]`

Get the relative starting and ending points.

Parameters

in	<code><i>_posA</i></code>	Returns the starting point.
in	<code><i>_posB</i></code>	Returns the ending point.

10.124.3.7 `float gazebo::physics::RayShape::GetRetro () const`

Get the retro-reflectiveness detected by this ray.

Returns

Retro reflectance value.

10.124.3.8 `virtual void gazebo::physics::RayShape::Init ()` `[virtual]`

In the ray.

Implements **gazebo::physics::Shape** (p. 708).

10.124.3.9 `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. 708).

10.124.3.10 `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.
-----------------	-------------------	-----------------------------------

10.124.3.11 `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.
-----------------	-------------------	----------------------

10.124.3.12 `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.

10.124.3.13 `void gazebo::physics::RayShape::SetRetro (float _retro)`

Set the retro-reflectiveness detected by this ray.

Parameters

<code>in</code>	<code>_retro</code>	Retro reflectance value.
-----------------	---------------------	--------------------------

10.124.3.14 `virtual void gazebo::physics::RayShape::Update () [pure virtual]`

Update the ray collision.

Reimplemented from `gazebo::physics::Base` (p. 146).

10.124.4 Member Data Documentation

10.124.4.1 `int gazebo::physics::RayShape::contactFiducial [protected]`

Fiducial ID value.

10.124.4.2 `double gazebo::physics::RayShape::contactLen [protected]`

Length of the ray.

10.124.4.3 `double gazebo::physics::RayShape::contactRetro [protected]`

Retro reflectance value.

10.124.4.4 `math::Vector3 gazebo::physics::RayShape::globalEndPos [protected]`

End position of the ray in global cs.

10.124.4.5 `math::Vector3 gazebo::physics::RayShape::globalStartPos [protected]`

Start position of the ray in global cs.

10.124.4.6 `math::Vector3 gazebo::physics::RayShape::relativeEndPos [protected]`

End position of the ray, relative to the body.

10.124.4.7 `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:

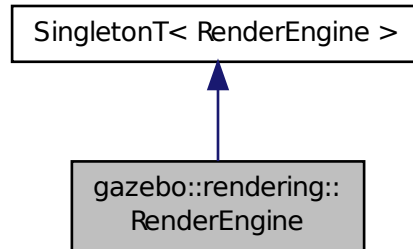
- **RayShape.hh**

10.125 gazebo::rendering::RenderEngine Class Reference

Adaptor to Ogre3d.

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

Inheritance diagram for gazebo::rendering::RenderEngine:



Public Types

- enum **RenderPathType** {
NONE, **VERTEX**, **FORWARD**, **DEFERRED**,
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. 106) to search for resources.*
- **ScenePtr CreateScene** (const std::string &_name, bool _enableVisualizations)
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.
- void **Init** ()
*Initialize **Ogre** (p. 106). Load must happen before Init.*
- void **Load** ()
*Load the parameters for **Ogre** (p. 106). 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

10.125.1 Detailed Description

Adaptor to Ogre3d.

Provides the interface to load, initialize the rendering engine.

10.125.2 Member Enumeration Documentation

10.125.2.1 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.

10.125.3 Member Function Documentation

10.125.3.1 void gazebo::rendering::RenderEngine::AddResourcePath (const std::string & _uri)

Add a new path for **Ogre** (p. 106) 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>
----	------	--

10.125.3.2 ScenePtr gazebo::rendering::RenderEngine::CreateScene (const std::string & *_name*, bool *_enableVisualizations*)

Create a scene.

Parameters

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

10.125.3.3 void gazebo::rendering::RenderEngine::Fini ()

Tears down the rendering engine.

10.125.3.4 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

10.125.3.5 ScenePtr gazebo::rendering::RenderEngine::GetScene (const std::string & *_name*)

Get a scene by name.

Parameters

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

Returns

A (p. 109) pointer to the **Scene** (p. 662), or NULL if the scene doesn't exist.

10.125.3.6 ScenePtr gazebo::rendering::RenderEngine::GetScene (unsigned int *_index*)

Get a scene by index.

The index should be between 0 and **GetSceneCount()** (p. 643).

Parameters

in	<i>_index</i>	The index of the scene.
----	---------------	-------------------------

Returns

A (p. 109) pointer to a **Scene** (p. 662), or NULL if the index was invalid.

10.125.3.7 `unsigned int gazebo::rendering::RenderEngine::GetSceneCount () const`

Get the number of scenes.

Returns

The number of scenes created by the **RenderEngine** (p. 639).

10.125.3.8 `void gazebo::rendering::RenderEngine::Init ()`

Initialize **Ogre** (p. 106). Load must happen before Init.

10.125.3.9 `void gazebo::rendering::RenderEngine::Load ()`

Load the parameters for **Ogre** (p. 106). Load must happen before Init.

10.125.3.10 `void gazebo::rendering::RenderEngine::RemoveScene (const std::string & _name)`

Remove a scene.

Parameters

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

10.125.4 Member Data Documentation

10.125.4.1 `void* gazebo::rendering::RenderEngine::dummyContext` [protected]

GLX context used to render the scenes.Used for gui-less operation.

10.125.4.2 `void* gazebo::rendering::RenderEngine::dummyDisplay` [protected]

Pointer to the dummy display.Used for gui-less operation.

10.125.4.3 `uint64_t gazebo::rendering::RenderEngine::dummyWindowId` [protected]

ID for a dummy window. Used for gui-less operation.

10.125.4.4 `Ogre::Root* gazebo::rendering::RenderEngine::root`

Pointer to the root scene node.

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

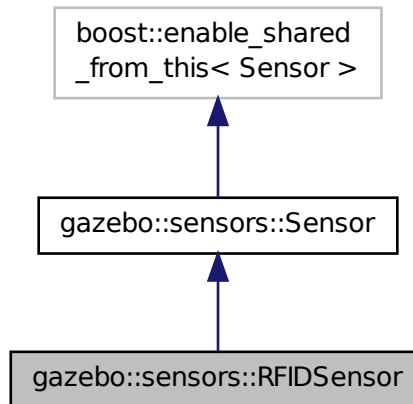
- **RenderEngine.hh**

10.126 gazebo::sensors::RFIDSensor Class Reference

Sensor (p. 683) 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

10.126.1 Detailed Description

Sensor (p. 683) class for RFID type of sensor.

10.126.2 Constructor & Destructor Documentation

10.126.2.1 gazebo::sensors::RFIDSensor::RFIDSensor ()

Constructor.

10.126.2.2 virtual gazebo::sensors::RFIDSensor::~~RFIDSensor () [virtual]

Destructor.

10.126.3 Member Function Documentation

10.126.3.1 void gazebo::sensors::RFIDSensor::AddTag (RFIDTag * *tag*)

10.126.3.2 virtual void gazebo::sensors::RFIDSensor::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 687).

10.126.3.3 virtual void gazebo::sensors::RFIDSensor::Init () [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 690).

10.126.3.4 virtual void gazebo::sensors::RFIDSensor::Load (const std::string & *_worldName*, sdf::ElementPtr *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

in	<i>_sdf</i>	SDF Sensor (p. 683) parameters.
in	<i>_worldName</i>	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 690).

10.126.3.5 virtual void gazebo::sensors::RFIDSensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	<code>_worldName</code>	Name of world to load from.
----	-------------------------	-----------------------------

Reimplemented from **gazebo::sensors::Sensor** (p. 690).

10.126.3.6 `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

in	<code>_force</code>	True if update is forced, false if not
----	---------------------	--

Reimplemented from **gazebo::sensors::Sensor** (p. 691).

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

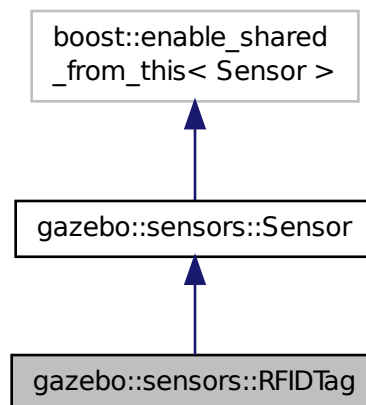
- **RFIDSensor.hh**

10.127 gazebo::sensors::RFIDTag Class Reference

RFIDTag (p. 646) to interact with RFIDTagSensors.

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

Inheritance diagram for gazebo::sensors::RFIDTag:



Public Member Functions

- **RFIDTag** ()
Constructor.
- virtual \sim **RFIDTag** ()
Destructor.
- virtual void **Fini** ()
Finalize the sensor.
- **math::Pose GetTagPose** () const
Returns pose of tag in world coordinate.
- virtual void **Init** ()
Initialize the sensor.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** &_sdf)
- 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

10.127.1 Detailed Description

RFIDTag (p. 646) to interact with RFIDTagSensors.

10.127.2 Constructor & Destructor Documentation

10.127.2.1 gazebo::sensors::RFIDTag::RFIDTag ()

Constructor.

10.127.2.2 virtual gazebo::sensors::RFIDTag::~~RFIDTag () [virtual]

Destructor.

10.127.3 Member Function Documentation

10.127.3.1 virtual void gazebo::sensors::RFIDTag::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 687).

10.127.3.2 `math::Pose gazebo::sensors::RFIDTag::GetTagPose () const [inline]`

Returns pose of tag in world coordinate.

Returns

Pose of object.

References `gazebo::physics::Entity::GetWorldPose()`.

10.127.3.3 `virtual void gazebo::sensors::RFIDTag::Init () [virtual]`

Initialize the sensor.

Reimplemented from `gazebo::sensors::Sensor` (p. 690).

10.127.3.4 `virtual void gazebo::sensors::RFIDTag::Load (const std::string & _worldName, sdf::ElementPtr & _sdf) [virtual]`

10.127.3.5 `virtual void gazebo::sensors::RFIDTag::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. 690).

10.127.3.6 `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

<code>in</code>	<code>_force</code>	True if update is forced, false if not
-----------------	---------------------	--

Reimplemented from `gazebo::sensors::Sensor` (p. 691).

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

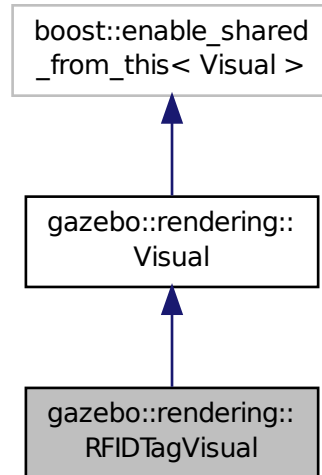
- `RFIDTag.hh`

10.128 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

10.128.1 Detailed Description

Visualization for RFID tags sensor.

10.128.2 Constructor & Destructor Documentation

10.128.2.1 gazebo::rendering::RFIDTagVisual::RFIDTagVisual (const std::string & *_name*, **VisualPtr** *_vis*, const std::string & *_topicName*)

Constructor.

Parameters

in	<i>_name</i>	Name of the visual.
in	<i>_vis</i>	Parent visual.
in	<i>_topicName</i>	Name of the topic that publishes RFID data.

See Also

sensors::RFIDSensor (p. 644)

10.128.2.2 virtual gazebo::rendering::RFIDTagVisual::~~RFIDTagVisual () [virtual]

Destructor.

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

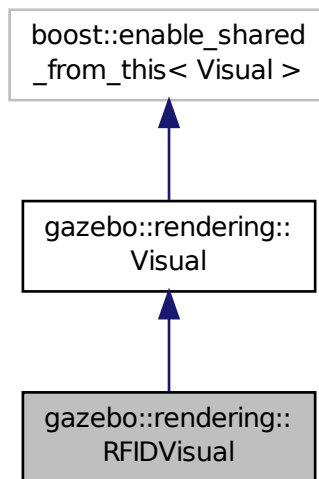
- **RFIDTagVisual.hh**

10.129 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

10.129.1 Detailed Description

Visualization for RFID sensor.

10.129.2 Constructor & Destructor Documentation

10.129.2.1 `gazebo::rendering::RFIDVisual::RFIDVisual (const std::string & _name, VisualPtr _vis, const std::string & _topicName)`

Constructor.

Parameters

<code>in</code>	<code><i>_name</i></code>	Name of the Visual (p. 868).
<code>in</code>	<code><i>_vis</i></code>	Parent Visual (p. 868).
<code>in</code>	<code><i>_topicName</i></code>	Name of the topic which publishes RFID data.

10.129.2.2 `virtual gazebo::rendering::RFIDVisual::~~RFIDVisual () [virtual]`

Destructor.

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

- **RFIDVisual.hh**

10.130 Road Class Reference

Used to render a strip of road.

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

10.130.1 Detailed Description

Used to render a strip of road.

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

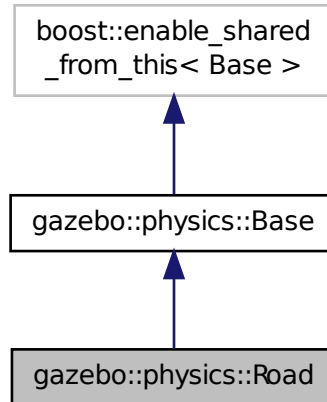
- **Road2d.hh**

10.131 gazebo::physics::Road Class Reference

for building a **Road** (p. 651) 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

10.131.1 Detailed Description

for building a **Road** (p. 651) from SDF

10.131.2 Constructor & Destructor Documentation

10.131.2.1 gazebo::physics::Road::Road (**BasePtr** _parent) [explicit]

Constructor.

Parameters

in	_parent	Parent of this road object.
----	---------	-----------------------------

10.131.2.2 virtual gazebo::physics::Road::~~Road () [virtual]

Destructor.

10.131.3 Member Function Documentation

10.131.3.1 virtual void gazebo::physics::Road::Init () [virtual]

Initialize the road.

Reimplemented from **gazebo::physics::Base** (p. 143).

10.131.3.2 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. 143).

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

- **Road.hh**

10.132 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.

10.132.1 Constructor & Destructor Documentation

10.132.1.1 gazebo::rendering::Road2d::Road2d ()

Constructor.

10.132.1.2 virtual gazebo::rendering::Road2d::~~Road2d () [virtual]

Destructor.

10.132.2 Member Function Documentation

10.132.2.1 void gazebo::rendering::Road2d::Load (VisualIPtr *_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**

10.133 gazebo::math::RotationSpline Class Reference

Spline (p. 740) 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

10.133.1 Detailed Description

Spline (p. 740) for rotations.

10.133.2 Constructor & Destructor Documentation

10.133.2.1 gazebo::math::RotationSpline::RotationSpline ()

Constructor. Sets the autoCalc to true.

10.133.2.2 gazebo::math::RotationSpline::~~RotationSpline ()

Destructor. Nothing is done.

10.133.3 Member Function Documentation

10.133.3.1 void gazebo::math::RotationSpline::AddPoint (const Quaternion & .p)

Adds a control point to the end of the spline.

Parameters

in	<code>_p</code>	control point
----	-----------------	---------------

10.133.3.2 void gazebo::math::RotationSpline::Clear ()

Clears all the points in the spline.

10.133.3.3 unsigned int gazebo::math::RotationSpline::GetNumPoints () const

Gets the number of control points in the spline.

Returns

the count

10.133.3.4 const Quaternion& gazebo::math::RotationSpline::GetPoint (unsigned int _index) const

Gets the detail of one of the control points of the spline.

Parameters

in	<i>_index</i>	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)

10.133.3.5 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	<i>_t</i>	Parametric value.
in	<i>_useShortestPath</i>	Defines if rotation should take the shortest possible path

Returns

the rotation

10.133.3.6 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

10.133.3.7 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.

10.133.3.8 void gazebo::math::RotationSpline::SetAutoCalculate (bool *_autoCalc*)

Tells the spline whether it should automatically calculate tangents on demand as points are added.

Remarks

The spline calculates tangents at each point automatically based on the input points. Normally it does this every time a point changes. However, if you have a lot of points to add in one go, you probably don't want to incur this overhead and would prefer to defer the calculation until you are finished setting all the points. You can do this by calling this method with a parameter of 'false'. Just remember to manually call the `recalcTangents` method when you are done.

Parameters

in	<i>_autoCalc</i>	If true, tangents are calculated for you whenever a point changes. If false, you must call <code>recalcTangents</code> to recalculate them when it best suits.
----	------------------	--

10.133.3.9 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

10.133.4 Member Data Documentation**10.133.4.1 bool gazebo::math::RotationSpline::autoCalc** [protected]

Automatic recalculation of tangents when control points are updated.

10.133.4.2 std::vector<Quaternion> gazebo::math::RotationSpline::points [protected]

the control points

10.133.4.3 std::vector<Quaternion> gazebo::math::RotationSpline::tangents [protected]

the tangents

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

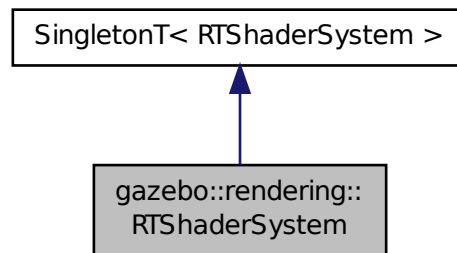
- [RotationSpline.hh](#)

10.134 gazebo::rendering::RTShaderSystem Class Reference

Implements **Ogre** (p. 106)'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. 106) 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

10.134.1 Detailed Description

Implements **Ogre** (p. 106)'s Run-Time Shader system.

This class allows Gazebo to generate per-pixel shaders for every material at run-time.

10.134.2 Member Enumeration Documentation

10.134.2.1 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.

10.134.3 Member Function Documentation

10.134.3.1 void gazebo::rendering::RTShaderSystem::AddScene (**ScenePtr** _scene)

Add a scene manager.

Parameters

in	<code>_scene</code>	The scene to process
----	---------------------	----------------------

10.134.3.2 `void gazebo::rendering::RTShaderSystem::ApplyShadows (ScenePtr _scene)`

Apply shadows to a scene.

Parameters

in	<code>_scene</code>	The scene to receive shadows.
----	---------------------	-------------------------------

10.134.3.3 `void gazebo::rendering::RTShaderSystem::AttachEntity (Visual * vis)`

Set an Ogre::Entity to use RT shaders.

Parameters

in	<code>_vis</code>	Visual (p. 868) that will use the RTShaderSystem (p. 658).
----	-------------------	--

10.134.3.4 `static void gazebo::rendering::RTShaderSystem::AttachViewport (Ogre::Viewport * _viewport, ScenePtr _scene)`
[static]

Set a viewport to use shaders.

Parameters

in	<code>_viewport</code>	The viewport to add.
in	<code>_scene</code>	The scene that the viewport uses.

10.134.3.5 `void gazebo::rendering::RTShaderSystem::Clear ()`

Clear the shader system.

10.134.3.6 `void gazebo::rendering::RTShaderSystem::DetachEntity (Visual * _vis)`

Remove and entity.

Parameters

in	<code>_vis</code>	Remove this visual.
----	-------------------	---------------------

10.134.3.7 `static void gazebo::rendering::RTShaderSystem::DetachViewport (Ogre::Viewport * _viewport, ScenePtr _scene)`
[static]

Set a viewport to not use shaders.

Parameters

in	<i>_viewport</i>	The viewport to remove.
in	<i>_scene</i>	The scene that the viewport uses.

10.134.3.8 void gazebo::rendering::RTShaderSystem::Fini ()

Finalize the shader system.

10.134.3.9 void gazebo::rendering::RTShaderSystem::GenerateShaders (Visual * *_vis*)

Generate shaders for an entity.

Parameters

in	<i>_vis</i>	The visual to generate shaders for.
----	-------------	-------------------------------------

10.134.3.10 Ogre::PSSMShadowCameraSetup* gazebo::rendering::RTShaderSystem::GetPSSMShadowCameraSetup () const

Get the **Ogre** (p. 106) PSSM Shadows camera setup.

Returns

The **Ogre** (p. 106) PSSM Shadows camera setup.

10.134.3.11 void gazebo::rendering::RTShaderSystem::Init ()

Init the run time shader system.

10.134.3.12 void gazebo::rendering::RTShaderSystem::RemoveScene (ScenePtr *_scene*)

Remove a scene.

Parameters

in	<i>The</i>	scene to remove
----	------------	-----------------

10.134.3.13 void gazebo::rendering::RTShaderSystem::RemoveShadows (ScenePtr *_scene*)

Remove shadows from a scene.

Parameters

in	<i>_scene</i>	The scene to remove shadows from.
----	---------------	-----------------------------------

10.134.3.14 `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.
-----------------	-------------------	--------------------------------------

10.134.3.15 `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:

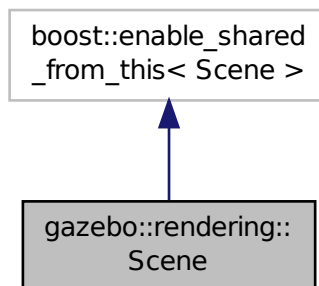
- **RTShaderSystem.hh**

10.135 gazebo::rendering::Scene Class Reference

Representation of an entire scene graph.

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

Inheritance diagram for gazebo::rendering::Scene:



Public Member Functions

- **Scene** (const std::string &_name, bool _enableVisualizations=false)
Constructor.
- virtual **~Scene** ()
Destructor.
- void **AddVisual** (**VisualPtr** _vis)
Add a visual to the scene.
- void **Clear** ()

Clear **rendering::Scene** (p. 662).

- **VisualPtr CloneVisual** (const std::string &_visualName, const std::string &_newName)
Clone a visual.
- **CameraPtr CreateCamera** (const std::string &_name, bool _autoRender=true)
Create a camera.
- **DepthCameraPtr CreateDepthCamera** (const std::string &_name, bool _autoRender=true)
Create depth camera.
- 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
Create laser that generates data from rendering.
- 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. 662) 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.
- **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 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.
- **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. 662).*
- **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 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 **SetTransparent** (bool _show)
Enable or disable transparency for all visuals.
- void **SetVisible** (const std::string &_name, bool _visible)
Hide or show a visual.
- 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.

10.135.1 Detailed Description

Representation of an entire scene graph.

Maintains all the Visuals, Lights, and Cameras for a World.

10.135.2 Constructor & Destructor Documentation

10.135.2.1 gazebo::rendering::Scene::Scene (const std::string & _name, bool *enableVisualizations* = false)

Constructor.

Parameters

in	<i>_name</i>	Name of the scene.
in	<i>_enable-Visualizations</i>	True to enable visualizations, this should be set to true for user interfaces, and false for sensor generation.

10.135.2.2 virtual gazebo::rendering::Scene::~Scene () [virtual]

Destructor.

10.135.3 Member Function Documentation

10.135.3.1 `void gazebo::rendering::Scene::AddVisual (VisualPtr _vis)`

Add a visual to the scene.

Parameters

in	_vis	Visual (p. 868) to add.
----	------	--------------------------------

10.135.3.2 `void gazebo::rendering::Scene::Clear ()`

Clear **rendering::Scene** (p. 662).

10.135.3.3 `VisualPtr gazebo::rendering::Scene::CloneVisual (const std::string & _visualName, const std::string & _newName)`

Clone a visual.

Parameters

in	_visualName	Name of the visual to clone.
in	_newName	New name of the visual.

Returns

Pointer to the cloned visual.

10.135.3.4 `CameraPtr gazebo::rendering::Scene::CreateCamera (const std::string & _name, bool _autoRender = true)`

Create a camera.

Parameters

in	_name	Name of the new camera.
in	_autoRender	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new camera.

10.135.3.5 `DepthCameraPtr gazebo::rendering::Scene::CreateDepthCamera (const std::string & _name, bool _autoRender = true)`

Create depth camera.

Parameters

in	_name	Name of the new camera.
in	_autoRender	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new camera.

10.135.3.6 `void gazebo::rendering::Scene::CreateGrid (uint32_t _cellCount, float _cellLength, float _lineWidth, const common::Color & _color)`

Create a square grid of cells.

Parameters

<code>in</code>	<code>_cellCount</code>	Number of grid cells in one direction.
<code>in</code>	<code>_cellLength</code>	Length of one grid cell.
<code>in</code>	<code>_lineWidth</code>	Width of the grid lines.
<code>in</code>	<code>_color</code>	Color of the grid lines.

10.135.3.7 `UserCameraPtr gazebo::rendering::Scene::CreateUserCamera (const std::string & _name)`

Create a user camera.

A (p. 109) user camera is one design for use with a GUI.

Parameters

<code>in</code>	<code>_name</code>	Name of the UserCamera (p. 814).
-----------------	--------------------	---

Returns

A (p. 109) pointer to the new **UserCamera** (p. 814).

10.135.3.8 `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.

10.135.3.9 `common::Color gazebo::rendering::Scene::GetAmbientColor () const`

Get the ambient color.

Returns

The scene's ambient color.

10.135.3.10 `common::Color gazebo::rendering::Scene::GetBackgroundColor () const`

Get the background color.

Returns

The background color.

10.135.3.11 `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. 668).

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.

10.135.3.12 `CameraPtr gazebo::rendering::Scene::GetCamera (const std::string & _name) const`

Get a camera by name.

Parameters

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

Returns

Pointer to the camera. Or NULL if the name is invalid.

10.135.3.13 `uint32_t gazebo::rendering::Scene::GetCameraCount () const`

Create laser that generates data from rendering.

Parameters

<code>in</code>	<code>_name</code>	Name of the new laser.
<code>in</code>	<code>_autoRender</code>	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new laser. Get the number of cameras in this scene
Number of lasers.

10.135.3.14 `bool gazebo::rendering::Scene::GetFirstContact (CameraPtr _camera, const math::Vector2i & _mousePos, math::Vector3 & _position)`

Get the world pos of a the first contact at a pixel location.

Parameters

in	<code>_camera</code>	Pointer to the camera.
in	<code>_mousePos</code>	2D position of the mouse in pixels.
out	<code>_position</code>	3D position of the first contact point.

Returns

True if a valid object was hit by the raycast.

10.135.3.15 `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. 669).

Parameters

in	<code>_index</code>	Index of the grid.
----	---------------------	--------------------

10.135.3.16 `uint32_t gazebo::rendering::Scene::GetGridCount () const`

Get the number of grids.

Returns

The number of grids.

10.135.3.17 `double gazebo::rendering::Scene::GetHeightBelowPoint (const math::Vector3 & _pt)`

Get the Z-value of the first object below the given point.

Parameters

in	<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.

10.135.3.18 `Heightmap* gazebo::rendering::Scene::GetHeightmap () const`

Get a pointer to the heightmap.

Returns

Pointer to the heightmap, NULL if no heightmap.

10.135.3.19 `uint32_t gazebo::rendering::Scene::GetId () const`

Get the scene ID.

Returns

The ID of the scene.

10.135.3.20 `std::string gazebo::rendering::Scene::GetIdString () const`

Get the scene Id as a string.

Returns

The ID as a string.

10.135.3.21 `bool gazebo::rendering::Scene::GetInitialized () const`

Return true if the **Scene** (p. 662) has been initialized.

10.135.3.22 `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.

10.135.3.23 `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. 671).

Parameters

<code>in</code>	<code>_index</code>	Index of the light.
-----------------	---------------------	---------------------

Returns

Pointer to the **Light** (p. 405) or NULL if index was invalid.

10.135.3.24 `uint32_t gazebo::rendering::Scene::GetLightCount () const`

Get the count of the lights.

Returns

The number of lights.

10.135.3.25 `Ogre::SceneManager* gazebo::rendering::Scene::GetManager () const`

Get the OGRE scene manager.

Returns

Pointer to the **Ogre** (p. 106) SceneManager.

10.135.3.26 `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.

10.135.3.27 `std::string gazebo::rendering::Scene::GetName () const`

Get the name of the scene.

Returns

Name of the scene.

10.135.3.28 `VisualPtr gazebo::rendering::Scene::GetSelectedVisual () const`

Get the currently selected visual.

Returns

Pointer to the currently selected visual, or NULL if nothing is selected.

10.135.3.29 `bool gazebo::rendering::Scene::GetShadowsEnabled () const`

Get whether shadows are on or off.

Returns

True if shadows are enabled.

10.135.3.30 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. 672).

Parameters

<i>in</i>	<i>_index</i>	Index of the UserCamera (p. 814) to get.
-----------	---------------	---

Returns

Pointer to the **UserCamera** (p. 814), or NULL if the index was invalid.

10.135.3.31 uint32_t gazebo::rendering::Scene::GetUserCameraCount () const

Get the number of user cameras in this scene.

Returns

The number of user cameras.

10.135.3.32 VisualPtr gazebo::rendering::Scene::GetVisual (const std::string & _name) const

Get a visual by name.

10.135.3.33 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

<i>in</i>	<i>_camera</i>	The ogre camera, used to do mouse picking
<i>in</i>	<i>_mousePos</i>	The position of the mouse in screen coordinates
<i>out</i>	<i>_mod</i>	Used for object manipulation

Returns

The selected entity, or NULL

10.135.3.34 VisualPtr gazebo::rendering::Scene::GetVisualAt (CameraPtr _camera, const math::Vector2i & _mousePos)

Get a visual at a mouse position.

Parameters

in	<code>_camera</code>	Pointer to the camera used to project the mouse position.
in	<code>_mousePos</code>	The 2d position of the mouse in pixels.

Returns

Pointer to the visual, NULL if none found.

10.135.3.35 `VisualPtr gazebo::rendering::Scene::GetVisualBelow (const std::string & _visualName)`

Get the closest visual below a given visual.

Parameters

in	<code>_visualName</code>	Name of the visual to search below.
----	--------------------------	-------------------------------------

Returns

Pointer to the visual below, or NULL if no visual.

10.135.3.36 `void gazebo::rendering::Scene::GetVisualsBelowPoint (const math::Vector3 & _pt, std::vector< VisualPtr > & _visuals)`

Get a visual directly below a point.

Parameters

in	<code>_pt</code>	3D point to get the visual below.
out	<code>_visuals</code>	The visuals below the point order in proximity.

10.135.3.37 `VisualPtr gazebo::rendering::Scene::GetWorldVisual () const`

Get the top level world visual.

Returns

Pointer to the world visual.

10.135.3.38 `void gazebo::rendering::Scene::Init ()`

Init `rendering::Scene` (p. 662).

10.135.3.39 `void gazebo::rendering::Scene::Load (sdf::ElementPtr _scene)`

Load the scene from a set of parameters.

Parameters

in	<code>_scene</code>	SDF scene element to load.
----	---------------------	----------------------------

10.135.3.40 `void gazebo::rendering::Scene::Load ()`

Load the scene with default parameters.

10.135.3.41 `void gazebo::rendering::Scene::PreRender ()`

Process all received messages.

10.135.3.42 `void gazebo::rendering::Scene::PrintSceneGraph ()`

Print the scene graph to `std_out`.

10.135.3.43 `void gazebo::rendering::Scene::RemoveVisual (VisualPtr _vis)`

Remove a visual from the scene.

Parameters

in	<code>_vis</code>	Visual (p. 868) to remove.
----	-------------------	-----------------------------------

10.135.3.44 `void gazebo::rendering::Scene::SelectVisual (const std::string & _name, const std::string & _mode)`

Select a visual by name.

Parameters

in	<code>_name</code>	Name of the visual to select.
in	<code>_mode</code>	Selection mode (normal, or move).

10.135.3.45 `void gazebo::rendering::Scene::SetAmbientColor (const common::Color & _color)`

Set the ambient color.

Parameters

in	<code>_color</code>	The ambient color to use.
----	---------------------	---------------------------

10.135.3.46 `void gazebo::rendering::Scene::SetBackgroundColor (const common::Color & _color)`

Set the background color.

Parameters

in	<code>_color</code>	The background color.
----	---------------------	-----------------------

10.135.3.47 `void gazebo::rendering::Scene::SetFog (const std::string & _type, const common::Color & _color, double _density, double _start, double _end)`

Set the fog parameters.

Parameters

in	<code>_type</code>	Type of fog: "linear", "exp", or "exp2".
in	<code>_color</code>	Color of the fog.
in	<code>_density</code>	Fog density.
in	<code>_start</code>	Distance from camera to start the fog.
in	<code>_end</code>	Distance from camera at which the fog is at max density.

10.135.3.48 `void gazebo::rendering::Scene::SetGrid (bool _enabled)`

Set the grid on or off.

Parameters

in	<code>_enabled</code>	Set to true to turn on the grid
----	-----------------------	---------------------------------

10.135.3.49 `void gazebo::rendering::Scene::SetShadowsEnabled (bool _value)`

Set whether shadows are on or off.

Parameters

in	<code>_value</code>	True to enable shadows, False to disable
----	---------------------	--

10.135.3.50 `void gazebo::rendering::Scene::SetTransparent (bool _show)`

Enable or disable transparency for all visuals.

Parameters

in	<code>_show</code>	True to enable transparency for all visuals.
----	--------------------	--

10.135.3.51 `void gazebo::rendering::Scene::SetVisible (const std::string & _name, bool _visible)`

Hide or show a visual.

Parameters

in	<code>_name</code>	Name of the visual to change.
in	<code>_visible</code>	True to make visual visible, False to make it invisible.

10.135.3.52 `void gazebo::rendering::Scene::ShowCollisions (bool _show)`

Enable or disable collision visualization.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to enable collision visualization.
-----------------	---------------------------	---

10.135.3.53 `void gazebo::rendering::Scene::ShowCOMs (bool _show)`

Enable or disable center of mass visualization.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to enable center of mass visualization.
-----------------	---------------------------	--

10.135.3.54 `void gazebo::rendering::Scene::ShowContacts (bool _show)`

Enable or disable contact visualization.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to enable contact visualization.
-----------------	---------------------------	---------------------------------------

10.135.3.55 `void gazebo::rendering::Scene::ShowJoints (bool _show)`

Enable or disable joint visualization.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to enable joint visualization.
-----------------	---------------------------	-------------------------------------

10.135.3.56 `void gazebo::rendering::Scene::SnapVisualToNearestBelow (const std::string & _visualName)`

Move the visual to be ontop of the nearest visual below it.

Parameters

<code>in</code>	<code><i>_visualName</i></code>	Name of the visual to move.
-----------------	---------------------------------	-----------------------------

10.135.3.57 `std::string gazebo::rendering::Scene::StripSceneName (const std::string & _name) const`

Remove the name of scene from a string.

Parameters

<code>in</code>	<code><i>_name</i></code>	Name to string the scene name from.
-----------------	---------------------------	-------------------------------------

Returns

The stripped name.

10.135.4 Member Data Documentation**10.135.4.1 SkyX::SkyX* gazebo::rendering::Scene::skyx**

Pointer to the sky.

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

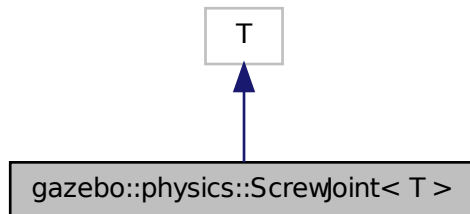
- **Scene.hh**

10.136 gazebo::physics::ScrewJoint< T > Class Template Reference

A (p. 109) 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. 677).*

- 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.

10.136.1 Detailed Description

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

A (p. 109) screw joint, which has both prismatic and rotational DOFs.

10.136.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	<i>_parent</i>	Parent of the joint.
----	----------------	----------------------

References gazebo::physics::Base::SCREW_JOINT.

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

Destructor.

10.136.3 Member Function Documentation

10.136.3.1 `template<class T > math::Vector3 gazebo::physics::ScrewJoint< T >::GetAnchor (int _index) const [virtual]`

Get the anchor.

Parameters

in	<i>_index</i>	Index of the axis. Not Used.
----	---------------	------------------------------

Returns

Anchor for the joint.

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

10.136.3.3 `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

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

`_threadPitch` Thread pitch value.

10.136.3.4 `template<class T > virtual void gazebo::physics::ScrewJoint< T >::Load (sdf::ElementPtr _sdf)`
`[inline],[virtual]`

Load a **ScrewJoint** (p. 677).

Parameters

<code>in</code>	<code><i>_sdf</i></code>	SDF value to load from
-----------------	--------------------------	------------------------

References `sdf::Element::GetElement()`, `sdf::Element::GetValueDouble()`, `sdf::Element::GetValueVector3()`, `gzerr`, `sdf::Element::HasElement()`, and `gazebo::physics::ScrewJoint< T >::threadPitch`.

10.136.3.5 `template<class T > void gazebo::physics::ScrewJoint< T >::SetAnchor (int _index, const math::Vector3 & _anchor)`
`[virtual]`

Set the anchor.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis. Not Used.
<code>in</code>	<code><i>_anchor</i></code>	Anchor value for the joint.

10.136.3.6 `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	<code>_index</code>	Index of the axis.
in	<code>_threadPitch</code>	Thread pitch value.

10.136.4 Member Data Documentation

10.136.4.1 `template<class T > math::Vector3 gazebo::physics::ScrewJoint< T >::fakeAnchor` [protected]

The anchor value is not used internally.

10.136.4.2 `template<class T > double gazebo::physics::ScrewJoint< T >::threadPitch` [protected]

Pitch of the thread.

Referenced by `gazebo::physics::ScrewJoint< T >::Load()`.

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

- **ScrewJoint.hh**

10.137 sdf::SDF Class Reference

Base **SDF** (p. 680) class.

```
#include <SDF.hh>
```

Public Member Functions

- **SDF** ()
- void **PrintDescription** ()
- void **PrintDoc** ()
- void **PrintValues** ()
- void **PrintWiki** ()
- void **SetFromString** (const std::string &_sdfData)
*Set **SDF** (p. 680) values from a string.*
- std::string **Tostring** () const
- void **Write** (const std::string &_filename)

Public Attributes

- **ElementPtr** root

Static Public Attributes

- static std::string **version**

10.137.1 Detailed Description

Base **SDF** (p. 680) class.

10.137.2 Constructor & Destructor Documentation

10.137.2.1 `sdf::SDF::SDF ()`

10.137.3 Member Function Documentation

10.137.3.1 `void sdf::SDF::PrintDescription ()`

10.137.3.2 `void sdf::SDF::PrintDoc ()`

10.137.3.3 `void sdf::SDF::PrintValues ()`

10.137.3.4 `void sdf::SDF::PrintWiki ()`

10.137.3.5 `void sdf::SDF::SetFromString (const std::string & _sdfData)`

Set **SDF** (p. 680) values from a string.

10.137.3.6 `std::string sdf::SDF::ToString () const`

10.137.3.7 `void sdf::SDF::Write (const std::string & _filename)`

10.137.4 Member Data Documentation

10.137.4.1 `ElementPtr sdf::SDF::root`

10.137.4.2 `std::string sdf::SDF::version [static]`

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

- **SDF.hh**

10.138 gazebo::rendering::SelectionObj Class Reference

A (p. 109) graphical selection object.

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

Public Member Functions

- **SelectionObj** (**Scene** *_scene)
Constructor.
- virtual **~SelectionObj** ()
Destructor.
- void **Attach** (**VisualPtr** _visual)
Set the position of the node.
- void **Clear** ()
Clear the `rendering::SelectionObj` (p. 681) object.
- `std::string` **GetVisualName** () const

Get the name of the visual the selection obj is attached to.

- void **Init** ()

Initialize the **rendering::SelectionObj** (p. 681) object.

- bool **IsActive** () const

Return true if the user is move the selection obj.

- void **SetActive** (bool _active)

Set true if the user is moving the selection obj.

- void **SetHighlight** (const std::string &_mod)

Highlight the selection object based on a modifier.

10.138.1 Detailed Description

A (p. 109) graphical selection object.

Used to draw a visual around a selected object.

10.138.2 Constructor & Destructor Documentation

10.138.2.1 gazebo::rendering::SelectionObj::SelectionObj (Scene * _scene)

Constructor.

Parameters

in	<i>_scene</i>	Scene (p. 662) to use.
----	---------------	-------------------------------

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

Destructor.

10.138.3 Member Function Documentation

10.138.3.1 void gazebo::rendering::SelectionObj::Attach (VisualPtr _visual)

Set the position of the node.

Parameters

in	<i>This</i>	draws the selection object around the passed in visual.
----	-------------	---

10.138.3.2 void gazebo::rendering::SelectionObj::Clear ()

Clear the **rendering::SelectionObj** (p. 681) object.

10.138.3.3 std::string gazebo::rendering::SelectionObj::GetVisualName () const

Get the name of the visual the selection obj is attached to.

Returns

Name of the selected visual.

10.138.3.4 `void gazebo::rendering::SelectionObj::Init ()`

Initialize the **rendering::SelectionObj** (p. 681) object.

10.138.3.5 `bool gazebo::rendering::SelectionObj::IsActive () const`

Return true if the user is move the selection obj.

Returns

True if something is selected.

10.138.3.6 `void gazebo::rendering::SelectionObj::SetActive (bool _active)`

Set true if the user is moving the selection obj.

Parameters

<code>in</code>	<code><i>_active</i></code>	True if the user is interacting with the selection object.
-----------------	-----------------------------	--

10.138.3.7 `void gazebo::rendering::SelectionObj::SetHighlight (const std::string & _mod)`

Highlight the selection object based on a modifier.

Parameters

<code>in</code>	<code><i>_mod</i></code>	Modifier used when highlighting the selection object.
-----------------	--------------------------	---

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

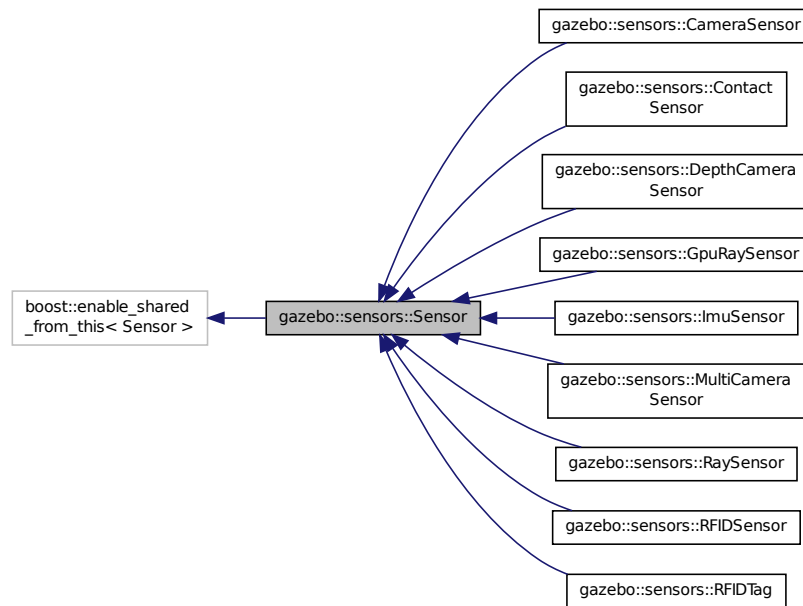
- **SelectionObj.hh**

10.139 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.
- **common::Time GetLastMeasurementTime** ()
Return last measurement time.
- **common::Time GetLastUpdateTime** ()
Return last update time.
- std::string **GetName** () const
Get name.

- 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)
Set the parent of the sensor.
- 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.
- **transport::NodePtr node**
Node for communication.
- **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.
- **sdf::ElementPtr sdf**
Pointer the the SDF element for the sensor.
- **common::Time updatePeriod**
*Desired time between updates, set indirectly by **Sensor::SetUpdateRate** (p. 691).*
- **gazebo::physics::WorldPtr world**
Pointer to the world.

10.139.1 Detailed Description

Base class for sensors.

10.139.2 Constructor & Destructor Documentation

10.139.2.1 gazebo::sensors::Sensor::Sensor (SensorCategory _cat) [explicit]

Constructor.

Parameters

in	<code>_class</code>
----	---------------------

10.139.2.2 virtual gazebo::sensors::Sensor::~~Sensor () [virtual]

Destructor.

10.139.3 Member Function Documentation

10.139.3.1 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	<code>_subscriber</code>	Callback that receives the signal.
----	--------------------------	------------------------------------

Returns

A (p. 109) pointer to the connection. This must be kept in scope.

See Also

Sensor::DisconnectUpdated (p. 687)

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

10.139.3.2 void gazebo::sensors::Sensor::DisconnectUpdated (event::ConnectionPtr & _c) [inline]

Disconnect from a the updated signal.

Parameters

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

See Also

Sensor::ConnectUpdated (p. 686)

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

10.139.3.3 void gazebo::sensors::Sensor::FillMsg (msgs::Sensor & _msg)

fills a msgs::Sensor message.

Parameters

out	<code>_msg</code>	Message to fill.
-----	-------------------	------------------

10.139.3.4 virtual void gazebo::sensors::Sensor::Fini () [virtual]

Finalize the sensor.

Reimplemented in **gazebo::sensors::MultiCameraSensor** (p. 515), **gazebo::sensors::CameraSensor** (p. 186), **gazebo::sensors::GpuRaySensor** (p. 328), **gazebo::sensors::ContactSensor** (p. 239), **gazebo::sensors::DepthCameraSensor** (p. 255), **gazebo::sensors::RFIDSensor** (p. 645), **gazebo::sensors::RaySensor** (p. 629), **gazebo::sensors::RFIDTag** (p. 647), and **gazebo::sensors::ImuSensor** (p. 363).

10.139.3.5 **SensorCategory** gazebo::sensors::Sensor::GetCategory () const

Get the category of the sensor.

Returns

The category of the sensor.

See Also

SensorCategory (p. 102)

10.139.3.6 common::Time gazebo::sensors::Sensor::GetLastMeasurementTime ()

Return last measurement time.

Returns

Time of last measurement.

10.139.3.7 common::Time gazebo::sensors::Sensor::GetLastUpdateTime ()

Return last update time.

Returns

Time of last update.

10.139.3.8 std::string gazebo::sensors::Sensor::GetName () const

Get name.

Returns

Name of sensor.

10.139.3.9 std::string gazebo::sensors::Sensor::GetParentName () const

Returns the name of the sensor parent.

The parent name is set by **Sensor::SetParent** (p. 691).

Returns

Name of Parent.

10.139.3.10 virtual math::Pose gazebo::sensors::Sensor::GetPose () const [virtual]

Get the current pose.

Returns

Current pose of the sensor.

10.139.3.11 `std::string gazebo::sensors::Sensor::GetScopedName () const`

Get fully scoped name of the sensor.

Returns

world_name::parent_name::sensor_name.

10.139.3.12 `virtual std::string gazebo::sensors::Sensor::GetTopic () const` [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented in **gazebo::sensors::RaySensor** (p. 632), **gazebo::sensors::CameraSensor** (p. 187), and **gazebo::sensors::MultiCameraSensor** (p. 517).

10.139.3.13 `std::string gazebo::sensors::Sensor::GetType () const`

Get sensor type.

Returns

Type of sensor.

10.139.3.14 `double gazebo::sensors::Sensor::GetUpdateRate ()`

Get the update rate of the sensor.

Returns

_hz update rate of sensor. Returns 0 if unthrottled.

10.139.3.15 `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.

10.139.3.16 `std::string gazebo::sensors::Sensor::GetWorldName () const`

Returns the name of the world the sensor is in.

Returns

Name of the world.

10.139.3.17 `virtual void gazebo::sensors::Sensor::Init () [virtual]`

Initialize the sensor.

Reimplemented in `gazebo::sensors::GpuRaySensor` (p. 333), `gazebo::sensors::ContactSensor` (p. 241), `gazebo::sensors::CameraSensor` (p. 187), `gazebo::sensors::DepthCameraSensor` (p. 256), `gazebo::sensors::RFIDSensor` (p. 645), `gazebo::sensors::RaySensor` (p. 633), `gazebo::sensors::RFIDTag` (p. 648), `gazebo::sensors::MultiCameraSensor` (p. 517), and `gazebo::sensors::ImuSensor` (p. 364).

10.139.3.18 `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::RaySensor` (p. 633), `gazebo::sensors::ContactSensor` (p. 241), `gazebo::sensors::CameraSensor` (p. 188), and `gazebo::sensors::MultiCameraSensor` (p. 517).

10.139.3.19 `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. 683) parameters.
in	<code>_worldName</code>	Name of world to load from.

Reimplemented in `gazebo::sensors::ContactSensor` (p. 241), `gazebo::sensors::CameraSensor` (p. 188), `gazebo::sensors::RFIDSensor` (p. 645), and `gazebo::sensors::ImuSensor` (p. 364).

10.139.3.20 `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. 333), `gazebo::sensors::ContactSensor` (p. 241), `gazebo::sensors::CameraSensor` (p. 188), `gazebo::sensors::DepthCameraSensor` (p. 256), `gazebo::sensors::RFIDSensor` (p. 645), `gazebo::sensors::RaySensor` (p. 633), `gazebo::sensors::RFIDTag` (p. 648), `gazebo::sensors::MultiCameraSensor` (p. 517), and `gazebo::sensors::ImuSensor` (p. 364).

10.139.3.21 `void gazebo::sensors::Sensor::ResetLastUpdateTime ()`

Reset the lastUpdateTime to zero.

10.139.3.22 `virtual void gazebo::sensors::Sensor::SetActive (bool _value) [virtual]`

Set whether the sensor is active or not.

Parameters

in	<i>_value</i>	True if active, false if not.
----	---------------	-------------------------------

Reimplemented in `gazebo::sensors::DepthCameraSensor` (p. 256).

10.139.3.23 `virtual void gazebo::sensors::Sensor::SetParent (const std::string & _name) [virtual]`

Set the parent of the sensor.

Parameters

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

Reimplemented in `gazebo::sensors::CameraSensor` (p. 188), and `gazebo::sensors::DepthCameraSensor` (p. 257).

10.139.3.24 `void gazebo::sensors::Sensor::SetUpdateRate (double _hz)`

Set the update rate of the sensor.

Parameters

in	<i>_hz</i>	update rate of sensor.
----	------------	------------------------

10.139.3.25 `void gazebo::sensors::Sensor::Update (bool _force)`

Update the sensor.

Parameters

in	<i>_force</i>	True to force update, false otherwise.
----	---------------	--

10.139.3.26 `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	<i>_force</i>	True if update is forced, false if not
----	---------------	--

Reimplemented in `gazebo::sensors::MultiCameraSensor` (p. 518), `gazebo::sensors::CameraSensor` (p. 189), `gazebo::sensors::GpuRaySensor` (p. 334), `gazebo::sensors::ContactSensor` (p. 242), `gazebo::sensors::DepthCameraSensor` (p. 257), `gazebo::sensors::RFIDSensor` (p. 646), `gazebo::sensors::RaySensor` (p. 633), `gazebo::sensors::RFIDTag` (p. 648), and `gazebo::sensors::ImuSensor` (p. 364).

10.139.4 Member Data Documentation

10.139.4.1 `bool gazebo::sensors::Sensor::active` [protected]

True if sensor generation is active.

10.139.4.2 `std::vector<event::ConnectionPtr> gazebo::sensors::Sensor::connections` [protected]

All event connections.

10.139.4.3 `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`.

10.139.4.4 `common::Time gazebo::sensors::Sensor::lastUpdateTime` [protected]

Time of the last update.

10.139.4.5 `transport::NodePtr gazebo::sensors::Sensor::node` [protected]

Node for communication.

10.139.4.6 `std::string gazebo::sensors::Sensor::parentName` [protected]

Name of the parent.

10.139.4.7 `std::vector<SensorPluginPtr> gazebo::sensors::Sensor::plugins` [protected]

All the plugins for the sensor.

10.139.4.8 `math::Pose gazebo::sensors::Sensor::pose` [protected]

Pose of the sensor.

10.139.4.9 `transport::SubscriberPtr gazebo::sensors::Sensor::poseSub` [protected]

Subscribe to pose updates.

10.139.4.10 `sdf::ElementPtr gazebo::sensors::Sensor::sdf` [protected]

Pointer to the SDF element for the sensor.

10.139.4.11 `common::Time gazebo::sensors::Sensor::updatePeriod` [protected]

Desired time between updates, set indirectly by `Sensor::SetUpdateRate` (p. 691).

10.139.4.12 `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`

10.140 SensorFactor Class Reference

The sensor factory; the class is just for namespacing purposes.

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

10.140.1 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`

10.141 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).

10.141.1 Member Function Documentation

10.141.1.1 `static void gazebo::sensors::SensorFactory::GetSensorTypes (std::vector< std::string > & _types) [static]`

Get all the sensor types.

Parameters

<i>_types</i>	Vector of strings of the sensor types, populated by function
---------------	--

10.141.1.2 `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	<i>_className</i>	Name of sensor class
----	-------------------	----------------------

Returns

Pointer to **Sensor** (p. 683)

10.141.1.3 `static void gazebo::sensors::SensorFactory::RegisterAll () [static]`

Register all known sensors.

- **sensors::CameraSensor** (p. 185)
- **sensors::DepthCameraSensor** (p. 254)
- **sensors::GpuRaySensor** (p. 324)
- **sensors::RaySensor** (p. 627)
- **sensors::ContactSensor** (p. 237)
- **sensors::RFIDSensor** (p. 644)
- **sensors::RFIDTag** (p. 646)

10.141.1.4 `static void gazebo::sensors::SensorFactory::RegisterSensor (const std::string & _className, SensorFactoryFn _factoryfn) [static]`

Register a sensor class (called by sensor registration function).

Parameters

in	<i>_className</i>	Name of class of sensor to register.
in	<i>_factoryfn</i>	Function handle for registration.

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

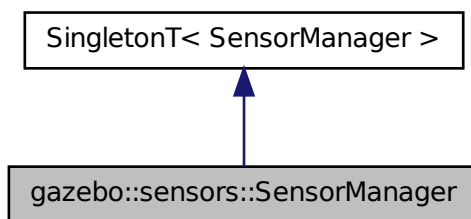
- **SensorFactory.hh**

10.142 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)
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` **Run** () **GAZEBO_DEPRECATED**

Deprecated.

- 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

10.142.1 Detailed Description

Class to manage and update all sensors.

10.142.2 Member Function Documentation

10.142.2.1 `std::string gazebo::sensors::SensorManager::CreateSensor (sdf::ElementPtr _elem, const std::string & _worldName, const std::string & _parentName)`

Add a sensor from an SDF element.

This function will also Load and Init the sensor.

Parameters

<code>in</code>	<code>_elem</code>	The SDF element that describes the sensor
<code>in</code>	<code>_worldName</code>	Name of the world in which to create the sensor
<code>in</code>	<code>_parentName</code>	The name of the parent link which the sensor is attached to.

Returns

The name of the sensor

10.142.2.2 `void gazebo::sensors::SensorManager::Fini ()`

Finalize all the sensors.

10.142.2.3 `SensorPtr gazebo::sensors::SensorManager::GetSensor (const std::string & _name) const`

Get a sensor.

Parameters

<code>in</code>	<code>_name</code>	The name of a sensor to find.
-----------------	--------------------	-------------------------------

Returns

A (p. 109) pointer to the sensor. NULL if not found.

10.142.2.4 **Sensor_V** gazebo::sensors::SensorManager::GetSensors () const

Get all the sensors.

Returns

Vector of all the sensors.

10.142.2.5 void gazebo::sensors::SensorManager::GetSensorTypes (std::vector< std::string > & *_types*) const

Get all the sensor types.

Parameters

out	<i>All</i>	the sensor types.
-----	------------	-------------------

10.142.2.6 void gazebo::sensors::SensorManager::Init ()

Init all the sensors.

10.142.2.7 void gazebo::sensors::SensorManager::RemoveSensor (const std::string & *_name*)

Remove a sensor.

Parameters

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

10.142.2.8 void gazebo::sensors::SensorManager::RemoveSensors ()

Remove all sensors.

10.142.2.9 void gazebo::sensors::SensorManager::ResetLastUpdateTimes ()

Reset last update times in all sensors.

10.142.2.10 void gazebo::sensors::SensorManager::Run ()

Deprecated.

See Also

RunThreads (p. 698)

10.142.2.11 `void gazebo::sensors::SensorManager::RunThreads ()`

Run sensor updates in separate threads.

This will only run non-image based sensor updates.

10.142.2.12 `bool gazebo::sensors::SensorManager::SensorsInitialized ()`

True if `SensorManager::initSensors` queue is empty i.e.

all sensors managed by **SensorManager** (p. 695) have been initialized

10.142.2.13 `void gazebo::sensors::SensorManager::Stop ()`

Stop the run thread.

10.142.2.14 `void gazebo::sensors::SensorManager::Update (bool _force = false)`

Update all the sensors.

Checks to see if any sensor need to be initialized first, then updates all sensors once.

Parameters

<code>in</code>	<code><i>_force</i></code>	True force update, false if not
-----------------	----------------------------	---------------------------------

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

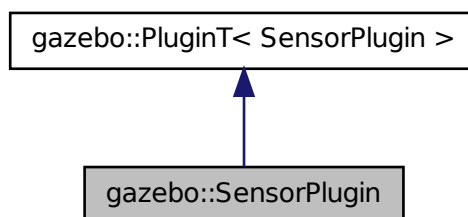
- **SensorManager.hh**

10.143 gazebo::SensorPlugin Class Reference

A (p. 109) 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

10.143.1 Detailed Description

A (p. 109) plugin with access to physics::Sensor.

See reference.

10.143.2 Constructor & Destructor Documentation

10.143.2.1 gazebo::SensorPlugin::SensorPlugin () [inline]

Constructor.

References gazebo::SENSOR_PLUGIN, and gazebo::PluginT< SensorPlugin >::type.

10.143.2.2 virtual gazebo::SensorPlugin::~~SensorPlugin () [inline],[virtual]

Destructor.

10.143.3 Member Function Documentation

10.143.3.1 virtual void gazebo::SensorPlugin::Init () [inline],[virtual]

Override this method for custom plugin initialization behavior.

10.143.3.2 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.

10.143.3.3 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:

- **common/Plugin.hh**

10.144 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)
- void **PrintUsage** ()
- void **Run** ()
- void **SetParams** (const **common::StrStr_M** ¶ms)
- void **Stop** ()

Public Attributes

- int **systemPluginsArgc**
- char ** **systemPluginsArgv**

10.144.1 Constructor & Destructor Documentation

10.144.1.1 gazebo::Server::Server ()

10.144.1.2 virtual gazebo::Server::~Server () [virtual]

10.144.2 Member Function Documentation

10.144.2.1 void gazebo::Server::Fini ()

10.144.2.2 bool gazebo::Server::GetInitialized () const

10.144.2.3 void gazebo::Server::Init ()

10.144.2.4 `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

<code>in</code>	<code>_filename</code>	Name of the world file to load.
<code>in</code>	<code>_physics</code>	Type of physics engine to use (ode bullet).

10.144.2.5 `bool gazebo::Server::LoadString (const std::string & _sdfString)`

10.144.2.6 `bool gazebo::Server::ParseArgs (int argc, char ** argv)`

10.144.2.7 `void gazebo::Server::PrintUsage ()`

10.144.2.8 `void gazebo::Server::Run ()`

10.144.2.9 `void gazebo::Server::SetParams (const common::StrStr_M & params)`

10.144.2.10 `void gazebo::Server::Stop ()`

10.144.3 Member Data Documentation

10.144.3.1 `int gazebo::Server::systemPluginsArgc`

10.144.3.2 `char** gazebo::Server::systemPluginsArgv`

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

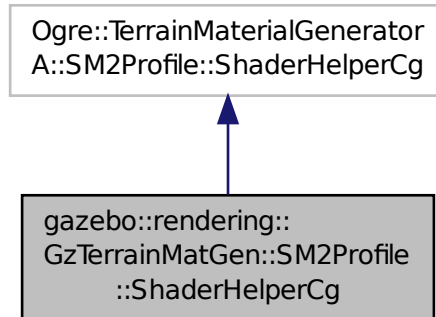
- **Server.hh**

10.145 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)

10.145.1 Detailed Description

Keeping the CG shader for reference.

Utility class to help with generating shaders for Cg / HLSL.

10.145.2 Member Function Documentation

- 10.145.2.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::defaultVpParams (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, const Ogre::HighLevelGpuProgramPtr & *_prog*)
[protected], [virtual]
- 10.145.2.2 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateFragmentProgram (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*)
[virtual]
- 10.145.2.3 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgram (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*)
[virtual]
- 10.145.2.4 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgramSource (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*)
[protected], [virtual]
- 10.145.2.5 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpDynamicShadows (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*)
[protected], [virtual]
- 10.145.2.6 virtual unsigned int gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpDynamicShadows-Params (unsigned int *_texCoordStart*, const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]
- 10.145.2.7 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpFooter (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*)
[protected], [virtual]
- 10.145.2.8 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:

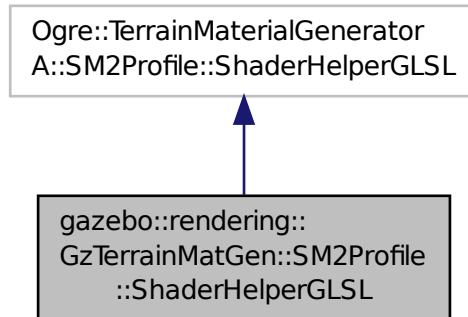
- **Heightmap.hh**

10.146 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)

10.146.1 Detailed Description

Utility class to help with generating shaders for GLSL.

10.146.2 Member Function Documentation

- 10.146.2.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::defaultVpParams (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, const Ogre::HighLevelGpuProgramPtr & .prog) [protected], [virtual]
- 10.146.2.2 void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadows (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & .outStream) [protected]
- 10.146.2.3 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadowsHelpers (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & .outStream) [protected], [virtual]
- 10.146.2.4 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]
- 10.146.2.5 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpFooter (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & .outStream) [protected], [virtual]
- 10.146.2.6 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpHeader (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & .outStream) [protected], [virtual]
- 10.146.2.7 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]
- 10.146.2.8 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFragmentProgram (const **SM2Profile** *_prof, const Ogre::Terrain *_terrain, TechniqueType _tt) [virtual]

- 10.146.2.9 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFragmentProgramSource (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.146.2.10 `virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVertexProgram (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt)` [virtual]
- 10.146.2.11 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVertexProgramSource (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.146.2.12 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpDynamicShadows (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.146.2.13 `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]
- 10.146.2.14 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpFooter (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.146.2.15 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpHeader (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream)` [protected],[virtual]
- 10.146.2.16 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::updateParams (const SM2Profile * _prof, const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain, bool _compositeMap)` [virtual]
- 10.146.2.17 `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:

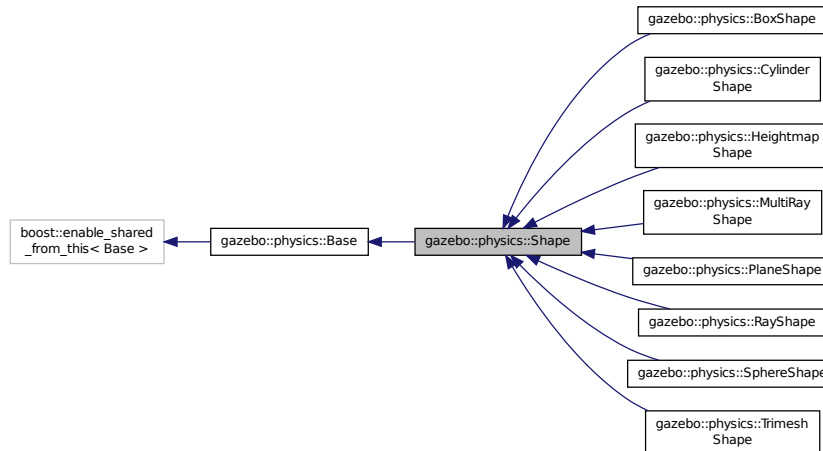
- Heightmap.hh

10.147 gazebo::physics::Shape Class Reference

Base (p. 135) class for all shapes.

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

Inheritance diagram for gazebo::physics::Shape:



Public Member Functions

- **Shape** (**CollisionPtr** _parent)
Constructor.
- virtual **~Shape** ()
Destructor.
- virtual void **FillMsg** (msgs::Geometry &_msg)=0
Fill in the values for a geometry message.
- virtual void **Init** ()=0
Initialize the shape.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)=0
Process a geometry message.

Protected Attributes

- **CollisionPtr** collisionParent
This shape's collision parent.

Additional Inherited Members

10.147.1 Detailed Description

Base (p. 135) class for all shapes.

10.147.2 Constructor & Destructor Documentation

10.147.2.1 `gazebo::physics::Shape::Shape (CollisionPtr _parent) [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Parent of the shape.
----	----------------------	----------------------

10.147.2.2 `virtual gazebo::physics::Shape::~~Shape () [virtual]`

Destructor.

10.147.3 Member Function Documentation

10.147.3.1 `virtual void gazebo::physics::Shape::FillMsg (msgs::Geometry & _msg) [pure virtual]`

Fill in the values for a geometry message.

Parameters

out	<code>_msg</code>	The geometry message to fill.
-----	-------------------	-------------------------------

Implemented in `gazebo::physics::MultiRayShape` (p.522), `gazebo::physics::RayShape` (p.636), `gazebo::physics::HeightmapShape` (p.350), `gazebo::physics::PlaneShape` (p.580), `gazebo::physics::TrimeshShape` (p.808), `gazebo::physics::CylinderShape` (p.248), `gazebo::physics::SphereShape` (p.739), and `gazebo::physics::BoxShape` (p.153).

10.147.3.2 `virtual void gazebo::physics::Shape::Init () [pure virtual]`

Initialize the shape.

Reimplemented from `gazebo::physics::Base` (p.143).

Implemented in `gazebo::physics::RayShape` (p.637), `gazebo::physics::HeightmapShape` (p.351), `gazebo::physics::TrimeshShape` (p.809), `gazebo::physics::MultiRayShape` (p.525), `gazebo::physics::PlaneShape` (p.580), `gazebo::physics::SphereShape` (p.739), `gazebo::physics::BoxShape` (p.153), and `gazebo::physics::CylinderShape` (p.249).

10.147.3.3 `virtual void gazebo::physics::Shape::ProcessMsg (const msgs::Geometry & _msg) [pure virtual]`

Process a geometry message.

Parameters

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

Implemented in `gazebo::physics::MultiRayShape` (p.525), `gazebo::physics::RayShape` (p.638), `gazebo::physics::HeightmapShape` (p.352), `gazebo::physics::PlaneShape` (p.580), `gazebo::physics::TrimeshShape` (p.809), `gazebo::physics::CylinderShape` (p.249), `gazebo::physics::SphereShape` (p.739), and `gazebo::physics::BoxShape` (p.153).

10.147.4 Member Data Documentation

10.147.4.1 CollisionPtr gazebo::physics::Shape::collisionParent [protected]

This shape's collision parent.

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

- **Shape.hh**

10.148 gazebo::sensors::SimTimeEvent Class Reference

```
#include <SensorManager.hh>
```

Public Attributes

- boost::condition_variable * **condition**
The condition to notify.
- common::Time **time**
The time at which to trigger the condition.

10.148.1 Detailed Description

A (p. 109) simulation time event

10.148.2 Member Data Documentation

10.148.2.1 boost::condition_variable* gazebo::sensors::SimTimeEvent::condition

The condition to notify.

10.148.2.2 common::Time gazebo::sensors::SimTimeEvent::time

The time at which to trigger the condition.

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

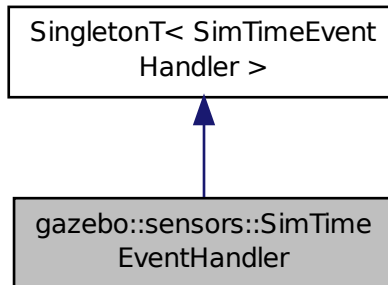
- **SensorManager.hh**

10.149 gazebo::sensors::SimTimeEventHandler Class Reference

Monitors simulation time, and notifies conditions when a specified time has been reached.

```
#include <SensorManager.hh>
```

Inheritance diagram for gazebo::sensors::SimTimeEventHandler:



Public Member Functions

- **SimTimeEventHandler** ()
Constructor.
- virtual **~SimTimeEventHandler** ()
Destructor.
- void **AddRelativeEvent** (const **common::Time** &_time, boost::condition_variable *_var)
Add a new event to the handler.

Additional Inherited Members

10.149.1 Detailed Description

Monitors simulation time, and notifies conditions when a specified time has been reached.

10.149.2 Constructor & Destructor Documentation

10.149.2.1 gazebo::sensors::SimTimeEventHandler::SimTimeEventHandler ()

Constructor.

10.149.2.2 virtual gazebo::sensors::SimTimeEventHandler::~~SimTimeEventHandler () [virtual]

Destructor.

10.149.3 Member Function Documentation

```
10.149.3.1 void gazebo::sensors::SimTimeEventHandler::AddRelativeEvent ( const common::Time & _time,
    boost::condition_variable * _var )
```

Add a new event to the handler.

Parameters

in	<i>_time</i>	Time of the new event. The current sim time will be add to this time.
in	<i>_var</i>	Condition to notify when the time has been reached.

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

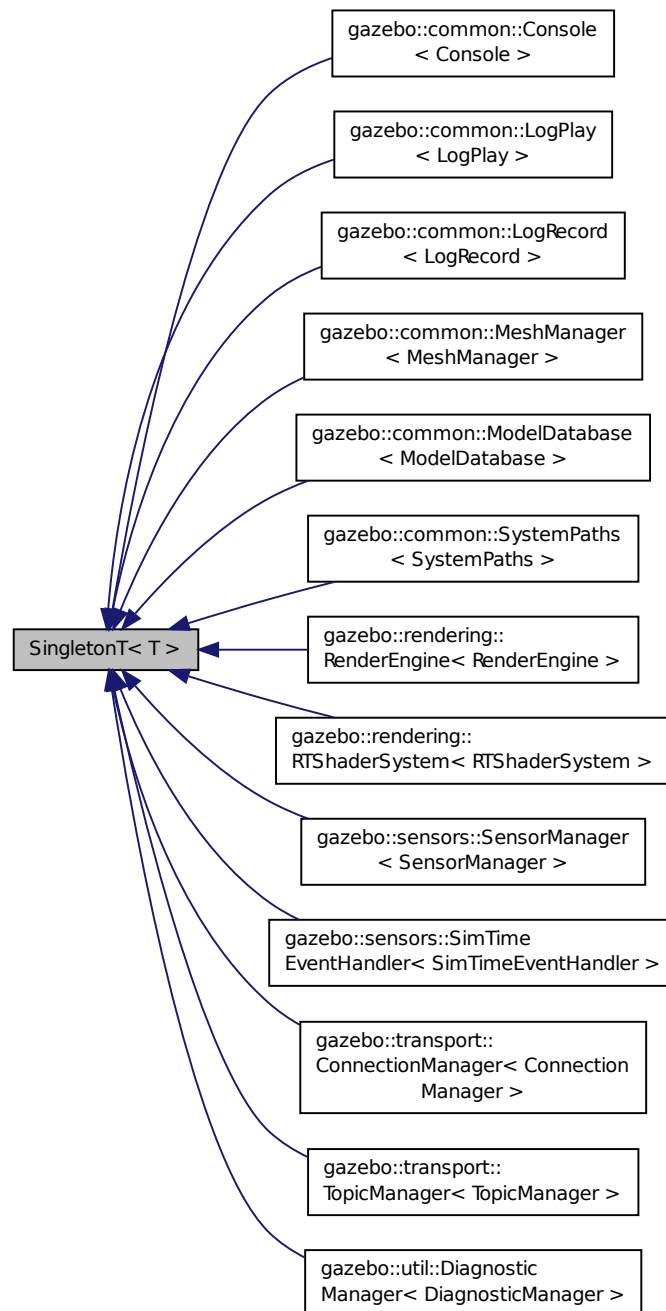
- **SensorManager.hh**

10.150 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.

10.150.1 Detailed Description

```
template<class T>class SingletonT< T >
```

Singleton template class.

10.150.2 Constructor & Destructor Documentation

10.150.2.1 `template<class T> SingletonT< T >::SingletonT () [inline],[protected]`

Constructor.

10.150.2.2 `template<class T> virtual SingletonT< T >::~~SingletonT () [inline],[protected],[virtual]`

Destructor.

10.150.3 Member Function Documentation

10.150.3.1 `template<class T> static T* SingletonT< T >::Instance () [inline],[static]`

Get an instance of the singleton.

Referenced by gazebo::transport::Node::Advertise(), gazebo::transport::TopicManager::Advertise(), gazebo::PluginT< ModelPlugin >::Create(), and gazebo::transport::Node::Subscribe().

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

- **SingletonT.hh**

10.151 gazebo::common::Skeleton Class Reference

A (p. 109) 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

10.151.1 Detailed Description

A (p. 109) skeleton.

10.151.2 Constructor & Destructor Documentation

10.151.2.1 gazebo::common::Skeleton::Skeleton ()

Constructor.

10.151.2.2 gazebo::common::Skeleton::Skeleton (**SkeletonNode** * *_root*)

Constructor.

Parameters

in	<i>_root</i>	node
----	--------------	------

10.151.2.3 virtual gazebo::common::Skeleton::~~Skeleton () [virtual]

Destructor.

10.151.3 Member Function Documentation

10.151.3.1 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
----	--------------	----------------------

10.151.3.2 `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)

10.151.3.3 `void gazebo::common::Skeleton::BuildNodeMap () [protected]`

Initializes the handle numbers for each node in the map using breadth first traversal.

10.151.3.4 `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

10.151.3.5 `math::Matrix4 gazebo::common::Skeleton::GetBindShapeTransform ()`

Return bind pose skeletal transform.

Returns

a matrix

10.151.3.6 `SkeletonNode* gazebo::common::Skeleton::GetNodeByHandle (unsigned int _handle)`

Find or create node with handle.

Parameters

in	<i>_handle</i>	
----	----------------	--

Returns

the node. **A** (p. 109) new node is created if it didn't exist

10.151.3.7 `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

10.151.3.8 `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

10.151.3.9 `NodeMap` gazebo::common::Skeleton::GetNodes ()

Get a copy of the node dictionary.

10.151.3.10 `unsigned int` gazebo::common::Skeleton::GetNumAnimations ()

Returns the number of animations.

Returns

the count

10.151.3.11 `unsigned int` gazebo::common::Skeleton::GetNumJoints ()

Returns the number of joints.

Returns

the count

10.151.3.12 `unsigned int` gazebo::common::Skeleton::GetNumNodes ()

Returns the node count.

Returns

the count

10.151.3.13 `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

10.151.3.14 `SkeletonNode* gazebo::common::Skeleton::GetRootNode ()`

Return the root.

Returns

the root

10.151.3.15 `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

10.151.3.16 `void gazebo::common::Skeleton::PrintTransforms ()`

Outputs the transforms to `std::err` stream.

10.151.3.17 `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
-----------------	-------------------------	----------------

10.151.3.18 `void gazebo::common::Skeleton::SetBindShapeTransform (math::Matrix4 _trans)`

Set the bind pose skeletal transform.

Parameters

in	<i>_trans</i>	the transform
----	---------------	---------------

10.151.3.19 void gazebo::common::Skeleton::SetNumVertAttached (unsigned int *_vertices*)

Resizes the raw node weight array.

Parameters

in	<i>_vertices</i>	the new size
----	------------------	--------------

10.151.3.20 void gazebo::common::Skeleton::SetRootNode (SkeletonNode * *_node*)

Change the root node.

Parameters

in	<i>_node</i>	the new node
----	--------------	--------------

10.151.4 Member Data Documentation

10.151.4.1 std::vector<SkeletonAnimation*> gazebo::common::Skeleton::anim [protected]

the array of animations

10.151.4.2 math::Matrix4 gazebo::common::Skeleton::bindShapeTransform [protected]

the bind pose skeletal transform

10.151.4.3 NodeMap gazebo::common::Skeleton::nodes [protected]

The dictionary of nodes, indexed by name.

10.151.4.4 RawNodeWeights gazebo::common::Skeleton::rawNW [protected]

the node weight table

10.151.4.5 SkeletonNode* gazebo::common::Skeleton::root [protected]

the root node

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

- **Skeleton.hh**

10.152 gazebo::common::SkeletonAnimation Class Reference

Skeleton (p. 713) 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

10.152.1 Detailed Description

Skeleton (p. 713) animation.

10.152.2 Constructor & Destructor Documentation

10.152.2.1 gazebo::common::SkeletonAnimation::SkeletonAnimation (const std::string & *_name*)

The Constructor.

Parameters

in	<i>_name</i>	the name of the animation
----	--------------	---------------------------

10.152.2.2 gazebo::common::SkeletonAnimation::~~SkeletonAnimation ()

The destructor.

Clears the list without destroying the animations

10.152.3 Member Function Documentation

10.152.3.1 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

10.152.3.2 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. 584)

10.152.3.3 double gazebo::common::SkeletonAnimation::GetLength () const

Returns the duration of the animations.

Returns

the duration in seconds

10.152.3.4 `std::string gazebo::common::SkeletonAnimation::GetName () const`

Returns the name.

Returns

the name

10.152.3.5 `unsigned int gazebo::common::SkeletonAnimation::GetNodeCount () const`

Returns the number of animation nodes.

Returns

the count

10.152.3.6 `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

10.152.3.7 `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

10.152.3.8 `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)

10.152.3.9 `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

10.152.3.10 `void gazebo::common::SkeletonAnimation::Scale (const double _scale)`

Scales every animation in the animations list.

Parameters

in	<i>_scale</i>	the scaling factor
----	---------------	--------------------

10.152.3.11 `void gazebo::common::SkeletonAnimation::SetName (const std::string & _name)`

Changes the name.

Parameters

in	<i>_name</i>	the new name
----	--------------	--------------

10.152.4 Member Data Documentation

10.152.4.1 `std::map<std::string, NodeAnimation*> gazebo::common::SkeletonAnimation::animations` [protected]

a dictionary of node animations

10.152.4.2 `double gazebo::common::SkeletonAnimation::length` [protected]

the duration of the longest animation

10.152.4.3 `std::string gazebo::common::SkeletonAnimation::name` [protected]

the node name

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

- **SkeletonAnimation.hh**

10.153 gazebo::common::SkeletonNode Class Reference

A (p. 109) skeleton node.

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

Public Types

- enum **SkeletonNodeType** { **NODE**, **JOINT** }
enumeration of node types

Public Member Functions

- **SkeletonNode** (**SkeletonNode** *_parent)
Constructor.
- **SkeletonNode** (**SkeletonNode** *_parent, std::string _name, std::string _id, **SkeletonNodeType** _type=**JOINT**)
Constructor.
- virtual ~**SkeletonNode** ()
Destructor.
- void **AddChild** (**SkeletonNode** *_child)
Add a new child.
- void **AddRawTransform** (**NodeTransform** _t)
Add a raw transform.
- **SkeletonNode** * **GetChild** (unsigned int _index)
Find a child by index.
- **SkeletonNode** * **GetChildById** (std::string _id)
Get child by string id.
- **SkeletonNode** * **GetChildByName** (std::string _name)
Get child by name.
- unsigned int **GetChildCount** ()
Returns the children count.
- unsigned int **GetHandle** ()
Get the handle index.
- std::string **GetId** ()
Returns the index.

- **math::Matrix4 GetInverseBindTransform ()**
Retrieve the inverse of the bind pose skeletal transform.
- **math::Matrix4 GetModelTransform ()**
Retrieve the model transform.
- **std::string GetName ()**
Returns the name.
- **unsigned int GetNumRawTrans ()**
Return the raw transformations count.
- **SkeletonNode * GetParent ()**
Returns the parent node.
- **NodeTransform GetRawTransform (unsigned int _i)**
Find a raw transformation.
- **std::vector< NodeTransform > GetRawTransforms ()**
Retrieve the raw transformations.
- **math::Matrix4 GetTransform ()**
Get transform relative to parent.
- **std::vector< NodeTransform > GetTransforms ()**
Returns a copy of the array of transformations.
- **bool IsJoint ()**
Is a joint query.
- **bool IsRootNode ()**
Queries wether a node has no parent parent.
- **void Reset (bool _resetChildren)**
Reset the transformation to the initial transformation.
- **void SetHandle (unsigned int _h)**
Assign a handle number.
- **void SetId (std::string _id)**
Change the id string.
- **void SetInitialTransform (math::Matrix4 _tras)**
Sets the initial transformation.
- **void SetInverseBindTransform (math::Matrix4 _invBM)**
Assign the inverse of the bind pose skeletal transform.
- **void SetModelTransform (math::Matrix4 _trans, bool _updateChildren=true)**
Set the model transformation.
- **void SetName (std::string _name)**
Change the name.
- **void SetParent (SkeletonNode *_parent)**
Set the parent node.
- **void SetTransform (math::Matrix4 _trans, bool _updateChildren=true)**
Set a transformation.
- **void SetType (SkeletonNodeType _type)**
Change the skeleton node type.
- **void UpdateChildrenTransforms ()**
Apply model transformations in order for each node in the tree.

Protected Attributes

- `std::vector< SkeletonNode * > children`
the children nodes
- `unsigned int handle`
handle index number
- `std::string id`
a string identifier
- `math::Matrix4 initialTransform`
the initial transformation
- `math::Matrix4 invBindTransform`
the inverse of the bind pose skeletal transform
- `math::Matrix4 modelTransform`
the model transformation
- `std::string name`
the name of the skeletal node
- `SkeletonNode * parent`
the parent node
- `std::vector< NodeTransform > rawTransforms`
the raw transformation
- `math::Matrix4 transform`
the transform
- `SkeletonNodeType type`
the type fo node

10.153.1 Detailed Description

A (p. 109) skeleton node.

10.153.2 Member Enumeration Documentation

10.153.2.1 enum `gazebo::common::SkeletonNode::SkeletonNodeType`

enumeration of node types

Enumerator:

NODE

JOINT

10.153.3 Constructor & Destructor Documentation

10.153.3.1 `gazebo::common::SkeletonNode::SkeletonNode (SkeletonNode * _parent)`

Constructor.

Parameters

<code>in</code>	<code><i>_parent</i></code>	The parent node
-----------------	-----------------------------	-----------------

10.153.3.2 `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

10.153.3.3 `virtual gazebo::common::SkeletonNode::~~SkeletonNode () [virtual]`

Destructor.

10.153.4 Member Function Documentation

10.153.4.1 `void gazebo::common::SkeletonNode::AddChild (SkeletonNode * _child)`

Add a new child.

Parameters

in	<code>_child</code>	a child
----	---------------------	---------

10.153.4.2 `void gazebo::common::SkeletonNode::AddRawTransform (NodeTransform _t)`

Add a raw transform.

Parameters

in	<code>_t</code>	the transform
----	-----------------	---------------

10.153.4.3 `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

10.153.4.4 `SkeletonNode* gazebo::common::SkeletonNode::GetChildById (std::string _id)`

Get child by string id.

Parameters

in	<code>_id</code>	the string id
----	------------------	---------------

Returns

the child skeleton or NULL if not found

10.153.4.5 `SkeletonNode*` gazebo::common::SkeletonNode::GetChildByName (`std::string _name`)

Get child by name.

Parameters

in	<code>_name</code>	the name of the child skeleton
----	--------------------	--------------------------------

Returns

the skeleton, or NULL if not found

10.153.4.6 `unsigned int` gazebo::common::SkeletonNode::GetChildCount ()

Returns the children count.

Returns

the count

10.153.4.7 `unsigned int` gazebo::common::SkeletonNode::GetHandle ()

Get the handle index.

Returns

the handle index

10.153.4.8 `std::string` gazebo::common::SkeletonNode::GetId ()

Returns the index.

Returns

the id string

10.153.4.9 `math::Matrix4` gazebo::common::SkeletonNode::GetInverseBindTransform ()

Retrieve the inverse of the bind pose skeletal transform.

Returns

the transform

10.153.4.10 `math::Matrix4 gazebo::common::SkeletonNode::GetModelTransform ()`

Retrieve the model transform.

Returns

the transform

10.153.4.11 `std::string gazebo::common::SkeletonNode::GetName ()`

Returns the name.

Returns

the name

10.153.4.12 `unsigned int gazebo::common::SkeletonNode::GetNumRawTrans ()`

Return the raw transformations count.

Returns

the count

10.153.4.13 `SkeletonNode* gazebo::common::SkeletonNode::GetParent ()`

Returns the parent node.

Returns

the parent

10.153.4.14 `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

10.153.4.15 `std::vector<NodeTransform> gazebo::common::SkeletonNode::GetRawTransforms ()`

Retrieve the raw transformations.

Returns

an array of transformations

10.153.4.16 `math::Matrix4 gazebo::common::SkeletonNode::GetTransform ()`

Get transform relative to parent.

10.153.4.17 `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)

10.153.4.18 `bool gazebo::common::SkeletonNode::IsJoint ()`

Is a joint query.

Returns

true if the skeleton type is a joint, false otherwise

10.153.4.19 `bool gazebo::common::SkeletonNode::IsRootNode ()`

Queries whether a node has no parent parent.

Returns

true if the node has no parent, false otherwise

10.153.4.20 `void gazebo::common::SkeletonNode::Reset (bool _resetChildren)`

Reset the transformation to the initial transformation.

Parameters

<code>in</code>	<code>_resetChildren</code>	when true, performs the operation for every node in the tree
-----------------	-----------------------------	--

10.153.4.21 `void gazebo::common::SkeletonNode::SetHandle (unsigned int _h)`

Assign a handle number.

Parameters

<code>in</code>	<code>_h</code>	the handle
-----------------	-----------------	------------

10.153.4.22 `void gazebo::common::SkeletonNode::SetId (std::string _id)`

Change the id string.

Parameters

<code>in</code>	<code>_id</code>	the new id string
-----------------	------------------	-------------------

10.153.4.23 `void gazebo::common::SkeletonNode::SetInitialTransform (math::Matrix4 _tras)`

Sets the initial transformation.

Parameters

<code>in</code>	<code>_tras</code>	the transformation matrix
-----------------	--------------------	---------------------------

10.153.4.24 `void gazebo::common::SkeletonNode::SetInverseBindTransform (math::Matrix4 _invBM)`

Assign the inverse of the bind pose skeletal transform.

Parameters

<code>in</code>	<code>_invBM</code>	the transform
-----------------	---------------------	---------------

10.153.4.25 `void gazebo::common::SkeletonNode::SetModelTransform (math::Matrix4 _trans, bool _updateChildren = true)`

Set the model transformation.

Parameters

<code>in</code>	<code>_trans</code>	the transformation
<code>in</code>	<code>_updateChildren</code>	when true the UpdateChildrenTransforms operation is performed

10.153.4.26 `void gazebo::common::SkeletonNode::SetName (std::string _name)`

Change the name.

Parameters

<code>in</code>	<code>_name</code>	the new name
-----------------	--------------------	--------------

10.153.4.27 `void gazebo::common::SkeletonNode::SetParent (SkeletonNode * _parent)`

Set the parent node.

Parameters

<code>in</code>	<code>_parent</code>	the new parent
-----------------	----------------------	----------------

10.153.4.28 `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

10.153.4.29 `void gazebo::common::SkeletonNode::SetType (SkeletonNodeType _type)`

Change the skeleton node type.

Parameters

in	<code>_type</code>	the new type
----	--------------------	--------------

10.153.4.30 `void gazebo::common::SkeletonNode::UpdateChildrenTransforms ()`

Apply model transformations in order for each node in the tree.

10.153.5 Member Data Documentation

10.153.5.1 `std::vector<SkeletonNode*> gazebo::common::SkeletonNode::children` [protected]

the children nodes

10.153.5.2 `unsigned int gazebo::common::SkeletonNode::handle` [protected]

handle index number

10.153.5.3 `std::string gazebo::common::SkeletonNode::id` [protected]

a string identifier

10.153.5.4 `math::Matrix4 gazebo::common::SkeletonNode::initialTransform` [protected]

the initial transformation

10.153.5.5 `math::Matrix4 gazebo::common::SkeletonNode::invBindTransform` [protected]

the inverse of the bind pose skeletal transform

10.153.5.6 `math::Matrix4 gazebo::common::SkeletonNode::modelTransform` [protected]

the model transformation

10.153.5.7 `std::string gazebo::common::SkeletonNode::name` [protected]

the name of the skeletal node

10.153.5.8 `SkeletonNode* gazebo::common::SkeletonNode::parent` [protected]

the parent node

10.153.5.9 `std::vector<NodeTransform> gazebo::common::SkeletonNode::rawTransforms` [protected]

the raw transformation

10.153.5.10 `math::Matrix4 gazebo::common::SkeletonNode::transform` [protected]

the transform

10.153.5.11 `SkeletonNodeType gazebo::common::SkeletonNode::type` [protected]

the type fo node

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

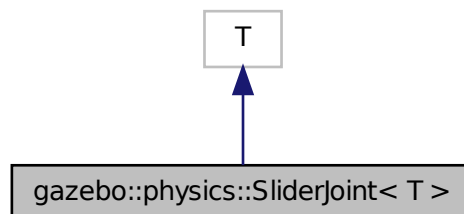
- **Skeleton.hh**

10.154 gazebo::physics::SliderJoint< T > Class Template Reference

A (p. 109) 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. 733).

- virtual void `SetAnchor (int _index, const math::Vector3 &_anchor)`

Set the anchor.

Protected Attributes

- `math::Vector3 fakeAnchor`

The anchor value is not used internally.

10.154.1 Detailed Description

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

A (p. 109) slider joint.

10.154.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

<code>in</code>	<code>_parent</code>	Parent of the joint.
-----------------	----------------------	----------------------

References `gazebo::physics::Base::SLIDER_JOINT`.

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

Destructor.

10.154.3 Member Function Documentation

10.154.3.1 `template<class T > math::Vector3 gazebo::physics::SliderJoint< T >::GetAnchor (int _index) const [virtual]`

Get the anchor.

Parameters

in	<i>_index</i>	Index of the axis. Not used.
----	---------------	------------------------------

Returns

Anchor for the joint.

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

10.154.3.3 `template<class T > virtual void gazebo::physics::SliderJoint< T >::Load (sdf::ElementPtr _sdf)`
`[inline],[virtual]`

Load a **SliderJoint** (p. 733).

Parameters

in	<i>_sdf</i>	SDF values to load from
----	-------------	-------------------------

10.154.3.4 `template<class T > void gazebo::physics::SliderJoint< T >::SetAnchor (int _index, const math::Vector3 & _anchor)`
`[virtual]`

Set the anchor.

Parameters

in	<i>_index</i>	Index of the axis. Not used.
in	<i>_anchor</i>	Anchor for the axis.

10.154.4 Member Data Documentation

10.154.4.1 `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:

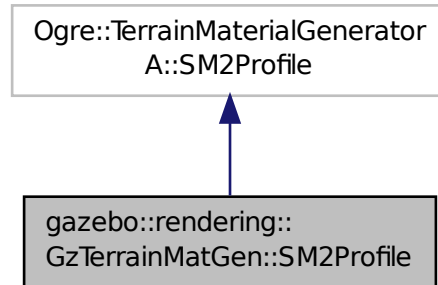
- **SliderJoint.hh**

10.155 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)

10.155.1 Detailed Description

Shader model 2 profile target.

10.155.2 Constructor & Destructor Documentation

10.155.2.1 `gazebo::rendering::GzTerrainMatGen::SM2Profile::SM2Profile (Ogre::TerrainMaterialGenerator * _parent, const Ogre::String & _name, const Ogre::String & _desc)`

Constructor.

10.155.2.2 `virtual gazebo::rendering::GzTerrainMatGen::SM2Profile::~SM2Profile () [virtual]`

Destructor.

10.155.3 Member Function Documentation

10.155.3.1 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::addTechnique (const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain, TechniqueType _tt) [protected],[virtual]`

10.155.3.2 `Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generate (const Ogre::Terrain * _terrain)`

10.155.3.3 `Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generateForCompositeMap (const Ogre::Terrain * _terrain)`

10.155.3.4 `void gazebo::rendering::GzTerrainMatGen::SM2Profile::UpdateParams (const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain)`

10.155.3.5 `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:

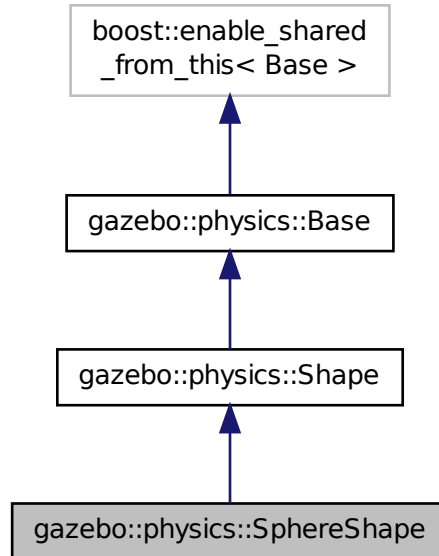
- **Heightmap.hh**

10.156 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.

Additional Inherited Members

10.156.1 Detailed Description

Sphere collision shape.

10.156.2 Constructor & Destructor Documentation

10.156.2.1 `gazebo::physics::SphereShape::SphereShape (CollisionPtr _parent) [explicit]`

Constructor.

Parameters

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

10.156.2.2 `virtual gazebo::physics::SphereShape::~~SphereShape () [virtual]`

Destructor.

10.156.3 Member Function Documentation

10.156.3.1 `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. 708).

10.156.3.2 `double gazebo::physics::SphereShape::GetRadius () const`

Get the sphere's radius.

Returns

Radius of the sphere.

10.156.3.3 `virtual void gazebo::physics::SphereShape::Init () [virtual]`

Initialize the sphere.

Implements **gazebo::physics::Shape** (p. 708).

10.156.3.4 `virtual void gazebo::physics::SphereShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]`

Process a geometry message.

Parameters

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

Implements **gazebo::physics::Shape** (p. 708).

10.156.3.5 virtual void gazebo::physics::SphereShape::SetRadius (double *_radius*) [virtual]

Set the size.

Parameters

in	<i>_radius</i>	Radius of the sphere.
----	----------------	-----------------------

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

- **SphereShape.hh**

10.157 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.

10.157.1 Detailed Description

Splines.

10.157.2 Constructor & Destructor Documentation

10.157.2.1 gazebo::math::Spline::Spline ()

constructor

10.157.2.2 gazebo::math::Spline::~~Spline ()

destructor

10.157.3 Member Function Documentation

10.157.3.1 void gazebo::math::Spline::AddPoint (const Vector3 & _pt)

Adds a control point to the end of the spline.

Parameters

in	_pt	point to add
----	-----	--------------

10.157.3.2 void gazebo::math::Spline::Clear ()

Clears all the points in the spline.

10.157.3.3 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

10.157.3.4 `unsigned int gazebo::math::Spline::GetPointCount () const`

Gets the number of control points in the spline.

Returns

the count

10.157.3.5 `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
-----------------	---------------------	-------------------------

10.157.3.6 `double gazebo::math::Spline::GetTension () const`

Get the tension value.

Returns

The value of the tension, which is between 0.0 and 1.0

10.157.3.7 `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)
-----------------	-----------------	--------------------------

10.157.3.8 `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

10.157.3.9 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.

10.157.3.10 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.
----	------------------	--

10.157.3.11 void gazebo::math::Spline::SetTension (double *_t*)

Set the tension parameter.

A (p. 109) value of 0 = Catmull-Rom spline.

Parameters

in	<i>_t</i>	Tension value between 0.0 and 1.0
----	-----------	-----------------------------------

10.157.3.12 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

10.157.4 Member Data Documentation

10.157.4.1 `bool gazebo::math::Spline::autoCalc` [protected]

when true, the tangents are recalculated when the control point change

10.157.4.2 `Matrix4 gazebo::math::Spline::coeffs` [protected]

Matrix of coefficients.

10.157.4.3 `std::vector<Vector3> gazebo::math::Spline::points` [protected]

control points

10.157.4.4 `std::vector<Vector3> gazebo::math::Spline::tangents` [protected]

tangents

10.157.4.5 `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:

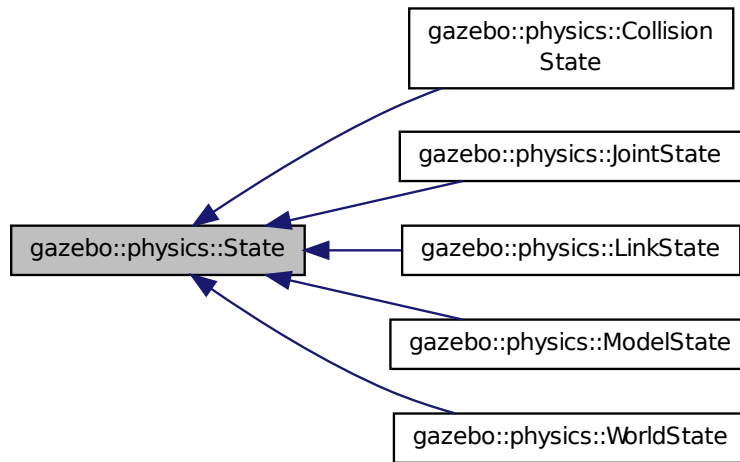
- [Spline.hh](#)

10.158 gazebo::physics::State Class Reference

State (p. 744) 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. 744).*
- **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. 744).*

Protected Attributes

- `std::string name`
Name associated with this **State** (p. 744).
- `common::Time realTime`
- `common::Time simTime`
- `common::Time wallTime`
Times for the state data.

10.158.1 Detailed Description

State (p. 744) of an entity.

This is the base class for all **State** (p. 744) information.

10.158.2 Constructor & Destructor Documentation

10.158.2.1 `gazebo::physics::State::State ()`

Default constructor.

10.158.2.2 `gazebo::physics::State::State (const std::string & _name, const common::Time & _realTime, const common::Time & _simTime)`

Constructor.

Construct a **State** (p. 744) object using some basic information.

Parameters

<code>_name</code>	Name associated with the State (p. 744) information. This is typically the name of an Entity (p. 277). <code>_realTime</code> Clock time since simulation started.
<code>_simTime</code>	Simulation time associated with this State (p. 744) info.

10.158.2.3 `virtual gazebo::physics::State::~~State () [virtual]`

Destructor.

10.158.3 Member Function Documentation

10.158.3.1 `std::string gazebo::physics::State::GetName () const`

Get the name associated with this **State** (p. 744).

Returns

Name associated with this state information. Typically a name of an **Entity** (p. 277).

10.158.3.2 `common::Time gazebo::physics::State::GetRealTime () const`

Get the real time when this state was generated.

Returns

Clock time since simulation was stated.

10.158.3.3 `common::Time gazebo::physics::State::GetSimTime () const`

Get the sim time when this state was generated.

Returns

Simulation time when the data was recorded.

10.158.3.4 `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. 744) data was recorded.

10.158.3.5 `virtual void gazebo::physics::State::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Populates the **State** (p. 744) information from data stored in an SDF::Element

Parameters

<code>_elem</code>	Pointer to the SDF::Element
--------------------	-----------------------------

Reimplemented in **gazebo::physics::ModelState** (p. 502), **gazebo::physics::LinkState** (p. 435), **gazebo::physics::WorldState** (p. 910), **gazebo::physics::CollisionState** (p. 203), and **gazebo::physics::JointState** (p. 398).

10.158.3.6 `State gazebo::physics::State::operator- (const State & .state) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_pt</code>	A (p. 109) state to subtract.
-----------------	------------------	--------------------------------------

Returns

The resulting state.

10.158.3.7 State& gazebo::physics::State::operator= (const State & *_state*)

Assignment operator.

Parameters

<i>in</i>	<i>_state</i>	State (p. 744) value
-----------	---------------	-----------------------------

Returns

this

10.158.3.8 void gazebo::physics::State::SetName (const std::string & *_name*)

Set the name associated with this **State** (p. 744).

Parameters

<i>in</i>	<i>_name</i>	Name associated with this state information. Typically the name of an Entity (p. 277).
-----------	--------------	---

10.158.4 Member Data Documentation**10.158.4.1 std::string gazebo::physics::State::name** [protected]

Name associated with this **State** (p. 744).

10.158.4.2 common::Time gazebo::physics::State::realTime [protected]**10.158.4.3 common::Time gazebo::physics::State::simTime** [protected]**10.158.4.4 common::Time gazebo::physics::State::wallTime** [protected]

Times for the state data.

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

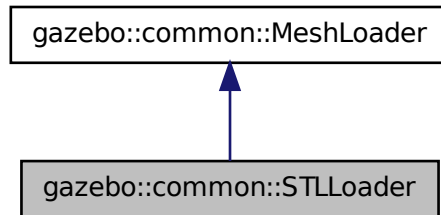
- **State.hh**

10.159 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.

10.159.1 Detailed Description

Class used to load STL mesh files.

10.159.2 Constructor & Destructor Documentation

10.159.2.1 gazebo::common::STLloader::STLloader ()

Constructor.

10.159.2.2 virtual gazebo::common::STLloader::~~STLloader () [virtual]

Destructor.

10.159.3 Member Function Documentation

10.159.3.1 virtual Mesh* gazebo::common::STLloader::Load (const std::string &_filename) [virtual]

Creates a new mesh and loads the data from a file.

Parameters

in	_filename	the mesh file
----	-----------	---------------

Implements **gazebo::common::MeshLoader** (p. 475).

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

- **STLloader.hh**

10.160 gazebo::common::SubMesh Class Reference

A (p. 109) 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.
- 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 **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 maximun X, Y, Z values.
- unsigned int **GetMaxIndex** () const

Get the highest index value.
- **math::Vector3 GetMin** () const

Get the minimum X, Y, Z values.
- **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 **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.

10.160.1 Detailed Description

A (p. 109) child mesh.

10.160.2 Member Enumeration Documentation

10.160.2.1 enum gazebo::common::SubMesh::PrimitiveType

An enumeration of the geometric mesh primitives.

Enumerator:

POINTS
LINES
LINESTRIPS
TRIANGLES
TRIFANS
TRISTRIPS

10.160.3 Constructor & Destructor Documentation

10.160.3.1 gazebo::common::SubMesh::SubMesh ()

Constructor.

10.160.3.2 virtual gazebo::common::SubMesh::~~SubMesh () [virtual]

Destructor.

10.160.4 Member Function Documentation

10.160.4.1 void gazebo::common::SubMesh::AddIndex (unsigned int _i)

Add an index to the mesh.

Parameters

in	<i>_i</i>	the new vertex index
----	-----------	----------------------

10.160.4.2 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)

10.160.4.3 void gazebo::common::SubMesh::AddNormal (const math::Vector3 & *_n*)

Add a normal to the mesh.

Parameters

in	<i>_n</i>	the normal
----	-----------	------------

10.160.4.4 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

10.160.4.5 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

10.160.4.6 void gazebo::common::SubMesh::AddVertex (const math::Vector3 & *_v*)

Add a vertex to the mesh.

Parameters

in	<i>_v</i>	the new position
----	-----------	------------------

10.160.4.7 void gazebo::common::SubMesh::AddVertex (double *_x*, double *_y*, double *_z*)

Add a vertex to the mesh.

Parameters

in	<i>_x</i>	position along x
in	<i>_y</i>	position along y
in	<i>_z</i>	position along z

10.160.4.8 void gazebo::common::SubMesh::CopyNormals (const std::vector< math::Vector3 > & *_norms*)

Copy normals from a vector.

Parameters

in	<i>_norms</i>	to copy from
----	---------------	--------------

10.160.4.9 void gazebo::common::SubMesh::CopyVertices (const std::vector< math::Vector3 > & *_verts*)

Copy vertices from a vector.

Parameters

in	<i>_verts</i>	the vertices to copy from
----	---------------	---------------------------

10.160.4.10 void gazebo::common::SubMesh::FillArrays (float ** *_vertArr*, int ** *_indArr*) const

Put all the data into flat arrays.

Parameters

in	<i>_vertArr</i>	
in	<i>_indArr</i>	

10.160.4.11 void gazebo::common::SubMesh::GenSphericalTexCoord (const math::Vector3 & *_center*)

Generate texture coordinates using spherical projection from center.

Parameters

in	<i>_center</i>	
----	----------------	--

10.160.4.12 unsigned int gazebo::common::SubMesh::GetIndex (unsigned int *_i*) const

Get an index.

Parameters

in	<i>_i</i>	
----	-----------	--

10.160.4.13 unsigned int gazebo::common::SubMesh::GetIndexCount () const

Return the number of indices.

10.160.4.14 unsigned int gazebo::common::SubMesh::GetMaterialIndex () const

Get the material index.

10.160.4.15 math::Vector3 gazebo::common::SubMesh::GetMax () const

Get the maximum X, Y, Z values.

Returns

10.160.4.16 unsigned int gazebo::common::SubMesh::GetMaxIndex () const

Get the highest index value.

10.160.4.17 math::Vector3 gazebo::common::SubMesh::GetMin () const

Get the minimum X, Y, Z values.

Returns

10.160.4.18 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
----	-----------	-----------------------------

10.160.4.19 unsigned int gazebo::common::SubMesh::GetNodeAssignmentsCount () const

Return the number of vertex - skeleton node assignments.

10.160.4.20 math::Vector3 gazebo::common::SubMesh::GetNormal (unsigned int *_i*) const

Get a normal.

Parameters

in	<code>_i</code>	the normal index
----	-----------------	------------------

Returns

the orientation of the normal, or throws an exception

10.160.4.21 `unsigned int gazebo::common::SubMesh::GetNormalCount () const`

Return the number of normals.

10.160.4.22 `PrimitiveType gazebo::common::SubMesh::GetPrimitiveType () const`

Get the primitive type.

Returns

the primitive type

10.160.4.23 `math::Vector2d gazebo::common::SubMesh::GetTexCoord (unsigned int _i) const`

Get a tex coord.

Parameters

in	<code>_i</code>	the texture index
----	-----------------	-------------------

Returns

the texture coordinates

10.160.4.24 `unsigned int gazebo::common::SubMesh::GetTexCoordCount () const`

Return the number of texture coordinates.

10.160.4.25 `math::Vector3 gazebo::common::SubMesh::GetVertex (unsigned int _i) const`

Get a vertex.

Parameters

in	<code>_i</code>	the vertex index
----	-----------------	------------------

Returns

the position or throws an exception

10.160.4.26 unsigned int gazebo::common::SubMesh::GetVertexCount () const

Return the number of vertices.

10.160.4.27 unsigned int gazebo::common::SubMesh::GetVertexIndex (const math::Vector3 & _v) const

Get the index of the vertex.

Parameters

in	_v	
----	----	--

10.160.4.28 bool gazebo::common::SubMesh::HasVertex (const math::Vector3 & _v) const

Return true if this submesh has the vertex.

Parameters

in	_v	
----	----	--

10.160.4.29 void gazebo::common::SubMesh::RecalculateNormals ()

Recalculate all the normals.

10.160.4.30 void gazebo::common::SubMesh::Scale (double _factor)

Scale all vertices by _factor.

Parameters

in	_factor	Scaling factor
----	---------	----------------

10.160.4.31 void gazebo::common::SubMesh::SetIndexCount (unsigned int _count)

Resize the index array.

Parameters

in	_count	the new size of the array
----	--------	---------------------------

10.160.4.32 void gazebo::common::SubMesh::SetMaterialIndex (unsigned int _index)

Set the material index.

Relates to the parent mesh material list

Parameters

in	<i>_index</i>	
----	---------------	--

10.160.4.33 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

10.160.4.34 void gazebo::common::SubMesh::SetNormalCount (unsigned int *_count*)

Resize the normal array.

Parameters

in	<i>_count</i>	the new size of the array
----	---------------	---------------------------

10.160.4.35 void gazebo::common::SubMesh::SetPrimitiveType (PrimitiveType *_type*)

Set the primitive type.

Parameters

in	<i>_type</i>	the type
----	--------------	----------

10.160.4.36 void gazebo::common::SubMesh::SetScale (const math::Vector3 & *_factor*)

Scale all vertices by the *_factor* vector.

Parameters

in	<i>_factor</i>	Scaling vector
----	----------------	----------------

10.160.4.37 void gazebo::common::SubMesh::SetSubMeshCenter (math::Vector3 *_center*)

Reset mesh center to geometric center.

Parameters

in	<i>_center</i>	
----	----------------	--

10.160.4.38 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>	

10.160.4.39 void gazebo::common::SubMesh::SetTexCoordCount (unsigned int *_count*)

Resize the texture coordinate array.

Parameters

in	<i>_count</i>	
----	---------------	--

10.160.4.40 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

10.160.4.41 void gazebo::common::SubMesh::SetVertexCount (unsigned int *_count*)

Resize the vertex array.

Parameters

in	<i>_count</i>	the new size of the array
----	---------------	---------------------------

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

- **Mesh.hh**

10.161 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.

10.161.1 Detailed Description

Options for a subscription.

10.161.2 Constructor & Destructor Documentation

10.161.2.1 `gazebo::transport::SubscribeOptions::SubscribeOptions` () [`inline`]

Constructor.

10.161.3 Member Function Documentation

10.161.3.1 `bool gazebo::transport::SubscribeOptions::GetLatching` () const [`inline`]

Are we latching?

Returns

true if we're latching the latest message, false otherwise

10.161.3.2 `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

10.161.3.3 `NodePtr gazebo::transport::SubscribeOptions::GetNode` () const [`inline`]

Get the node we're subscribed to.

Returns

The associated node

10.161.3.4 `std::string gazebo::transport::SubscribeOptions::GetTopic () const` `[inline]`

Get the topic we're subscribed to.

Returns

The topic we're subscribed to

10.161.3.5 `template<class M> void gazebo::transport::SubscribeOptions::Init (const std::string & _topic, NodePtr _node, bool _latching)` `[inline]`

Initialize the options.

Parameters

<code>in</code>	<code>_topic</code>	Topic we're subscribing to
<code>in, out</code>	<code>_node</code>	The associated node
<code>in</code>	<code>_latching</code>	If true, latch the latest message; if false, don't latch

References `gthrow`, and `NULL`.

Referenced by `gazebo::transport::Node::Subscribe()`.

10.161.3.6 `void gazebo::transport::SubscribeOptions::Init (const std::string & _topic, NodePtr _node, bool _latching)` `[inline]`

Initialize the options.

This version of `init` is only used when creating subscribers of raw data.

Parameters

<code>in</code>	<code>_topic</code>	Topic we're subscribing to
<code>in, out</code>	<code>_node</code>	The associated node
<code>in</code>	<code>_latching</code>	If true, latch the latest message; if false, don't latch

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

- **SubscribeOptions.hh**

10.162 gazebo::transport::Subscriber Class Reference

A (p. 109) 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.

10.162.1 Detailed Description

A (p. 109) subscriber to a topic.

10.162.2 Constructor & Destructor Documentation

10.162.2.1 gazebo::transport::Subscriber::Subscriber (const std::string & *_topic*, NodePtr *_node*)

Constructor.

Parameters

in	<i>_topic</i>	The topic we're subscribing to
in	<i>_node</i>	The associated node

10.162.2.2 virtual gazebo::transport::Subscriber::~~Subscriber () [virtual]

Destructor.

10.162.3 Member Function Documentation

10.162.3.1 unsigned int gazebo::transport::Subscriber::GetCallbackId () const

10.162.3.2 std::string gazebo::transport::Subscriber::GetTopic () const

Get the topic name.

Returns

The topic name

10.162.3.3 void gazebo::transport::Subscriber::SetCallbackId (unsigned int *_id*)

Referenced by gazebo::transport::Node::Subscribe().

10.162.3.4 void gazebo::transport::Subscriber::Unsubscribe () const

Unsubscribe from the topic.

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

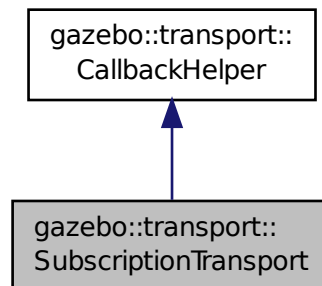
- [Subscriber.hh](#)

10.163 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)
Output a message to a connection.
- void **Init** (const **ConnectionPtr** &_conn, bool _latching)
Initialize the publication link.
- virtual bool **IsLocal** () const
Is the callback local?

Additional Inherited Members

10.163.1 Detailed Description

transport/transport.hh

Handles sending data over the wire to remote subscribers

10.163.2 Constructor & Destructor Documentation

10.163.2.1 gazebo::transport::SubscriptionTransport::SubscriptionTransport ()

Constructor.

10.163.2.2 virtual gazebo::transport::SubscriptionTransport::~~SubscriptionTransport () [virtual]

Destructor.

10.163.3 Member Function Documentation

10.163.3.1 const ConnectionPtr& gazebo::transport::SubscriptionTransport::GetConnection () const

Get the connection we're using.

Returns

Pointer to the connection we're using

10.163.3.2 virtual bool gazebo::transport::SubscriptionTransport::HandleData (const std::string & *_newdata*) [virtual]

Output a message to a connection.

Parameters

in	<i>_newdata</i>	The message to be handled
----	-----------------	---------------------------

Returns

true if the message was handled successfully, false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 157).

10.163.3.3 void gazebo::transport::SubscriptionTransport::Init (const ConnectionPtr & *_conn*, bool *_latching*)

Initialize the publication link.

Parameters

in	<i>_conn</i>	The connection to use
in	<i>_latching</i>	If true, latch the latest message; if false, don't latch

10.163.3.4 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. 157).

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

- **SubscriptionTransport.hh**

10.164 gazebo::physics::SurfaceParams Class Reference

SurfaceParams (p. 765) 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.
- double **erp**
Error Reduction Parameter.
- **math::Vector3** **fdir1**
*Primary friction direction for dry friction coefficient (**SurfaceParams::mu1** (p. 768)) of the friction pyramid.*
- double **kd**
*spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 767) and **SurfaceParams::erp** (p. 767).*
- double **kp**
*spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 767) and **SurfaceParams::erp** (p. 767).*
- 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 direction.

10.164.1 Detailed Description

SurfaceParams (p. 765) defines various Surface contact parameters.

These parameters defines the properties of a **physics::Contact** (p. 232) constraint.

10.164.2 Constructor & Destructor Documentation

10.164.2.1 gazebo::physics::SurfaceParams::SurfaceParams ()

Constructor.

10.164.2.2 virtual gazebo::physics::SurfaceParams::~~SurfaceParams () [virtual]

Destructor.

10.164.3 Member Function Documentation

10.164.3.1 void gazebo::physics::SurfaceParams::FillMsg (msgs::Surface & _msg)

Fill in a surface message.

Parameters

in	<code>_msg</code>	Message to fill with this object's values.
----	-------------------	--

10.164.3.2 virtual void gazebo::physics::SurfaceParams::Load (sdf::ElementPtr _sdf) [virtual]

Load the contact params.

Parameters

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

10.164.3.3 virtual void gazebo::physics::SurfaceParams::ProcessMsg (const msgs::Surface & _msg) [virtual]

10.164.4 Member Data Documentation

10.164.4.1 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

10.164.4.2 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

10.164.4.3 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.

10.164.4.4 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.

10.164.4.5 math::Vector3 gazebo::physics::SurfaceParams::fdir1

Primary friction direction for dry friction coefficient (**SurfaceParams::mu1** (p. 768)) 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

10.164.4.6 double gazebo::physics::SurfaceParams::kd

spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 767) and **SurfaceParams::erp** (p. 767).

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_2 for more details.

10.164.4.7 double gazebo::physics::SurfaceParams::kp

spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p.767) and **SurfaceParams::erp** (p.767).

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_2 for more details.

10.164.4.8 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)

10.164.4.9 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)

10.164.4.10 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

10.164.4.11 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

10.164.4.12 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

10.164.4.13 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:

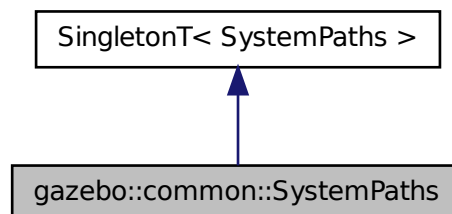
- **SurfaceParams.hh**

10.165 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. 773)*
- bool **modelPathsFromEnv**
 - if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 773)*
- bool **ogrePathsFromEnv**
 - if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 773)*
- bool **pluginPathsFromEnv**
 - if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 773)*

Additional Inherited Members

10.165.1 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. 106) RenderSystem_GL.so et. al.
- SystemPaths::pluginPaths - plugin library paths for common::WorldPlugin

10.165.2 Member Function Documentation

10.165.2.1 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
----	--------------	----------------------

10.165.2.2 void gazebo::common::SystemPaths::AddModelPaths (const std::string & *_path*)

Add colon delimited paths to modelPaths.

Parameters

in	<i>_path</i>	the directory to add
----	--------------	----------------------

10.165.2.3 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
----	--------------	----------------------

10.165.2.4 void gazebo::common::SystemPaths::AddPluginPaths (const std::string & *_path*)

Add colon delimited paths to plugins.

Parameters

in	<i>_path</i>	the directory to add
----	--------------	----------------------

10.165.2.5 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
----	----------------	-------------------

10.165.2.6 void gazebo::common::SystemPaths::ClearGazeboPaths ()

clear out SystemPaths::gazeboPaths

10.165.2.7 void gazebo::common::SystemPaths::ClearModelPaths ()

clear out SystemPaths::modelPaths

10.165.2.8 void gazebo::common::SystemPaths::ClearOgrePaths ()

clear out SystemPaths::ogrePaths

10.165.2.9 void gazebo::common::SystemPaths::ClearPluginPaths ()

clear out SystemPaths::pluginPaths

10.165.2.10 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

10.165.2.11 std::string gazebo::common::SystemPaths::FindFileURI (const std::string & *_uri*)

Find a file or path using a URI.

Parameters

in	<i>_uri</i>	the uniform resource identifier
----	-------------	---------------------------------

Returns

Returns full path name to file

10.165.2.12 `const std::list<std::string>& gazebo::common::SystemPaths::GetGazeboPaths ()`

Get the gazebo install paths.

Returns

a list of paths

10.165.2.13 `std::string gazebo::common::SystemPaths::GetLogPath () const`

Get the log path.

Returns

the path

10.165.2.14 `const std::list<std::string>& gazebo::common::SystemPaths::GetModelPaths ()`

Get the model paths.

Returns

a list of paths

10.165.2.15 `const std::list<std::string>& gazebo::common::SystemPaths::GetOgrePaths ()`

Get the ogre install paths.

Returns

a list of paths

10.165.2.16 `const std::list<std::string>& gazebo::common::SystemPaths::GetPluginPaths ()`

Get the plugin paths.

Returns

a list of paths

10.165.2.17 `std::string gazebo::common::SystemPaths::GetWorldPathExtension ()`

Returns the world path extension.

Returns

Right now, it just returns "/worlds"

10.165.3 Member Data Documentation

10.165.3.1 bool gazebo::common::SystemPaths::gazeboPathsFromEnv

if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 773)

10.165.3.2 bool gazebo::common::SystemPaths::modelPathsFromEnv

if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 773)

10.165.3.3 bool gazebo::common::SystemPaths::ogrePathsFromEnv

if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 773)

10.165.3.4 bool gazebo::common::SystemPaths::pluginPathsFromEnv

if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 773)

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

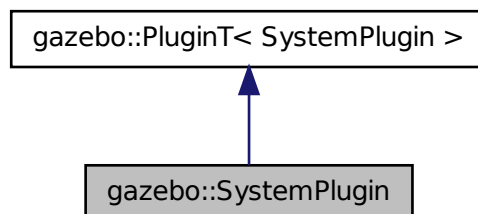
- **SystemPaths.hh**

10.166 gazebo::SystemPlugin Class Reference

A (p. 109) 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

10.166.1 Detailed Description

A (p. 109) plugin loaded within the gzserver on startup.

See [reference](#).

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

10.166.2 Constructor & Destructor Documentation

10.166.2.1 gazebo::SystemPlugin::SystemPlugin () [inline]

Constructor.

References `gazebo::SYSTEM_PLUGIN`, and `gazebo::PluginT< SystemPlugin >::type`.

10.166.2.2 virtual gazebo::SystemPlugin::~~SystemPlugin () [inline],[virtual]

Destructor.

10.166.3 Member Function Documentation

10.166.3.1 virtual void gazebo::SystemPlugin::Init () [inline],[virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.166.3.2 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.

10.166.3.3 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:

- **common/Plugin.hh**

10.167 gazebo::common::Time Class Reference

A (p. 109) **Time** (p. 776) 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 **Time NSleep** (**Time** _time) **GAZEBO_DEPRECATED**
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**
Microseconds.
- int32_t **sec**
Seconds.

Static Public Attributes

- static const **Time Zero**
A (p. 109) 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.

10.167.1 Detailed Description

A (p. 109) **Time** (p. 776) class, can be used to hold wall- or sim-time. stored as sec and nano-sec.

10.167.2 Constructor & Destructor Documentation

10.167.2.1 gazebo::common::Time::Time ()

Constructors.

10.167.2.2 gazebo::common::Time::Time (const Time & *time*)

Copy constructor.

Parameters

in	<i>time</i>	Time (p. 776) to copy
----	-------------	------------------------------

10.167.2.3 gazebo::common::Time::Time (const struct timeval & *_tv*)

Constructor.

Parameters

in	<i>_tv</i>	Time (p. 776) to initialize to
----	------------	---------------------------------------

10.167.2.4 gazebo::common::Time::Time (const struct timespec & *_tv*)

Constructor.

Parameters

in	<i>_tv</i>	Time (p. 776) to initialize to
----	------------	---------------------------------------

10.167.2.5 gazebo::common::Time::Time (int32_t *_sec*, int32_t *_nsec*)

Constructor.

Parameters

in	<i>_sec</i>	Seconds
in	<i>_nsec</i>	Nanoseconds

10.167.2.6 gazebo::common::Time::Time (double *time*)

Constuctor.

Parameters

in	<i>time</i>	Time (p. 776) in double format sec.nsec
----	-------------	--

10.167.2.7 virtual gazebo::common::Time::~~Time () [virtual]

Destructor.

10.167.3 Member Function Documentation

10.167.3.1 double gazebo::common::Time::Double () const

Get the time as a double.

Returns

Time (p. 776) as a double in seconds

10.167.3.2 float gazebo::common::Time::Float () const

Get the time as a float.

Returns

Time (p. 776) as a float in seconds

10.167.3.3 static const Time& gazebo::common::Time::GetWallTime () [static]

Get the wall time.

Returns

the current time

10.167.3.4 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.

10.167.3.5 static double gazebo::common::Time::MicToNano (double *_ms*) [inline],[static]

Convert microseconds to nanoseconds.

Parameters

<i>_ms</i>	microseconds
------------	--------------

Returns

nanoseconds

10.167.3.6 `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

10.167.3.7 `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. 776) actually slept

10.167.3.8 `static Time gazebo::common::Time::NSleep (unsigned int _ns) [static]`

Nano sleep.

Parameters

<code>in</code>	<code><i>_ns</i></code>	nanoseconds
-----------------	-------------------------	-------------

Returns

Time (p. 776) actually slept

10.167.3.9 `static Time gazebo::common::Time::NSleep (Time _time) [static]`

Nano sleep.

Parameters

<code>in</code>	<code><i>_time</i></code>	is a Time (p. 776)
-----------------	---------------------------	---------------------------

Returns

Time (p. 776) actually slept

10.167.3.10 `bool gazebo::common::Time::operator!=(const struct timeval & _tv) const`

Equal to operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare to
-----------------	------------------	------------------------

Returns

true if values are the same, false otherwise

10.167.3.11 `bool gazebo::common::Time::operator!=(const struct timespec & _tv) const`

Equal to operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare to
-----------------	------------------	------------------------

Returns

true if values are the same, false otherwise

10.167.3.12 `bool gazebo::common::Time::operator!=(const Time & _time) const`

Equal to operator.

Parameters

<code>in</code>	<code>_time</code>	the time to compare to
-----------------	--------------------	------------------------

Returns

true if values are the same, false otherwise

10.167.3.13 `bool gazebo::common::Time::operator!=(double _time) const`

Equal to operator.

Parameters

<code>in</code>	<code>_time</code>	the time to compare to
-----------------	--------------------	------------------------

Returns

true if values are the same, false otherwise

10.167.3.14 **Time** gazebo::common::Time::operator* (const struct timeval & *_tv*) const

Multiplication operator.

Parameters

<i>in</i>	<i>_tv</i>	The scaling duration
-----------	------------	----------------------

Returns

Time (p. 776) instance

10.167.3.15 **Time** gazebo::common::Time::operator* (const struct timespec & *_tv*) const

Multiplication operator.

Parameters

<i>in</i>	<i>_tv</i>	the scaling duration
-----------	------------	----------------------

Returns

Time (p. 776) instance

10.167.3.16 **Time** gazebo::common::Time::operator* (const Time & *_time*) const

Multiplication operators.

Parameters

<i>in</i>	<i>_time</i>	the scaling factor
-----------	--------------	--------------------

Returns

a scaled **Time** (p. 776) instance

10.167.3.17 **const Time&** gazebo::common::Time::operator*=(const struct timeval & *_tv*)

Multiplication assignment operator.

Parameters

<i>in</i>	<i>_tv</i>	the scaling duration
-----------	------------	----------------------

Returns

a reference to this instance

10.167.3.18 `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

10.167.3.19 `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. 776) instance

10.167.3.20 `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. 776) instance

10.167.3.21 `Time gazebo::common::Time::operator+(const struct timespec & _tv) const`

Addition operators.

Parameters

<code>in</code>	<code>_tv</code>	the time to add
-----------------	------------------	-----------------

Returns

a **Time** (p. 776) instance

10.167.3.22 `Time gazebo::common::Time::operator+ (const Time & _time) const`

Addition operators.

Parameters

<code>in</code>	<code>_time</code>	The time to add
-----------------	--------------------	-----------------

Returns

a **Time** (p. 776) instance

10.167.3.23 `const Time& gazebo::common::Time::operator+= (const struct timeval & _tv)`

Addition assignment operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to add
-----------------	------------------	-----------------

Returns

a reference to this instance

10.167.3.24 `const Time& gazebo::common::Time::operator+= (const struct timespec & _tv)`

Addition assignment operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to add
-----------------	------------------	-----------------

Returns

a reference to this instance

10.167.3.25 `const Time& gazebo::common::Time::operator+= (const Time & _time)`

Addition assignemtn operator.

Parameters

<code>in</code>	<code>_time</code>	The time to add
-----------------	--------------------	-----------------

Returns

a **Time** (p. 776) instance

10.167.3.26 Time gazebo::common::Time::operator- (const struct timeval & *_tv*) const

Subtraction operator.

Parameters

<i>in</i>	<i>_tv</i>	The time to subtract
-----------	------------	----------------------

Returns

a **Time** (p. 776) instance

10.167.3.27 Time gazebo::common::Time::operator- (const struct timespec & *_tv*) const

Subtraction operator.

Parameters

<i>in</i>	<i>_tv</i>	The time to subtract
-----------	------------	----------------------

Returns

a **Time** (p. 776) instance

10.167.3.28 Time gazebo::common::Time::operator- (const Time & *_time*) const

Subtraction operator.

Parameters

<i>in</i>	<i>_time</i>	The time to subtract
-----------	--------------	----------------------

Returns

a **Time** (p. 776) instance

10.167.3.29 const Time& gazebo::common::Time::operator-= (const struct timeval & *_tv*)

Subtraction assignment operator.

Parameters

<i>in</i>	<i>_tv</i>	The time to subtract
-----------	------------	----------------------

Returns

a **Time** (p. 776) instance

10.167.3.30 `const Time& gazebo::common::Time::operator-= (const struct timespec & _tv)`

Subtraction assignment operator.

Parameters

<code>in</code>	<code>_tv</code>	The time to subtract
-----------------	------------------	----------------------

Returns

a **Time** (p. 776) instance

10.167.3.31 `const Time& gazebo::common::Time::operator-= (const Time & _time)`

Subtraction assignment operator.

Parameters

<code>in</code>	<code>_time</code>	The time to subtract
-----------------	--------------------	----------------------

Returns

a reference to this instance

10.167.3.32 `Time gazebo::common::Time::operator/ (const struct timeval & _tv) const`

Division operator.

Parameters

<code>in</code>	<code>_tv</code>	a timeval divisor
-----------------	------------------	-------------------

Returns

a **Time** (p. 776) instance

10.167.3.33 `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. 776) instance

10.167.3.34 `Time gazebo::common::Time::operator/ (const Time & _time) const`

Division operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the divisor
-----------------	---------------------------	-------------

Returns

a **Time** (p. 776) instance

10.167.3.35 `const Time& gazebo::common::Time::operator/= (const struct timeval & _tv)`

Division assignment operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	a divisor
-----------------	-------------------------	-----------

Returns

a **Time** (p. 776) instance

10.167.3.36 `const Time& gazebo::common::Time::operator/= (const struct timespec & _tv)`

Division assignment operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	a divisor
-----------------	-------------------------	-----------

Returns

a **Time** (p. 776) instance

10.167.3.37 `const Time& gazebo::common::Time::operator/= (const Time & time)`

Division assignment operator.

Parameters

<code>in</code>	<code><i>time</i></code>	the divisor
-----------------	--------------------------	-------------

Returns

a **Time** (p. 776) instance

10.167.3.38 `bool gazebo::common::Time::operator< (const struct timeval & _tv) const`

Less than operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if tv is shorter than this, false otherwise

10.167.3.39 `bool gazebo::common::Time::operator< (const struct timespec & _tv) const`

Less than operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if tv is shorter than this, false otherwise

10.167.3.40 `bool gazebo::common::Time::operator< (const Time & _time) const`

Less than operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is shorter than this, false otherwise

10.167.3.41 `bool gazebo::common::Time::operator< (double _time) const`

Less than operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is shorter than this, false otherwise

10.167.3.42 `bool gazebo::common::Time::operator<= (const struct timeval & _tv) const`

Less than or equal to operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare with
-----------------	------------------	--------------------------

Returns

true if tv is shorter than or equal to this, false otherwise

10.167.3.43 `bool gazebo::common::Time::operator<= (const struct timespec & _tv) const`

Less than or equal to operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare with
-----------------	------------------	--------------------------

Returns

true if tv is shorter than or equal to this, false otherwise

10.167.3.44 `bool gazebo::common::Time::operator<= (const Time & _time) const`

Less than or equal to operator.

Parameters

<code>in</code>	<code>_time</code>	the time to compare with
-----------------	--------------------	--------------------------

Returns

true if time is shorter than or equal to this, false otherwise

10.167.3.45 `bool gazebo::common::Time::operator<= (double _time) const`

Less than or equal to operator.

Parameters

<code>in</code>	<code>_time</code>	the time to compare with
-----------------	--------------------	--------------------------

Returns

true if time is shorter than or equal to this, false otherwise

10.167.3.46 **Time& gazebo::common::Time::operator= (const struct timeval & *_tv*)**

Assignment operator.

Parameters

<i>in</i>	<i>_tv</i>	the new time
-----------	------------	--------------

Returns

a reference to this instance

10.167.3.47 **Time& gazebo::common::Time::operator= (const struct timespec & *_tv*)**

Assignment operator.

Parameters

<i>in</i>	<i>_tv</i>	the new time
-----------	------------	--------------

Returns

a reference to this instance

10.167.3.48 **Time& gazebo::common::Time::operator= (const Time & *_time*)**

Assignment operator.

Parameters

<i>in</i>	<i>_time</i>	the new time
-----------	--------------	--------------

Returns

a reference to this instance

10.167.3.49 **bool gazebo::common::Time::operator== (const struct timeval & *_tv*) const**

Equal to operator.

Parameters

<i>in</i>	<i>_tv</i>	the time to compare to
-----------	------------	------------------------

Returns

true if values are the same, false otherwise

10.167.3.50 `bool gazebo::common::Time::operator==(const struct timespec & _tv) const`

Equal to operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare to
-----------------	------------------	------------------------

Returns

true if values are the same, false otherwise

10.167.3.51 `bool gazebo::common::Time::operator==(const Time & _time) const`

Equal to operator.

Parameters

<code>in</code>	<code>_time</code>	the time to compare to
-----------------	--------------------	------------------------

Returns

true if values are the same, false otherwise

10.167.3.52 `bool gazebo::common::Time::operator==(double _time) const`

Equal to operator.

Parameters

<code>in</code>	<code>_time</code>	the time to compare to
-----------------	--------------------	------------------------

Returns

true if values are the same, false otherwise

10.167.3.53 `bool gazebo::common::Time::operator>(const struct timeval & _tv) const`

Greater than operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare with
-----------------	------------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.167.3.54 `bool gazebo::common::Time::operator> (const struct timespec & _tv) const`

Greater than operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.167.3.55 `bool gazebo::common::Time::operator> (const Time & _time) const`

Greater than operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.167.3.56 `bool gazebo::common::Time::operator> (double _time) const`

Greater than operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.167.3.57 `bool gazebo::common::Time::operator>= (const struct timeval & _tv) const`

Greater than or equal operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if tv is greater than or equal to this, false otherwise

10.167.3.58 `bool gazebo::common::Time::operator>= (const struct timespec & _tv) const`

Greater than or equal operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if *tv* is greater than or equal to this, false otherwise

10.167.3.59 `bool gazebo::common::Time::operator>= (const Time & _time) const`

Greater than or equal operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if *time* is greater than or equal to this, false otherwise

10.167.3.60 `bool gazebo::common::Time::operator>= (double _time) const`

Greater than or equal operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if *time* is greater than or equal to this, false otherwise

10.167.3.61 `static double gazebo::common::Time::SecToNano (double _sec) [inline],[static]`

Convert seconds to nanoseconds.

Parameters

<code>in</code>	<code><i>_sec</i></code>	duration in seconds
-----------------	--------------------------	---------------------

Returns

nanoseconds

10.167.3.62 `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

10.167.3.63 `void gazebo::common::Time::Set (double _seconds)`

Set to seconds.

Parameters

in	<code>_seconds</code>	Number of seconds
----	-----------------------	-------------------

10.167.3.64 `void gazebo::common::Time::SetToWallTime ()`

Set the time to the wall time.

10.167.3.65 `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. 776) actually slept

10.167.4 Friends And Related Function Documentation

10.167.4.1 `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

10.167.4.2 `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

10.167.5 Member Data Documentation

10.167.5.1 `int32_t gazebo::common::Time::nsec`

Microseconds.

10.167.5.2 `int32_t gazebo::common::Time::sec`

Seconds.

10.167.5.3 `const Time gazebo::common::Time::Zero` [*static*]

A (p. 109) static zero time variable set to `common::Time(0, 0)`.

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

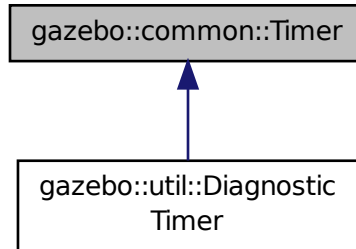
- **Time.hh**

10.168 gazebo::common::Timer Class Reference

A (p. 109) 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.

10.168.1 Detailed Description

A (p. 109) timer class, used to time things in real world walltime.

10.168.2 Constructor & Destructor Documentation

10.168.2.1 gazebo::common::Timer::Timer ()

Constructor.

10.168.2.2 `virtual gazebo::common::Timer::~~Timer () [virtual]`

Destructor.

10.168.3 Member Function Documentation

10.168.3.1 `Time gazebo::common::Timer::GetElapsed () const`

Get the elapsed time.

Returns

The time

10.168.3.2 `bool gazebo::common::Timer::GetRunning () const`

Returns true if the timer is running.

Returns

True if the timer has been started and not stopped.

10.168.3.3 `virtual void gazebo::common::Timer::Start () [virtual]`

Start the timer.

Reimplemented in `gazebo::util::DiagnosticTimer` (p. 262).

10.168.3.4 `virtual void gazebo::common::Timer::Stop () [virtual]`

Stop the timer.

Reimplemented in `gazebo::util::DiagnosticTimer` (p. 262).

10.168.4 Friends And Related Function Documentation

10.168.4.1 `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:

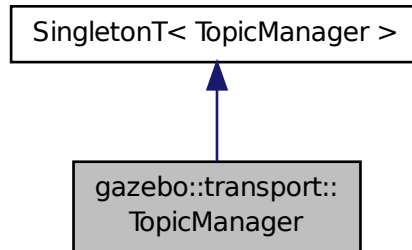
- `Timer.hh`

10.169 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 (p. 109) map of string->list of **Node** (p. 526) pointers.*

Public Member Functions

- void **AddNode** (**NodePtr** _node)
Add a node to the manager.
- 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. 219) a local **Publisher** (p. 606) to a remote **Subscriber** (p. 761).*
- 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. 761) to a remote **Publisher** (p. 606).*
- 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.
- std::map< std::string, std::list< std::string > > **GetAdvertisedTopics** () const **GAZEBO_DEPRECATED**

- Get a list of all the topics.*

 - void **GetTopicNamespaces** (std::list< std::string > &_namespaces)

Get all the topic namespaces.
- void **Init** ()

Initialize the manager.
- bool **IsAdvertised** (const std::string &_topic)

Has the topic been advertised?
- void **PauseIncoming** (bool _pause)

Pause or unpaue processing of incoming messages.
- void **ProcessNodes** (bool _onlyOut=false)

Process all nodes under management.
- void **Publish** (const std::string &_topic, const google::protobuf::Message &_message, const boost::function< void()> &_cb=NULL)

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

10.169.1 Detailed Description

Manages topics and their subscriptions.

10.169.2 Member Typedef Documentation

10.169.2.1 typedef std::map<std::string, std::list<NodePtr> > gazebo::transport::TopicManager::SubNodeMap

A (p. 109) map of string->list of **Node** (p. 526) pointers.

10.169.3 Member Function Documentation

10.169.3.1 void gazebo::transport::TopicManager::AddNode (NodePtr _node)

Add a node to the manager.

Parameters

in, out	<code>_node</code>	The node to be added
---------	--------------------	----------------------

10.169.3.2 `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. 606)

References `gazebo::transport::Publication::AddPublisher()`, `gazebo::transport::Publication::AddSubscription()`, `FindPublication()`, `gazebo::transport::Publication::GetLocallyAdvertised()`, `gzthrow`, `SingletonT< T >::Instance()`, `NULL`, `gazebo::transport::Publication::SetLocallyAdvertised()`, and `UpdatePublications()`.

10.169.3.3 `void gazebo::transport::TopicManager::ClearBuffers ()`

Clear all buffers.

10.169.3.4 `void gazebo::transport::TopicManager::ConnectPubToSub (const std::string & _topic, const SubscriptionTransportPtr _sublink)`

Connection (p. 219) a local **Publisher** (p. 606) to a remote **Subscriber** (p. 761).

Parameters

in	<i>_topic</i>	The topic to use
in	<i>_sublink</i>	The subscription transport object to use

10.169.3.5 `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
----	---------------	---------------------------

10.169.3.6 `void gazebo::transport::TopicManager::ConnectSubToPub (const msgs::Publish & _pub)`

Connect a local **Subscriber** (p. 761) to a remote **Publisher** (p. 606).

Parameters

in	<i>_pub</i>	The publish object to use
----	-------------	---------------------------

10.169.3.7 `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

10.169.3.8 `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

10.169.3.9 `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()`.

10.169.3.10 `void gazebo::transport::TopicManager::Fini ()`

Finalize the manager.

10.169.3.11 `std::map<std::string, std::list<std::string> > gazebo::transport::TopicManager::GetAdvertisedTopics () const`

Get a list of all the topics.

Returns

A (p. 109) map where keys are message types, and values are a list of topic names.

See Also

`transport::GetAdvertisedTopics`

10.169.3.12 `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
-----	--------------------------	---

10.169.3.13 `void gazebo::transport::TopicManager::Init ()`

Initialize the manager.

10.169.3.14 `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

10.169.3.15 `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
----	---------------------	--

10.169.3.16 `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
----	-----------------------	---

10.169.3.17 `void gazebo::transport::TopicManager::Publish (const std::string & _topic, const google::protobuf::Message & _message, const boost::function< void()> & _cb = NULL)`

Send a message.

Use a **Publisher** (p. 606) instead of calling this function directly.

Parameters

<code>_topic</code>	Name of the topic
<code>_message</code>	The message to send.
<code>_cb</code>	Callback, used when the publish is completed.

10.169.3.18 `void gazebo::transport::TopicManager::RegisterTopicNamespace (const std::string & _name)`

Register a new topic namespace.

Parameters

<code>in</code>	<code>_name</code>	The name of the new namespace
-----------------	--------------------	-------------------------------

10.169.3.19 `void gazebo::transport::TopicManager::RemoveNode (unsigned int _id)`

Remove a node by its id.

Parameters

<code>in</code>	<code>_id</code>	The ID of the node to be removed
-----------------	------------------	----------------------------------

10.169.3.20 `SubscriberPtr gazebo::transport::TopicManager::Subscribe (const SubscribeOptions & _options)`

Subscribe to a topic.

Parameters

<code>in</code>	<code>_options</code>	The options to use for the subscription
-----------------	-----------------------	---

Returns

Pointer to the newly created subscriber

10.169.3.21 `void gazebo::transport::TopicManager::Unadvertise (const std::string & _topic)`

Unadvertise a topic.

Parameters

<code>in</code>	<code>_topic</code>	The topic to be unadvertised
-----------------	---------------------	------------------------------

10.169.3.22 `void gazebo::transport::TopicManager::Unsubscribe (const std::string & _topic, const NodePtr & _sub)`

Unsubscribe from a topic.

Use a **Subscriber** (p. 761) rather than calling this function directly

Parameters

in	<code>_topic</code>	The topic to unsubscribe from
in	<code>_sub</code>	The node to unsubscribe

10.169.3.23 **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**

10.170 gazebo::physics::TrajectoryInfo Struct Reference

```
#include <Actor.hh>
```

Public Attributes

- double **duration**
- double **endTime**
- unsigned int **id**
- double **startTime**
- bool **translated**
- std::string **type**

10.170.1 Member Data Documentation

10.170.1.1 double gazebo::physics::TrajectoryInfo::duration

10.170.1.2 double gazebo::physics::TrajectoryInfo::endTime

10.170.1.3 unsigned int gazebo::physics::TrajectoryInfo::id

10.170.1.4 double gazebo::physics::TrajectoryInfo::startTime

10.170.1.5 bool gazebo::physics::TrajectoryInfo::translated

10.170.1.6 `std::string gazebo::physics::TrajectoryInfo::type`

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

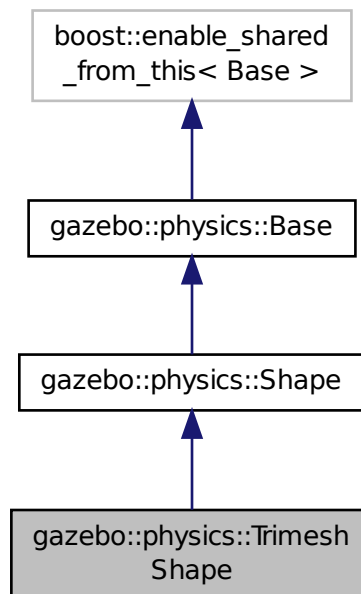
- **Actor.hh**

10.171 gazebo::physics::TrimeshShape Class Reference

Triangle mesh collision shape.

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

Inheritance diagram for gazebo::physics::TrimeshShape:



Public Member Functions

- **TrimeshShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~TrimeshShape** ()
Destructor.
- void **FillMsg** (msgs::Geometry &_msg)
Populate a msgs::Geometry message with data from this shape.
- std::string **GetFilename** () const
Get the filename 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 **SetFilename** (const std::string &_filename)
Set the filename of the triangle mesh.
- 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.

Additional Inherited Members

10.171.1 Detailed Description

Triangle mesh collision shape.

10.171.2 Constructor & Destructor Documentation

10.171.2.1 gazebo::physics::TrimeshShape::TrimeshShape (**CollisionPtr** *_parent*) [explicit]

Constructor.

Parameters

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

10.171.2.2 virtual gazebo::physics::TrimeshShape::~~TrimeshShape () [virtual]

Destructor.

10.171.3 Member Function Documentation

10.171.3.1 void gazebo::physics::TrimeshShape::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. 708).

10.171.3.2 `std::string gazebo::physics::TrimeshShape::GetFilename () const`

Get the filename of the mesh data.

Returns

The filename of the mesh data.

10.171.3.3 `virtual math::Vector3 gazebo::physics::TrimeshShape::GetSize () const [virtual]`

Get the size of the triangle mesh.

Returns

The size of the triangle mesh.

10.171.3.4 `virtual void gazebo::physics::TrimeshShape::Init () [virtual]`

Initialize the shape.

Implements **gazebo::physics::Shape** (p. 708).

10.171.3.5 `virtual void gazebo::physics::TrimeshShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]`

Update this shape from a message.

Parameters

<code>in</code>	<code>_msg</code>	Message that contains triangle mesh info.
-----------------	-------------------	---

Implements **gazebo::physics::Shape** (p. 708).

10.171.3.6 `void gazebo::physics::TrimeshShape::SetFilename (const std::string & _filename)`

Set the filename of the triangle mesh.

Parameters

<code>in</code>	<code>_filename</code>	Filename of the mesh file to load from.
-----------------	------------------------	---

10.171.3.7 `void gazebo::physics::TrimeshShape::SetScale (const math::Vector3 & _scale)`

Set the scaling factor.

Parameters

<code>in</code>	<code>_scale</code>	Scaling factor.
-----------------	---------------------	-----------------

10.171.3.8 `virtual void gazebo::physics::TrimeshShape::Update () [inline],[virtual]`

Update the tri mesh.

Reimplemented from `gazebo::physics::Base` (p. 146).

10.171.4 Member Data Documentation

10.171.4.1 `const common::Mesh* gazebo::physics::TrimeshShape::mesh [protected]`

Pointer to the mesh data.

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

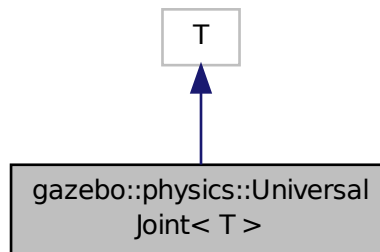
- `TrimeshShape.hh`

10.172 gazebo::physics::UniversalJoint< T > Class Template Reference

A (p. 109) universal joint.

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

Inheritance diagram for `gazebo::physics::UniversalJoint< T >`:



Public Member Functions

- `UniversalJoint (BasePtr _parent)`
Constructor.
- `virtual ~UniversalJoint ()`
Destructor.
- `virtual unsigned int GetAngleCount () const`
- `virtual void Load (sdf::ElementPtr _sdf)`
Load a UniversalJoint (p. 810).

10.172.1 Detailed Description

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

A (p. 109) universal joint.

10.172.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	_parent	Parent link of the univernal joint.
----	---------	-------------------------------------

References gazebo::physics::Base::UNIVERSAL_JOINT.

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

Destuctor.

10.172.3 Member Function Documentation

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

10.172.3.2 `template<class T > virtual void gazebo::physics::UniversalJoint< T >::Load (sdf::ElementPtr _sdf)`
`[inline], [virtual]`

Load a **UniversalJoint** (p. 810).

Parameters

in	_sdf	SDF values to load from.
----	------	--------------------------

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

- **UniversalJoint.hh**

10.173 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.

10.173.1 Detailed Description

Information for use in an update event.

10.173.2 Member Data Documentation

10.173.2.1 `common::Time gazebo::common::UpdateInfo::realTime`

Current real time.

10.173.2.2 `common::Time gazebo::common::UpdateInfo::simTime`

Current simulation time.

10.173.2.3 `std::string gazebo::common::UpdateInfo::worldName`

Name of the world.

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

- **UpdateInfo.hh**

10.174 urdf2gazebo::URDF2Gazebo Class Reference

```
#include <parser_urdf.hh>
```

Public Member Functions

- **URDF2Gazebo** ()
constructor
- **~URDF2Gazebo** ()
destructor
- `TiXmlDocument` **InitModelDoc** (`TiXmlDocument *` _xmlDoc)
convert urdf xml document string to sdf xml document
- `TiXmlDocument` **InitModelFile** (`const std::string &` _filename)
convert urdf file to sdf xml document
- `TiXmlDocument` **InitModelString** (`const std::string &` _urdfStr, `bool` _enforceLimits=true)
convert urdf string to sdf xml document, with option to enforce limits.

10.174.1 Constructor & Destructor Documentation

10.174.1.1 urdf2gazebo::URDF2Gazebo::URDF2Gazebo ()

constructor

10.174.1.2 urdf2gazebo::URDF2Gazebo::~~URDF2Gazebo ()

destructor

10.174.2 Member Function Documentation

10.174.2.1 TiXmlDocument urdf2gazebo::URDF2Gazebo::InitModelDoc (TiXmlDocument * *_xmlDoc*)

convert urdf xml document string to sdf xml document

Parameters

in	<i>_xmlDoc</i>	a tinyxml document containing the urdf model
----	----------------	--

Returns

a tinyxml document containing sdf of the model

10.174.2.2 TiXmlDocument urdf2gazebo::URDF2Gazebo::InitModelFile (const std::string & *_filename*)

convert urdf file to sdf xml document

Parameters

in	<i>_urdfStr</i>	a string containing filename of the urdf model
----	-----------------	--

Returns

a tinyxml document containing sdf of the model

10.174.2.3 TiXmlDocument urdf2gazebo::URDF2Gazebo::InitModelString (const std::string & *_urdfStr*, bool *_enforceLimits* = true)

convert urdf string to sdf xml document, with option to enforce limits.

Parameters

in	<i>_urdfStr</i>	a string containing model urdf
in	<i>_enforceLimits</i>	option to enforce joint limits

Returns

a tinyxml document containing sdf of the model

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

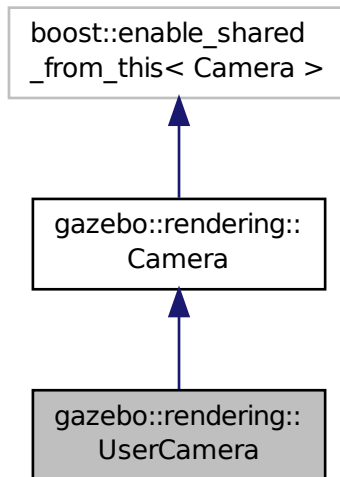
- **parser_urdf.hh**

10.175 gazebo::rendering::UserCamera Class Reference

A (p. 109) 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

10.175.1 Detailed Description

A (p. 109) camera used for user visualization of a scene.

10.175.2 Constructor & Destructor Documentation

10.175.2.1 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. 662) to put the camera in.

10.175.2.2 virtual gazebo::rendering::UserCamera::~UserCamera () [virtual]

Destructor.

10.175.3 Member Function Documentation

10.175.3.1 virtual void gazebo::rendering::UserCamera::AnimationComplete () [protected],[virtual]

Internal function used to indicate that an animation has completed.

Reimplemented from **gazebo::rendering::Camera** (p. 166).

10.175.3.2 virtual bool gazebo::rendering::UserCamera::AttachToVisualImpl (**VisualPtr** _visual, bool _inheritOrientation, double _minDist = 0, double _maxDist = 0) [protected],[virtual]

Set the camera to be attached to a visual.

This causes the camera to move in relation to the specified visual.

Parameters

in	<i>_visual</i>	The visual to attach to.
in	<i>_inherit-Orientation</i>	True if the camera should also rotate when the visual rotates.
in	<i>_minDist</i>	Minimum distance the camera can get to the visual.
in	<i>_maxDist</i>	Maximum distance the camera can get from the visual.

Returns

True if successfully attach to the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 167).

10.175.3.3 void gazebo::rendering::UserCamera::EnableViewController (bool *_value*) const

Set whether the view controller is enabled.

The view controller is used to handle user camera movements.

Parameters

in	<i>_value</i>	True to enable viewcontroller, False to disable.
----	---------------	--

10.175.3.4 void gazebo::rendering::UserCamera::Fini () [virtual]

Finalize.

Reimplemented from **gazebo::rendering::Camera** (p. 168).

10.175.3.5 float gazebo::rendering::UserCamera::GetAvgFPS () const

Get the average frames per second.

Returns

The average rendering frames per second

10.175.3.6 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. 341).

10.175.3.7 virtual unsigned int gazebo::rendering::UserCamera::GetImageHeight () const [virtual]

Get the height of the image.

Returns

Image height

Reimplemented from **gazebo::rendering::Camera** (p. 170).

10.175.3.8 `virtual unsigned int gazebo::rendering::UserCamera::GetImageWidth () const` [virtual]

Get the width of the image.

Returns

Image width

Reimplemented from **gazebo::rendering::Camera** (p. 170).

10.175.3.9 `float gazebo::rendering::UserCamera::GetTriangleCount () const`

Get the triangle count.

Returns

The number of triangles currently being rendered.

10.175.3.10 `std::string gazebo::rendering::UserCamera::GetViewControllerTypeString ()`

Get current view controller type.

Returns

Type of the current view controller: "orbit", "fps"

10.175.3.11 `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

10.175.3.12 `VisualPtr gazebo::rendering::UserCamera::GetVisual (const math::Vector2i & _mousePos) const`

Get a visual at a mouse position.

Parameters

in	<code>_mousePos</code>	2D position of the mouse in pixels.
----	------------------------	-------------------------------------

10.175.3.13 `void gazebo::rendering::UserCamera::HandleKeyPressEvent (const std::string & _key)`

Handle a key press.

Parameters

in	<code>_key</code>	The key pressed.
----	-------------------	------------------

10.175.3.14 `void gazebo::rendering::UserCamera::HandleKeyReleaseEvent (const std::string & _key)`

Handle a key release.

Parameters

in	<code>_key</code>	The key released.
----	-------------------	-------------------

10.175.3.15 `void gazebo::rendering::UserCamera::HandleMouseEvent (const common::MouseEvent & _evt)`

Handle a mouse event.

Parameters

in	<code>_evt</code>	The mouse event.
----	-------------------	------------------

10.175.3.16 `void gazebo::rendering::UserCamera::Init ()` [virtual]

Initialize.

Reimplemented from `gazebo::rendering::Camera` (p. 175).

10.175.3.17 `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. 176).

10.175.3.18 `void gazebo::rendering::UserCamera::Load ()` [virtual]

Generic load function.

Reimplemented from `gazebo::rendering::Camera` (p. 176).

10.175.3.19 `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. 177)

Parameters

<code>in</code>	<code>_pose</code>	End position of the camera
<code>in</code>	<code>_time</code>	Duration of the camera's movement

Reimplemented from **gazebo::rendering::Camera** (p. 176).

10.175.3.20 `void gazebo::rendering::UserCamera::MoveToVisual (VisualPtr _visual)`

Move the camera to focus on a visual.

Parameters

<code>in</code>	<code>_visual</code>	Visual (p. 868) to move the camera to.
-----------------	----------------------	---

10.175.3.21 `void gazebo::rendering::UserCamera::MoveToVisual (const std::string & _visualName)`

Move the camera to focus on a visual.

Parameters

<code>in</code>	<code>_visualName</code>	Name of the visual to move the camera to.
-----------------	--------------------------	---

10.175.3.22 `virtual void gazebo::rendering::UserCamera::PostRender ()` `[virtual]`

Post render.

Reimplemented from **gazebo::rendering::Camera** (p. 177).

10.175.3.23 `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.

10.175.3.24 `void gazebo::rendering::UserCamera::SetFocalPoint (const math::Vector3 & _pt)`

Set the point the camera should orbit around.

Parameters

in	<code>_pt</code>	The focal point
----	------------------	-----------------

10.175.3.25 `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

in	<code>_target</code>	The new rendering target.
----	----------------------	---------------------------

Reimplemented from `gazebo::rendering::Camera` (p. 180).

10.175.3.26 `void gazebo::rendering::UserCamera::SetViewController (const std::string & _type)`

Set view controller.

Parameters

in	<code>_type</code>	The type of view controller: "orbit", "fps"
----	--------------------	---

10.175.3.27 `void gazebo::rendering::UserCamera::SetViewController (const std::string & _type, const math::Vector3 & _pos)`

Set view controller.

Parameters

in	<code>_type</code>	The type of view controller: "orbit", "fps"
in	<code>_pos</code>	The initial pose of the camera.

10.175.3.28 `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.

10.175.3.29 `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. 180).

10.175.3.30 `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. 868) to track.
----	----------------------	----------------------------------

Returns

True if the camera is now tracking the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 182).

10.175.3.31 `virtual void gazebo::rendering::UserCamera::Update ()` [virtual]

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 182).

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

- **UserCamera.hh**

10.176 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.

10.176.1 Detailed Description

Generic double x, y vector.

10.176.2 Constructor & Destructor Documentation

10.176.2.1 `gazebo::math::Vector2d::Vector2d ()`

Constructor.

10.176.2.2 `gazebo::math::Vector2d::Vector2d (const double &_x, const double &_y)`

Constructor.

Parameters

<code>in</code>	<code>_x</code>	value along x
<code>in</code>	<code>_y</code>	value along y

10.176.2.3 `gazebo::math::Vector2d::Vector2d (const Vector2d &_v)`

Copy constructor.

Parameters

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

10.176.2.4 `virtual gazebo::math::Vector2d::~~Vector2d () [virtual]`

Destructor.

10.176.3 Member Function Documentation

10.176.3.1 `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

10.176.3.2 `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

10.176.3.3 `bool gazebo::math::Vector2d::IsFinite () const`

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.176.3.4 `void gazebo::math::Vector2d::Normalize ()`

Normalize the vector length.

10.176.3.5 `bool gazebo::math::Vector2d::operator!=(const Vector2d & _v) const`

Not equal to operator.

Returns

true if elements are of diffent values (tolerence 1e-6)

10.176.3.6 `const Vector2d gazebo::math::Vector2d::operator* (const Vector2d & _v) const`

Multiplication operators.

Parameters

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

Returns

the result

10.176.3.7 `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

10.176.3.8 `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

10.176.3.9 `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

10.176.3.10 `Vector2d gazebo::math::Vector2d::operator+ (const Vector2d & _v) const`

Addition operator.

Parameters

<code>in</code>	<code>_v</code>	vector to add
-----------------	-----------------	---------------

Returns

sum vector

10.176.3.11 `const Vector2d& gazebo::math::Vector2d::operator+=(const Vector2d & _v)`

Addition assignment operator.

Parameters

in	_v	the vector to add
----	----	-------------------

10.176.3.12 `Vector2d gazebo::math::Vector2d::operator-(const Vector2d & _v) const`

Subtraction operator.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

the subtracted vector

10.176.3.13 `const Vector2d& gazebo::math::Vector2d::operator-= (const Vector2d & _v)`

Subtraction assignment operator.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

this

10.176.3.14 `const Vector2d gazebo::math::Vector2d::operator/(const Vector2d & _v) const`

Division operator.

Remarks

this is an element wise division

Parameters

in	_v	a vector
----	----	----------

Returns

a result

10.176.3.15 `const Vector2d gazebo::math::Vector2d::operator/ (double _v) const`

Division operator.

Parameters

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

Returns

a vector

10.176.3.16 `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

10.176.3.17 `const Vector2d& gazebo::math::Vector2d::operator/= (double _v)`

Division operator.

Parameters

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

Returns

a vector

10.176.3.18 `Vector2d& gazebo::math::Vector2d::operator= (const Vector2d & _v)`

Assignment operator.

Parameters

in	_v	a value for x and y element
----	----	-----------------------------

Returns

this

10.176.3.19 `const Vector2d& gazebo::math::Vector2d::operator= (double _v)`

Assignment operator.

Parameters

in	_v	the value for x and y element
----	----	-------------------------------

Returns

this

10.176.3.20 `bool gazebo::math::Vector2d::operator== (const Vector2d & _v) const`

Equal to operator.

Parameters

in	_v	the vector to compare to
----	----	--------------------------

Returns

true if the elements of the 2 vectors are equal within a tolerance (1e-6)

10.176.3.21 `double gazebo::math::Vector2d::operator[] (unsigned int _index) const`

Array subscript operator.

Parameters

in	_index	the index
----	--------	-----------

Returns

the value, or 0 if _index is out of bounds

10.176.3.22 `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

10.176.4 Friends And Related Function Documentation

10.176.4.1 `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. 822) to output

Returns

The stream

10.176.4.2 `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. 839) to read values into

Returns

The stream

10.176.5 Member Data Documentation

10.176.5.1 `double gazebo::math::Vector2d::x`

x data

10.176.5.2 `double gazebo::math::Vector2d::y`

y data

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

- **Vector2d.hh**

10.177 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.

10.177.1 Detailed Description

Generic integer x, y vector.

10.177.2 Constructor & Destructor Documentation

10.177.2.1 gazebo::math::Vector2i::Vector2i ()

Constructor.

10.177.2.2 gazebo::math::Vector2i::Vector2i (const int & _x, const int & _y)

Constructor.

Parameters

in	_x	value along x
in	_y	value along y

10.177.2.3 gazebo::math::Vector2i::Vector2i (const Vector2i & _pt)

Copy onstructor.

Parameters

in	<code>_pt</code>	a point
----	------------------	---------

10.177.2.4 virtual gazebo::math::Vector2i::~~Vector2i () [virtual]

Destructor.

10.177.3 Member Function Documentation

10.177.3.1 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

10.177.3.2 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

10.177.3.3 bool gazebo::math::Vector2i::IsFinite () const

See if a point is finite (e.g., not nan)

Returns

the result

10.177.3.4 void gazebo::math::Vector2i::Normalize ()

Normalize the vector length.

10.177.3.5 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

10.177.3.6 `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

10.177.3.7 `const Vector2i gazebo::math::Vector2i::operator* (int _v) const`

Multiplication operator.

Parameters

<code>in</code>	<code>_v</code>	the scaling factor
-----------------	-----------------	--------------------

Returns

the result

10.177.3.8 `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

10.177.3.9 const Vector2i& gazebo::math::Vector2i::operator*=(int _v)

Multiplication operator.

Parameters

in	_v	scaling factor
----	----	----------------

Returns

this

10.177.3.10 Vector2i gazebo::math::Vector2i::operator+(const Vector2i & _v) const

Addition operator.

Parameters

in	_v	the vector to add
----	----	-------------------

Returns

the sum vector

10.177.3.11 const Vector2i& gazebo::math::Vector2i::operator+=(const Vector2i & _v)

Addition assignment operator.

Parameters

in	_v	the vector to add
----	----	-------------------

Returns

this

10.177.3.12 Vector2i gazebo::math::Vector2i::operator-(const Vector2i & _v) const

Subtraction operator.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

the result vector

10.177.3.13 `const Vector2i& gazebo::math::Vector2i::operator-= (const Vector2i & _v)`

Subtraction operators.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

this

10.177.3.14 `const Vector2i gazebo::math::Vector2i::operator/ (const Vector2i & _v) const`

Division operator.

Remarks

this is an element wise division.

Parameters

<code>in</code>	<code>_v</code>	the vector to divide
-----------------	-----------------	----------------------

Returns

the result

10.177.3.15 `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

10.177.3.16 `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

10.177.3.17 `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

10.177.3.18 `Vector2i& gazebo::math::Vector2i::operator= (const Vector2i & _v)`

Assignment operator.

Parameters

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

Returns

this

10.177.3.19 `const Vector2i& gazebo::math::Vector2i::operator= (int _value)`

Assignment operator.

Parameters

in	<i>_value</i>	the value for x and y
----	---------------	-----------------------

Returns

this

10.177.3.20 `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

10.177.3.21 `int gazebo::math::Vector2i::operator[](unsigned int _index) const`

Array subscript operator.

Parameters

in	<i>_index</i>	the array index
----	---------------	-----------------

10.177.3.22 `void gazebo::math::Vector2i::Set (int _x, int _y)`

Set the contents of the vector.

Parameters

in	<i>_x</i>	value along x
in	<i>_y</i>	value along y

10.177.4 Friends And Related Function Documentation

10.177.4.1 `std::ostream& operator<<(std::ostream & _out, const gazebo::math::Vector2i & _pt)` [*friend*]

Stream insertion operator.

Parameters

in	<i>_out</i>	output stream
in	<i>pt</i>	Vector2i (p. 830) to output

Returns

the stream

10.177.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector2i & _pt)` [friend]

Stream extraction operator.

Parameters

<code>in</code>	<code>_in</code>	input stream
<code>in</code>	<code>_pt</code>	Vector3 (p. 839) to read values into

Returns

The stream

10.177.5 Member Data Documentation

10.177.5.1 `int gazebo::math::Vector2i::x`

x data

10.177.5.2 `int gazebo::math::Vector2i::y`

y data

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

- **Vector2i.hh**

10.178 gazebo::math::Vector3 Class Reference

The **Vector3** (p. 839) 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 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.

10.178.1 Detailed Description

The **Vector3** (p. 839) 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.

10.178.2 Constructor & Destructor Documentation

10.178.2.1 gazebo::math::Vector3::Vector3 ()

Constructor.

Referenced by operator-().

10.178.2.2 gazebo::math::Vector3::Vector3 (const double & *_x*, const double & *_y*, const double & *_z*)

Constructor.

Parameters

<i>in</i>	<i>_x</i>	value along x
<i>in</i>	<i>_y</i>	value along y
<i>in</i>	<i>_z</i>	value along z

10.178.2.3 gazebo::math::Vector3::Vector3 (const Vector3 & *_v*)

Copy constructor.

Parameters

<i>in</i>	<i>_v</i>	a vector
-----------	-----------	----------

10.178.2.4 virtual gazebo::math::Vector3::~~Vector3 () [virtual]

Destructor.

10.178.3 Member Function Documentation

10.178.3.1 void gazebo::math::Vector3::Correct () [inline]

Corrects any nan values.

References x, y, and z.

Referenced by gazebo::math::Pose::Correct().

10.178.3.2 Vector3 gazebo::math::Vector3::Cross (const Vector3 & _pt) const

Return the cross product of this vector and pt.

Returns

the product

10.178.3.3 double gazebo::math::Vector3::Distance (const Vector3 & _pt) const

Calc distance to the given point.

Parameters

in	_pt	the point
----	-----	-----------

Returns

the distance

10.178.3.4 double gazebo::math::Vector3::Distance (double _x, double _y, double _z) const

Calc distance to the given point.

Parameters

in	_x	value along x
in	_y	value along y
in	_z	value along z

Returns

the distance

10.178.3.5 `double gazebo::math::Vector3::Dot (const Vector3 & _pt) const`

Return the dot product of this vector and pt.

Returns

the product

10.178.3.6 `bool gazebo::math::Vector3::Equal (const Vector3 & _v) const`

Equality test.

Remarks

This is equivalent to the == operator

Parameters

in	_v	the other vector
----	----	------------------

Returns

true if the 2 vectors have the same values, false otherwise

10.178.3.7 `Vector3 gazebo::math::Vector3::GetAbs () const`

Get the absolute value of the vector.

Returns

a vector with positive elements

10.178.3.8 `double gazebo::math::Vector3::GetDistToLine (const Vector3 & _pt1, const Vector3 & _pt2)`

Get distance to a line.

Parameters

in	_pt1	first point on the line
in	_pt2	second point on the line

Returns

the minimum distance from this point to the line

10.178.3.9 `double gazebo::math::Vector3::GetLength () const`

Returns the length (magnitude) of the vector \ return the length.

10.178.3.10 `double gazebo::math::Vector3::GetMax () const`

Get the maximum value in the vector.

Returns

the maximum element

10.178.3.11 `double gazebo::math::Vector3::GetMin () const`

Get the minimum value in the vector.

Returns

the minimum element

10.178.3.12 `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

10.178.3.13 `Vector3 gazebo::math::Vector3::GetPerpendicular () const`

Return a vector that is perpendicular to this one.

Returns

an orthogonal vector

10.178.3.14 `Vector3 gazebo::math::Vector3::GetRounded () const`

Get a rounded version of this vector.

Returns

a rounded vector

10.178.3.15 `double gazebo::math::Vector3::GetSquaredLength () const`

Return the square of the length (magnitude) of the vector.

Returns

the squared length

10.178.3.16 `double gazebo::math::Vector3::GetSum () const`

Return the sum of the values.

Returns

the sum

10.178.3.17 `bool gazebo::math::Vector3::IsFinite () const`

See if a point is finite (e.g., not nan)

10.178.3.18 `Vector3 gazebo::math::Vector3::Normalize ()`

Normalize the vector length.

Returns

unit length vector

10.178.3.19 `bool gazebo::math::Vector3::operator!=(const Vector3 & _v) const`

Not equal to operator.

Parameters

<code>in</code>	<code>_v</code>	The vector to compare against
-----------------	-----------------	-------------------------------

Returns

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.178.3.20 `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	
----	----	--

10.178.3.21 Vector3 gazebo::math::Vector3::operator* (double _v) const

Multiplication operators.

Parameters

in	_v	the scaling factor
----	----	--------------------

Returns

a scaled vector

10.178.3.22 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

10.178.3.23 const Vector3& gazebo::math::Vector3::operator*= (double _v)

Multiplication operator.

Parameters

in	_v	scaling factor
----	----	----------------

Returns

this

10.178.3.24 Vector3 gazebo::math::Vector3::operator+ (const Vector3 & _v) const

Addition operator.

Parameters

in	<code>_v</code>	vector to add
----	-----------------	---------------

Returns

the sum vector

10.178.3.25 `const Vector3& gazebo::math::Vector3::operator+=(const Vector3 & _v)`

Addition assignment operator.

Parameters

in	<code>_v</code>	vector to add
----	-----------------	---------------

10.178.3.26 `Vector3 gazebo::math::Vector3::operator-() const [inline]`

Negation operator.

Returns

negative of this vector

References Vector3(), x, y, and z.

10.178.3.27 `Vector3 gazebo::math::Vector3::operator-(const Vector3 & _pt) const [inline]`

Subtraction operators.

Parameters

in	<code>_pt</code>	a vector to subtract
----	------------------	----------------------

Returns

a vector

References Vector3(), x, y, and z.

10.178.3.28 `const Vector3& gazebo::math::Vector3::operator-=(const Vector3 & _pt)`

Subtraction operators.

Parameters

in	<code>_pt</code>	subtrahend
----	------------------	------------

10.178.3.29 `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

10.178.3.30 `const Vector3 gazebo::math::Vector3::operator/ (double _v) const`

Division operator.

Remarks

this is an element wise division

Returns

a vector

10.178.3.31 `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

10.178.3.32 `const Vector3& gazebo::math::Vector3::operator/= (double _v)`

Division operator.

Remarks

this is an element wise division

Returns

this

10.178.3.33 `Vector3& gazebo::math::Vector3::operator=(const Vector3 & _v)`

Assignment operator.

Parameters

<code>in</code>	<code>_v</code>	a new value
-----------------	-----------------	-------------

Returns

this

10.178.3.34 `Vector3& gazebo::math::Vector3::operator=(double _value)`

Assignment operator.

Parameters

<code>in</code>	<code>_value</code>	assigned to all elements
-----------------	---------------------	--------------------------

Returns

this

10.178.3.35 `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

10.178.3.36 `double gazebo::math::Vector3::operator[](unsigned int index) const`

[] operator

10.178.3.37 `Vector3 gazebo::math::Vector3::Round ()`

Round to near whole number, return the result.

Returns

the result

10.178.3.38 void gazebo::math::Vector3::Round (int *_precision*)

Round all values to *_precision* decimal places.

Parameters

in	<i>_precision</i>	the decimal places
----	-------------------	--------------------

10.178.3.39 void gazebo::math::Vector3::Set (double *_x* = 0, double *_y* = 0, double *_z* = 0) [inline]

Set the contents of the vector.

Parameters

in	<i>_x</i>	value along x
in	<i>_y</i>	value along y
in	<i>_z</i>	value along z

References *x*, *y*, and *z*.

10.178.3.40 void gazebo::math::Vector3::SetToMax (const Vector3 & *_v*)

Set this vector's components to the maximum of itself and the passed in vector.

Parameters

in	<i>_v</i>	the maximum clamping vector
----	-----------	-----------------------------

10.178.3.41 void gazebo::math::Vector3::SetToMin (const Vector3 & *_v*)

Set this vector's components to the minimum of itself and the passed in vector.

Parameters

in	<i>_v</i>	the minimum clamping vector
----	-----------	-----------------------------

10.178.4 Friends And Related Function Documentation

10.178.4.1 Vector3 operator* (double *_s*, const Vector3 & *_v*) [friend]

Multiplication operators.

Parameters

in	<i>_v</i>	the scaling factor
----	-----------	--------------------

Returns

a scaled vector

10.178.4.2 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector3 & _pt)` [*friend*]

Stream insertion operator.

Parameters

<code><i>_out</i></code>	output stream
<code><i>_pt</i></code>	Vector3 (p. 839) to output

Returns

the stream

10.178.4.3 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector3 & _pt)` [*friend*]

Stream extraction operator.

Parameters

<code><i>_in</i></code>	input stream
<code><i>_pt</i></code>	vector3 to read values into

Returns

the stream

10.178.5 Member Data Documentation

10.178.5.1 `double gazebo::math::Vector3::x`

X location.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `operator-`, `gazebo::math::Quaternion::RotateVector()`, and `Set()`.

10.178.5.2 `double gazebo::math::Vector3::y`

Y location.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `operator-`, `gazebo::math::Quaternion::RotateVector()`, and `Set()`.

10.178.5.3 `double gazebo::math::Vector3::z`

Z location.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `operator-`, `gazebo::math::Quaternion::RotateVector()`, and `Set()`.

10.178.5.4 `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**

10.179 gazebo::math::Vector4 Class Reference

```
double Generic x, y, z, w vector
#include <math/gzmath.hh>
```

Public Member Functions

- **Vector4** ()
Constructor.
- **Vector4** (const double &x, const double &y, const double &z, const double &w)
Constructor with component values.
- **Vector4** (const **Vector4** &v)
Copy constructor.
- virtual ~**Vector4** ()
Destructor.
- double **Distance** (const **Vector4** &pt) const
Calc distance to the given point.
- double **GetLength** () const
Returns the length (magnitude) of the vector.
- double **GetSquaredLength** () const
Return the square of the length (magnitude) of the vector.
- bool **IsFinite** () const
See if a point is finite (e.g., not nan)
- void **Normalize** ()
Normalize the vector length.
- bool **operator!=** (const **Vector4** &pt) const
Not equal to operator.
- const **Vector4 operator*** (const **Vector4** &pt) const
Multiplication operator.
- const **Vector4 operator*** (const **Matrix4** &m) const
Matrix multiplication operator.
- const **Vector4 operator*** (double _v) const
Multiplication operators.
- const **Vector4 & operator*=** (const **Vector4** &pt)
Multiplication assignment operator.
- const **Vector4 & operator*=** (double _v)
Multiplication assignment operator.
- **Vector4 operator+** (const **Vector4** &v) const
Addition operator.
- const **Vector4 & operator+=** (const **Vector4** &v)
Addition operator.
- **Vector4 operator-** (const **Vector4** &v) const

Subtraction operator.

- const **Vector4** & **operator-=** (const **Vector4** &_v)

Subtraction assignment operators.

- const **Vector4** **operator/** (const **Vector4** &_v) const

Division assignment operator.

- const **Vector4** **operator/** (double _v) const

Division assignment operator.

- const **Vector4** & **operator/=** (const **Vector4** &_v)

Division assignment operator.

- const **Vector4** & **operator/=** (double _v)

Division operator.

- **Vector4** & **operator=** (const **Vector4** &_v)

Assignment operator.

- **Vector4** & **operator=** (double _value)

Assignment operator.

- bool **operator==** (const **Vector4** &_pt) const

Equal to operator.

- double **operator[]** (unsigned int _index) const

Array subscript operator.

- void **Set** (double _x=0, double _y=0, double _z=0, double _w=0)

Set the contents of the vector.

Public Attributes

- double **w**

W value.

- double **x**

X value.

- double **y**

Y value.

- double **z**

Z value.

Friends

- std::ostream & **operator<<** (std::ostream &_out, const **gazebo::math::Vector4** &_pt)

Stream insertion operator.

- std::istream & **operator>>** (std::istream &_in, **gazebo::math::Vector4** &_pt)

Stream extraction operator.

10.179.1 Detailed Description

double Generic x, y, z, w vector

10.179.2 Constructor & Destructor Documentation

10.179.2.1 gazebo::math::Vector4::Vector4 ()

Constructor.

10.179.2.2 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

10.179.2.3 gazebo::math::Vector4::Vector4 (const Vector4 & _v)

Copy constructor.

Parameters

in	<code>_v</code>	vector
----	-----------------	--------

10.179.2.4 virtual gazebo::math::Vector4::~~Vector4 () [virtual]

Destructor.

10.179.3 Member Function Documentation

10.179.3.1 double gazebo::math::Vector4::Distance (const Vector4 & _pt) const

Calc distance to the given point.

Parameters

in	<code>_pt</code>	the point
----	------------------	-----------

Returns

the distance

10.179.3.2 double gazebo::math::Vector4::GetLength () const

Returns the length (magnitude) of the vector.

10.179.3.3 `double gazebo::math::Vector4::GetSquaredLength () const`

Return the square of the length (magnitude) of the vector.

Returns

the length

10.179.3.4 `bool gazebo::math::Vector4::IsFinite () const`

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.179.3.5 `void gazebo::math::Vector4::Normalize ()`

Normalize the vector length.

10.179.3.6 `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

10.179.3.7 `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

10.179.3.8 `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`

10.179.3.9 `const Vector4 gazebo::math::Vector4::operator* (double _v) const`

Multiplication operators.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

a scaled vector

10.179.3.10 `const Vector4& gazebo::math::Vector4::operator*=(const Vector4 & _pt)`

Multiplication assignment operator.

Remarks

Performs element wise multiplication, which has limited use.

Parameters

<code>in</code>	<code>_pt</code>	a vector
-----------------	------------------	----------

Returns

this

10.179.3.11 `const Vector4& gazebo::math::Vector4::operator*=(double _v)`

Multiplication assignment operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

this

10.179.3.12 `Vector4 gazebo::math::Vector4::operator+ (const Vector4 & _v) const`

Addition operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to add
-----------------	-----------------	-------------------

Returns

a sum vector

10.179.3.13 `const Vector4& gazebo::math::Vector4::operator+= (const Vector4 & _v)`

Addition operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to add
-----------------	-----------------	-------------------

Returns

this vector

10.179.3.14 `Vector4 gazebo::math::Vector4::operator- (const Vector4 & _v) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

a vector

10.179.3.15 `const Vector4& gazebo::math::Vector4::operator-= (const Vector4 & _v)`

Subtraction assignment operators.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

this vector

10.179.3.16 `const Vector4 gazebo::math::Vector4::operator/ (const Vector4 & _v) const`

Division assignment operator.

Remarks

Performs element wise division, which has limited use.

Parameters

<code>in</code>	<code>_v</code>	the vector to perform element wise division with
-----------------	-----------------	--

Returns

a result vector

10.179.3.17 `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

10.179.3.18 `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

10.179.3.19 `const Vector4& gazebo::math::Vector4::operator/= (double _v)`

Division operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

a vector

10.179.3.20 `Vector4& gazebo::math::Vector4::operator= (const Vector4 & _v)`

Assignment operator.

Parameters

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

Returns

a reference to this vector

10.179.3.21 `Vector4& gazebo::math::Vector4::operator= (double _value)`

Assignment operator.

Parameters

<code>in</code>	<code>_value</code>	
-----------------	---------------------	--

10.179.3.22 `bool gazebo::math::Vector4::operator==(const Vector4 & _pt) const`

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

10.179.3.23 `double gazebo::math::Vector4::operator[] (unsigned int _index) const`

Array subscript operator.

Parameters

<code>in</code>	<code><i>_index</i></code>	
-----------------	----------------------------	--

10.179.3.24 `void gazebo::math::Vector4::Set (double _x = 0, double _y = 0, double _z = 0, double _w = 0)`

Set the contents of the vector.

Parameters

<code>in</code>	<code><i>_x</i></code>	value along x axis
<code>in</code>	<code><i>_y</i></code>	value along y axis
<code>in</code>	<code><i>_z</i></code>	value along z axis
<code>in</code>	<code><i>_w</i></code>	value along w axis

10.179.4 Friends And Related Function Documentation

10.179.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector4 & _pt)` [*friend*]

Stream insertion operator.

Parameters

<code>in</code>	<code><i>_out</i></code>	output stream
<code>in</code>	<code><i>_pt</i></code>	Vector4 (p. 853) to output

Returns

The stream

10.179.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector4 & _pt)` [*friend*]

Stream extraction operator.

Parameters

<code>in</code>	<code><i>_in</i></code>	input stream
<code>in</code>	<code><i>_pt</i></code>	Vector4 (p. 853) to read values into

Returns

the stream

10.179.5 Member Data Documentation

10.179.5.1 double gazebo::math::Vector4::w

W value.

10.179.5.2 double gazebo::math::Vector4::x

X value.

10.179.5.3 double gazebo::math::Vector4::y

Y value.

10.179.5.4 double gazebo::math::Vector4::z

Z value.

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

- **Vector4.hh**

10.180 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.

10.180.1 Detailed Description

Handle video encoding and decoding using libavcodec.

10.180.2 Constructor & Destructor Documentation

10.180.2.1 gazebo::common::Video::Video ()

Constructor.

10.180.2.2 virtual gazebo::common::Video::~~Video () [virtual]

Destructor.

10.180.3 Member Function Documentation

10.180.3.1 int gazebo::common::Video::GetHeight () const

Get the height of the video in pixels.

Returns

the height

10.180.3.2 bool gazebo::common::Video::GetNextFrame (unsigned char ** *_buffer*)

Get the next frame of the video.

Parameters

out	<i>_img</i>	Image (p. 356) in which the frame is stored
-----	-------------	--

Returns

false if HAVE_FFmpeg is not defined, true otherwise

10.180.3.3 int gazebo::common::Video::GetWidth () const

Get the width of the video in pixels.

Returns

the width

10.180.3.4 bool gazebo::common::Video::Load (const std::string & *_filename*)

Load a video file.

Parameters

in	<i>_filename</i>	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:

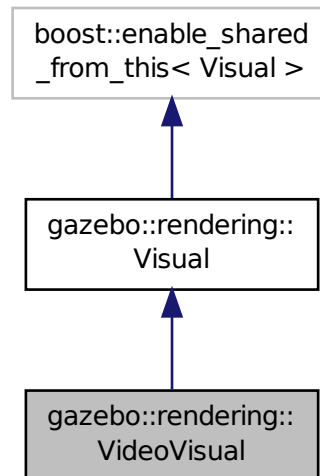
- **Video.hh**

10.181 gazebo::rendering::VideoVisual Class Reference

A (p. 109) 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

10.181.1 Detailed Description

A (p. 109) visual element that displays a video as a texture.

10.181.2 Constructor & Destructor Documentation

10.181.2.1 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.

10.181.2.2 virtual gazebo::rendering::VideoVisual::~~VideoVisual () [virtual]

Destructor.

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

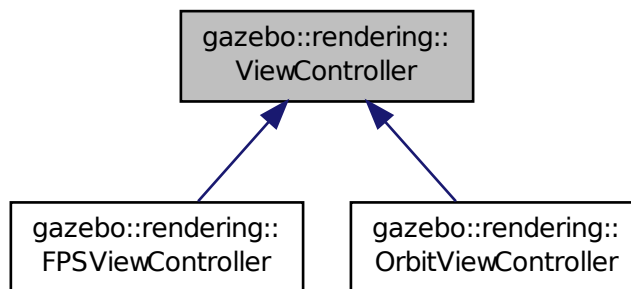
- **VideoVisual.hh**

10.182 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. 159).*

Protected Attributes

- **UserCameraPtr** **camera**
Pointer to the camera to control.
- bool **enabled**
True if enabled.
- std::string **typeString**
Type of view controller.

10.182.1 Detailed Description

Base class for view controllers.

10.182.2 Constructor & Destructor Documentation

10.182.2.1 gazebo::rendering::ViewController::ViewController (**UserCameraPtr** _camera)

Constructor.

Parameters

in	_camera	The user camera to controll.
----	---------	------------------------------

10.182.2.2 virtual gazebo::rendering::ViewController::~~ViewController () [virtual]

Destructor.

10.182.3 Member Function Documentation

10.182.3.1 `std::string gazebo::rendering::ViewController::GetTypeString () const`

Get the type of view controller.

Returns

The view controller type string.

10.182.3.2 `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.547), and **gazebo::rendering::FPSViewController** (p.318).

10.182.3.3 `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.548), and **gazebo::rendering::FPSViewController** (p.318).

10.182.3.4 `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.548), and **gazebo::rendering::FPSViewController** (p.318).

10.182.3.5 `virtual void gazebo::rendering::ViewController::Init () [pure virtual]`

Initialize the view controller.

Implemented in **gazebo::rendering::OrbitViewController** (p.548), and **gazebo::rendering::FPSViewController** (p.319).

10.182.3.6 `virtual void gazebo::rendering::ViewController::Init (const math::Vector3 & _focalPoint) [virtual]`

Initialize with a focus point.

Parameters

in	<code>_focalPoint</code>	The point to look at.
----	--------------------------	-----------------------

Reimplemented in `gazebo::rendering::OrbitViewController` (p. 548).

10.182.3.7 `void gazebo::rendering::ViewController::SetEnabled (bool _value)`

Set whether the controller is enabled.

Parameters

in	<code>_value</code>	True if the controller is enabled.
----	---------------------	------------------------------------

10.182.3.8 `virtual void gazebo::rendering::ViewController::Update () [pure virtual]`

Update the controller, which should update the position of the **Camera** (p. 159).

Implemented in `gazebo::rendering::OrbitViewController` (p. 549), and `gazebo::rendering::FPSViewController` (p. 319).

10.182.4 Member Data Documentation

10.182.4.1 `UserCameraPtr gazebo::rendering::ViewController::camera [protected]`

Pointer to the camera to control.

10.182.4.2 `bool gazebo::rendering::ViewController::enabled [protected]`

True if enabled.

10.182.4.3 `std::string gazebo::rendering::ViewController::typeString [protected]`

Type of view controller.

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

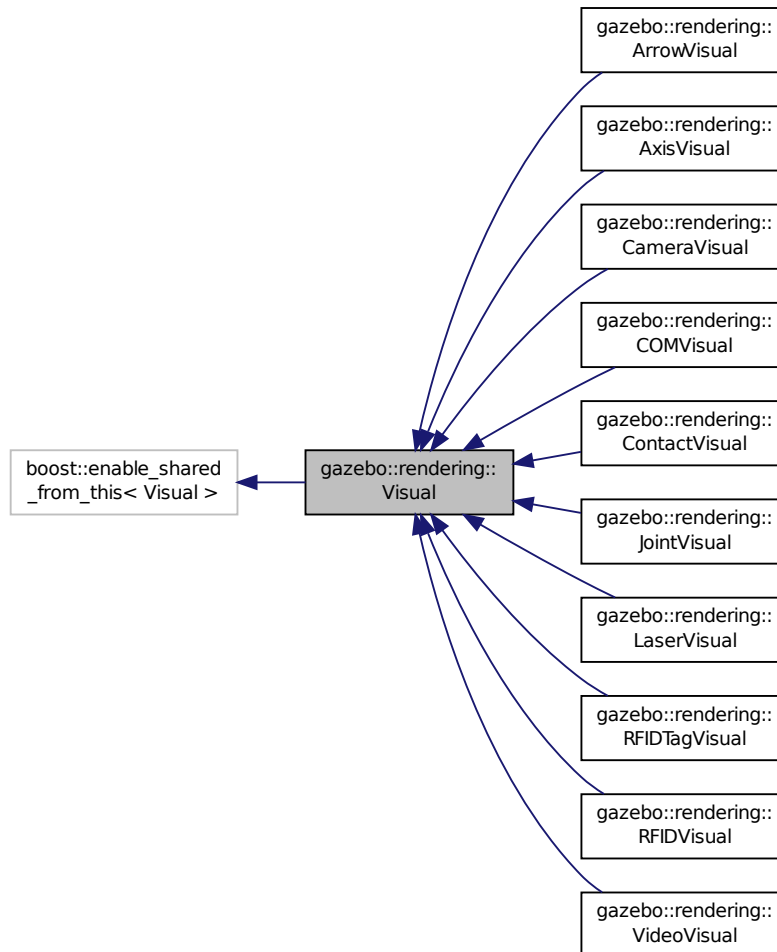
- **ViewController.hh**

10.183 gazebo::rendering::Visual Class Reference

A (p. 109) 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 &_objName="")

- Attach a mesh to this visual by name.*

 - void **AttachObject** (Ogre::MovableObject * _obj)
- Attach a renerable 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.*

 - 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.
- **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)**
*Insert a mesh into **Ogre** (p. 106).*
- **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 **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** &_changeColor)
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 **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)
*Insert a mesh into **Ogre** (p. 106).*

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. 106).*

10.183.1 Detailed Description

A (p. 109) renderable object.

10.183.2 Constructor & Destructor Documentation

10.183.2.1 gazebo::rendering::Visual::Visual (const std::string & *_name*, VisualPtr *_parent*, bool *_useRTShader* = true)

Constructor.

Parameters

in	<i>_name</i>	Name of the visual.
in	<i>_parent</i>	Parent of the visual.
in	<i>_useRTShader</i>	True if the visual should use the real-time shader system (RTShader).

10.183.2.2 `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. 662) containing the visual.
in	<code>_useRTShader</code>	True if the visual should use the real-time shader system (RTShader).

10.183.2.3 `virtual gazebo::rendering::Visual::~Visual () [virtual]`

Destructor.

10.183.3 Member Function Documentation

10.183.3.1 `void gazebo::rendering::Visual::AttachAxes ()`

Attach visualization axes.

10.183.3.2 `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.

10.183.3.3 `Ogre::MovableObject* gazebo::rendering::Visual::AttachMesh (const std::string & _meshName, const std::string & _objName = " ")`

Attach a mesh to this visual by name.

Parameters

in	<code>_meshName</code>	Name of the mesh.
in	<code>_objName</code>	Name of the attached Object to put the mesh onto.

10.183.3.4 `void gazebo::rendering::Visual::AttachObject (Ogre::MovableObject * _obj)`

Attach a renewable object to the visual.

Parameters

in	<code>_obj</code>	A (p. 109) movable object to attach to the visual.
----	-------------------	---

10.183.3.5 void gazebo::rendering::Visual::AttachVisual (VisualPtr _vis)

Attach a visual to this visual.

Parameters

in	_vis	Visual (p. 868) to attach.
----	------	----------------------------

10.183.3.6 void gazebo::rendering::Visual::ClearParent ()

Clear parents.

10.183.3.7 VisualPtr gazebo::rendering::Visual::Clone (const std::string & _name, VisualPtr _newParent)

Clone the visual with a new name.

Parameters

in	_name	Name of the cloned Visual (p. 868).
in	_newParent	Parent of the cloned Visual (p. 868).

Returns

The visual.

10.183.3.8 DynamicLines* gazebo::rendering::Visual::CreateDynamicLine (RenderOpType _type = RENDERING_LINE_STRIP)

Add a line to the visual.

Parameters

in	_type	The type of line to make.
----	-------	---------------------------

Returns

A (p. 109) pointer to the new dynamic line.

10.183.3.9 void gazebo::rendering::Visual::DeleteDynamicLine (DynamicLines * _line)

Delete a dynamic line.

Parameters

in	_line	Pointer to the line to delete.
----	-------	--------------------------------

10.183.3.10 void gazebo::rendering::Visual::DetachObjects ()

Detach all objects.

10.183.3.11 `void gazebo::rendering::Visual::DetachVisual (VisualPtr _vis)`

Detach a visual.

Parameters

<code>in</code>	<code>_vis</code>	Visual (p. 868) to detach.
-----------------	-------------------	-----------------------------------

10.183.3.12 `void gazebo::rendering::Visual::DetachVisual (const std::string & _name)`

Detach a visual.

Parameters

<code>in</code>	<code>_name</code>	Name of the visual to detach.
-----------------	--------------------	-------------------------------

10.183.3.13 `void gazebo::rendering::Visual::DisableTrackVisual ()`

Disable tracking of a visual.

10.183.3.14 `void gazebo::rendering::Visual::EnableTrackVisual (VisualPtr _vis)`

Set one visual to track/follow another.

Parameters

<code>in</code>	<code>_vis</code>	Visual (p. 868) to track.
-----------------	-------------------	----------------------------------

10.183.3.15 `void gazebo::rendering::Visual::Fini ()`

Helper for the destructor.

10.183.3.16 `unsigned int gazebo::rendering::Visual::GetAttachedObjectCount () const`

Return the number of attached movable objects.

Returns

The number of attached movable objects.

10.183.3.17 `math::Box gazebo::rendering::Visual::GetBoundingBox () const`

Get the bounding box for the visual.

Returns

The bounding box in world coordinates.

10.183.3.18 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. 877).

Parameters

<i>in</i>	<i>_index</i>	Index of the child to retrieve.
-----------	---------------	---------------------------------

Returns

Pointer to the child visual, NULL if index is invalid.

10.183.3.19 unsigned int gazebo::rendering::Visual::GetChildCount ()

Get the number of attached visuals.

Returns

The number of children.

10.183.3.20 std::string gazebo::rendering::Visual::GetMaterialName () const

Get the name of the material.

Returns

The name of the visual applied to this visual.

10.183.3.21 std::string gazebo::rendering::Visual::GetMeshName () const

The name of the mesh set in the visual's SDF.

Returns

Name of the mesh.

10.183.3.22 std::string gazebo::rendering::Visual::GetName () const

Get the name of the visual.

Returns

The name of the visual.

10.183.3.23 std::string gazebo::rendering::Visual::GetNormalMap () const

Get the normal map.

Returns

The name of the normal map material.

10.183.3.24 **VisualPtr** gazebo::rendering::Visual::GetParent () const

Get the parent visual, if one exists.

Returns

Pointer to the parent visual, NULL if no parent.

10.183.3.25 **math::Pose** gazebo::rendering::Visual::GetPose () const

Get the pose of the visual.

Returns

The **Visual** (p. 868)'s pose.

10.183.3.26 **math::Vector3** gazebo::rendering::Visual::GetPosition () const

Get the position of the visual.

Returns

The visual's position.

10.183.3.27 **VisualPtr** gazebo::rendering::Visual::GetRootVisual ()

Get the root visual.

Returns

The root visual, which is one level below the world visual.

10.183.3.28 **math::Quaternion** gazebo::rendering::Visual::GetRotation () const

Get the rotation of the visual.

Returns

The visual's rotation.

10.183.3.29 **math::Vector3** gazebo::rendering::Visual::GetScale ()

Get the scale.

Returns

The scaling factor.

10.183.3.30 `ScenePtr gazebo::rendering::Visual::GetScene () const`

Get current.

Returns

Pointer to the scene.

10.183.3.31 `Ogre::SceneNode* gazebo::rendering::Visual::GetSceneNode () const`

Return the scene Node of this visual entity.

Returns

The **Ogre** (p. 106) scene node.

10.183.3.32 `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".

10.183.3.33 `float gazebo::rendering::Visual::GetTransparency ()`

Get the transparency.

Returns

The transparency.

10.183.3.34 `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. 1049)

GZ_VISIBILITY_GUI (p. 1049)

GZ_VISIBILITY_NOT_SELECTABLE (p. 1049)

10.183.3.35 `bool gazebo::rendering::Visual::GetVisible () const`

Get whether the visual is visible.

Returns

True if the visual is visible.

10.183.3.36 `math::Pose gazebo::rendering::Visual::GetWorldPose () const`

Get the global pose of the node.

Returns

The pose in the world coordinate frame.

10.183.3.37 `bool gazebo::rendering::Visual::HasAttachedObject (const std::string & _name)`

Returns true if an object with `_name` is attached.

Parameters

<code>in</code>	<code>_name</code>	Name of an object to find.
-----------------	--------------------	----------------------------

10.183.3.38 `void gazebo::rendering::Visual::Init ()`

Helper for the constructor.

10.183.3.39 `void gazebo::rendering::Visual::InsertMesh (const std::string & _meshName)`

Insert a mesh into **Ogre** (p. 106).

Parameters

<code>in</code>	<code>_meshName</code>	Name of the mesh to insert.
-----------------	------------------------	-----------------------------

10.183.3.40 `static void gazebo::rendering::Visual::InsertMesh (const common::Mesh * _mesh) [static]`

Insert a mesh into **Ogre** (p. 106).

Parameters

<code>in</code>	<code>_mesh</code>	Pointer to the mesh to insert.
-----------------	--------------------	--------------------------------

10.183.3.41 `bool gazebo::rendering::Visual::IsPlane () const`

Return true if the visual is a plane.

Returns

True if a plane.

10.183.3.42 `bool gazebo::rendering::Visual::IsStatic () const`

Return true if the visual is a static geometry.

Returns

True if the visual is static.

10.183.3.43 `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.
-----------------	-------------------	---------------------------

10.183.3.44 `virtual void gazebo::rendering::Visual::Load () [virtual]`

Load the visual with default parameters.

Reimplemented in **gazebo::rendering::ArrowVisual** (p. 129), and **gazebo::rendering::AxisVisual** (p. 132).

10.183.3.45 `void gazebo::rendering::Visual::LoadFromMsg (ConstVisualPtr & _msg)`

Load from a message.

Parameters

<code>in</code>	<code>_msg</code>	A (p. 109) visual message.
-----------------	-------------------	-----------------------------------

10.183.3.46 `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 (p. 109) unique name for the plugin
<code>_sdf</code>	The SDF to pass into the plugin.

10.183.3.47 `void gazebo::rendering::Visual::MakeStatic ()`

Make the visual objects static renderables.

10.183.3.48 `void gazebo::rendering::Visual::MoveToPosition (const math::Pose & _pose, double _time)`

Move to a pose and over a given time.

Parameters

in	<code><i>_pose</i></code>	Pose the visual will end at.
in	<code><i>_time</i></code>	Time it takes the visual to move to the pose.

10.183.3.49 `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

in	<code><i>_poses</i></code>	Series of poses the visual will move to.
in	<code><i>_time</i></code>	Time it takes the visual to move to the pose.
in	<code><i>_onComplete</i></code>	Callback used when the move is complete.

10.183.3.50 `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
---------------------------	---

10.183.3.51 `void gazebo::rendering::Visual::SetAmbient (const common::Color & _color)`

Set the ambient color of the visual.

Parameters

in	<code><i>_color</i></code>	The ambient color.
----	----------------------------	--------------------

10.183.3.52 `void gazebo::rendering::Visual::SetCastShadows (bool _shadows)`

Set whether the visual should cast shadows.

Parameters

in	<code><i>_shadows</i></code>	True to enable shadows.
----	------------------------------	-------------------------

10.183.3.53 `void gazebo::rendering::Visual::SetDiffuse (const common::Color & _color)`

Set the diffuse color of the visual.

Parameters

in	<i>_color</i>	Set the diffuse color.
----	---------------	------------------------

10.183.3.54 `virtual void gazebo::rendering::Visual::SetEmissive (const common::Color & _color) [virtual]`

Set the emissive value.

Parameters

in	<i>_color</i>	The emissive color.
----	---------------	---------------------

Reimplemented in `gazebo::rendering::LaserVisual` (p. 405).

10.183.3.55 `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

in	<i>_highlighted</i>	True to enable the highlighting.
----	---------------------	----------------------------------

10.183.3.56 `void gazebo::rendering::Visual::SetMaterial (const std::string & _materialName, bool _unique = true)`

Set the material.

Parameters

in	<i>_materialName</i>	The name of the material.
in	<i>_unique</i>	True to make the material unique, which allows the material to change without changing materials that originally had the same name.

10.183.3.57 `void gazebo::rendering::Visual::SetName (const std::string & _name)`

Set the name of the visual.

Parameters

in	<i>_name</i>	Name of the visual
----	--------------	--------------------

10.183.3.58 `void gazebo::rendering::Visual::SetNormalMap (const std::string & _nmap)`

Set the normal map.

Parameters

in	<i>_nmap</i>	Name of the normal map material.
----	--------------	----------------------------------

10.183.3.59 `void gazebo::rendering::Visual::SetPose (const math::Pose & _pose)`

Set the pose of the visual.

Parameters

<i>in</i>	<i>_pose</i>	The new pose of the visual.
-----------	--------------	-----------------------------

10.183.3.60 `void gazebo::rendering::Visual::SetPosition (const math::Vector3 & _pos)`

Set the position of the visual.

Parameters

<i>in</i>	<i>_pos</i>	The position to set the visual to.
-----------	-------------	------------------------------------

10.183.3.61 `void gazebo::rendering::Visual::SetRibbonTrail (bool _value, const common::Color & _initialColor, const common::Color & _changeColor)`

True on or off a ribbon trail.

Parameters

<i>in</i>	<i>_value</i>	True to enable ribbon trail.
<i>in</i>	<i>_initialColor</i>	The initial color of the ribbon trail.
<i>in</i>	<i>_changeColor</i>	Color to change too as the trail grows.

10.183.3.62 `void gazebo::rendering::Visual::SetRotation (const math::Quaternion & _rot)`

Set the rotation of the visual.

Parameters

<i>in</i>	<i>_rot</i>	The rotation of the visual.
-----------	-------------	-----------------------------

10.183.3.63 `void gazebo::rendering::Visual::SetScale (const math::Vector3 & _scale)`

Set the scale.

Parameters

<i>in</i>	<i>_scale</i>	The scaling factor for the visual.
-----------	---------------	------------------------------------

10.183.3.64 `void gazebo::rendering::Visual::SetScene (ScenePtr _scene)`

Set current scene.

Parameters

<i>in</i>	<i>_scene</i>	Pointer to the scene.
-----------	---------------	-----------------------

10.183.3.65 `void gazebo::rendering::Visual::SetShaderType (const std::string & .type)`

Set the shader type for the visual's material.

Parameters

<i>in</i>	<i>_type</i>	Shader type string: "vertex", "pixel", "normal_map_object_space", "normal_map_tangent_space".
-----------	--------------	---

10.183.3.66 `void gazebo::rendering::Visual::SetSkeletonPose (const msgs::PoseAnimation & .pose)`

Set animation skeleton pose.

Parameters

<i>in</i>	<i>_pose</i>	Skelton message
-----------	--------------	-----------------

10.183.3.67 `void gazebo::rendering::Visual::SetSpecular (const common::Color & _color)`

Set the specular color of the visual.

Parameters

<i>in</i>	<i>_color</i>	Specular color.
-----------	---------------	-----------------

10.183.3.68 `void gazebo::rendering::Visual::SetTransparency (float .trans)`

Set the transparency.

Parameters

<i>in</i>	<i>_trans</i>	The transparency, between 0 and 1 where 0 is no transparency.
-----------	---------------	---

10.183.3.69 `void gazebo::rendering::Visual::SetVisibilityFlags (uint32_t .flags)`

Set visibility flags for this visual and all children.

Parameters

<i>in</i>	<i>_flags</i>	The visibility flags.
-----------	---------------	-----------------------

See Also

GZ_VISIBILITY_ALL (p. 1049)

GZ_VISIBILITY_GUI (p. 1049)

GZ_VISIBILITY_NOT_SELECTABLE (p. 1049)

10.183.3.70 `void gazebo::rendering::Visual::SetVisible (bool _visible, bool _cascade = true)`

Set whether the visual is visible.

Parameters

<code>in</code>	<code><i>_visible</i></code>	set this node visible.
<code>in</code>	<code><i>_cascade</i></code>	setting this parameter in children too.

10.183.3.71 `void gazebo::rendering::Visual::SetWorldPose (const math::Pose _pose)`

Set the world pose of the visual.

Parameters

<code>in</code>	<code><i>_pose</i></code>	Pose of the visual in the world coordinate frame.
-----------------	---------------------------	---

10.183.3.72 `void gazebo::rendering::Visual::SetWorldPosition (const math::Vector3 & _pos)`

Set the world linear position of the visual.

Parameters

<code>in</code>	<code><i>_pose</i></code>	Position in the world coordinate frame.
-----------------	---------------------------	---

10.183.3.73 `void gazebo::rendering::Visual::SetWorldRotation (const math::Quaternion & _rot)`

Set the world orientation of the visual.

Parameters

<code>in</code>	<code><i>_rot</i></code>	Rotation in the world coordinate frame.
-----------------	--------------------------	---

10.183.3.74 `void gazebo::rendering::Visual::ShowBoundingBox ()`

Display the bounding box visual.

10.183.3.75 `void gazebo::rendering::Visual::ShowCollision (bool _show)`

Display the collision visuals.

Parameters

<code>in</code>	<code><i>_show</i></code>	True to show visuals labeled as collision objects.
-----------------	---------------------------	--

10.183.3.76 void gazebo::rendering::Visual::ShowCOM (bool *_show*)

Display Center of Mass visuals.

Parameters

<i>in</i>	<i>_show</i>	True to show center of mass visualizations.
-----------	--------------	---

10.183.3.77 void gazebo::rendering::Visual::ShowJoints (bool *_show*)

Display joint visuals.

Parameters

<i>in</i>	<i>_show</i>	True to show joint visualizations.
-----------	--------------	------------------------------------

10.183.3.78 void gazebo::rendering::Visual::ShowSkeleton (bool *_show*)

Display the skeleton visuals.

Parameters

<i>in</i>	<i>_show</i>	True to show skeleton visuals.
-----------	--------------	--------------------------------

10.183.3.79 void gazebo::rendering::Visual::ToggleVisible ()

Toggle whether this visual is visible.

10.183.3.80 void gazebo::rendering::Visual::Update ()

Update the visual.

10.183.3.81 void gazebo::rendering::Visual::UpdateFromMsg (ConstVisualPtr & *_msg*)

Update a visual based on a message.

Parameters

<i>in</i>	<i>_msg</i>	The visual message.
-----------	-------------	---------------------

10.183.4 Member Data Documentation

10.183.4.1 VisualPtr gazebo::rendering::Visual::parent [protected]

Parent visual.

10.183.4.2 `ScenePtr gazebo::rendering::Visual::scene` [protected]

Pointer to the visual's scene.

10.183.4.3 `Ogre::SceneNode* gazebo::rendering::Visual::sceneNode` [protected]

Pointer to the visual's scene node in **Ogre** (p. 106).

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

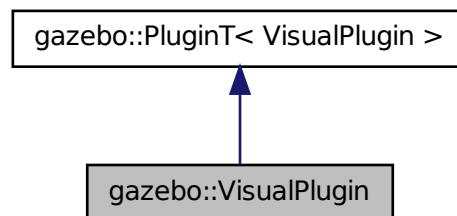
- **Visual.hh**

10.184 gazebo::VisualPlugin Class Reference

A (p. 109) 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

10.184.1 Detailed Description

A (p. 109) plugin loaded within the gzserver on startup.

See [reference](#).

10.184.2 Constructor & Destructor Documentation

10.184.2.1 `gazebo::VisualPlugin::VisualPlugin ()` `[inline]`

References `gazebo::PluginT< VisualPlugin >::type`, and `gazebo::VISUAL_PLUGIN`.

10.184.3 Member Function Documentation

10.184.3.1 `virtual void gazebo::VisualPlugin::Init ()` `[inline]`, `[virtual]`

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.184.3.2 `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.

10.184.3.3 `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:

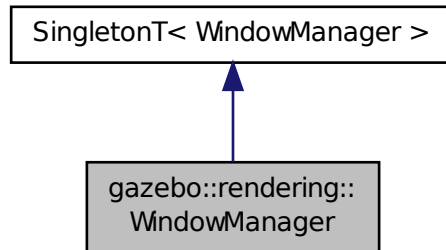
- `common/Plugin.hh`

10.185 gazebo::rendering::WindowManager Class Reference

Class to manage render windows.

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

Inheritance diagram for gazebo::rendering::WindowManager:



Public Member Functions

- 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. 106) 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.

Additional Inherited Members

10.185.1 Detailed Description

Class to manage render windows.

10.185.2 Member Function Documentation

- 10.185.2.1 int gazebo::rendering::WindowManager::CreateWindow (const std::string & _ogreHandle, uint32_t _width, uint32_t _height)

Create a window.

Parameters

in	<i>_ogreHandle</i>	String representing the ogre window handle.
in	<i>_width</i>	Width of the window in pixels.
in	<i>_height</i>	Height of the window in pixels.

10.185.2.2 void gazebo::rendering::WindowManager::Fini ()

Shutdown all the windows.

10.185.2.3 float gazebo::rendering::WindowManager::GetAvgFPS (uint32_t *_id*)

Get the average FPS.

Parameters

in	<i>_id</i>	ID of the window.
----	------------	-------------------

Returns

The frames per second.

10.185.2.4 uint32_t gazebo::rendering::WindowManager::GetTriangleCount (uint32_t *_id*)

Get the triangle count.

Parameters

in	<i>_id</i>	ID of the window.
----	------------	-------------------

Returns

The triangle count.

10.185.2.5 Ogre::RenderWindow* gazebo::rendering::WindowManager::GetWindow (uint32_t *_id*)

Get the render window associated with the given id.

Parameters

in	<i>_id</i>	ID of the window.
----	------------	-------------------

Returns

Pointer to the render window, NULL if the id is invalid.

10.185.2.6 void gazebo::rendering::WindowManager::Moved (uint32_t *_id*)

Tells **Ogre** (p. 106) the window has moved, and needs updating.

Parameters

in	<code>_id</code>	ID of the window.
----	------------------	-------------------

10.185.2.7 `void gazebo::rendering::WindowManager::Resize (uint32_t _id, int _width, int _height)`

Resize a window.

Parameters

in	<code>_id</code>	Id of the window to resize.
in	<code>_width</code>	New width of the window.
in	<code>_height</code>	New height of the window.

10.185.2.8 `void gazebo::rendering::WindowManager::SetCamera (int _windowId, CameraPtr _camera)`

Attach a camera to a window.

Parameters

in	<code>_windowId</code>	Id of the window to add the camera to.
in	<code>_camera</code>	Pointer to the camera to attach.

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

- **WindowManager.hh**

10.186 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.

10.186.1 Detailed Description

Draws a wireframe box.

10.186.2 Constructor & Destructor Documentation

10.186.2.1 gazebo::rendering::WireBox::WireBox (VisualPtr *_parent*, const math::Box & *_box*) [explicit]

Constructor.

Parameters

in	<i>_box</i>	Dimension of the box to draw.
----	-------------	-------------------------------

10.186.2.2 gazebo::rendering::WireBox::~~WireBox ()

Destructor.

10.186.3 Member Function Documentation

10.186.3.1 void gazebo::rendering::WireBox::Init (const math::Box & *_box*)

Builds the wireframe line list.

Parameters

in	<i>_box</i>	Box to build a wireframe from.
----	-------------	--------------------------------

10.186.3.2 void gazebo::rendering::WireBox::SetVisible (bool *_visible*)

Set the visibility of the box.

Parameters

in	<i>_visible</i>	True to make the box visible, False to hide.
----	-----------------	--

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

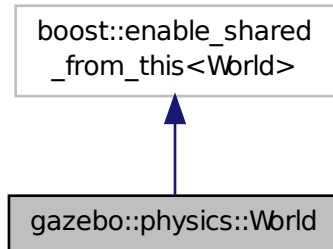
- **WireBox.hh**

10.187 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. 277) 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).
- **EntityPtr GetSelectedEntity** () const
*Get the selected **Entity** (p. 277).*
- 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. 781).*
- **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. 277) tree.*
- void **PublishModelPose** (const std::string &_modelName)
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** ()

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. 286) on the **physics::Link** (p. 412) in **World::Update**.*

10.187.1 Detailed Description

The world provides access to all other object within a simulated environment.

The **World** (p. 893) is the container for all models and their components (links, joints, sensors, plugins, etc), and **World-Plugin** (p. 905) instances. Many core function are also handled in the **World** (p. 893), including physics update, model updates, and message processing.

10.187.2 Constructor & Destructor Documentation

10.187.2.1 `gazebo::physics::World::World (const std::string & _name = " ") [explicit]`

Constructor.

Constructor for the **World** (p. 893). Must specify a unique name.

Parameters

in	_name	Name of the world.
----	-------	--------------------

10.187.2.2 gazebo::physics::World::~~World ()

Destructor.

10.187.3 Member Function Documentation

10.187.3.1 void gazebo::physics::World::Clear ()

Remove all entities from the world.

10.187.3.2 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.

10.187.3.3 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.

10.187.3.4 void gazebo::physics::World::EnablePhysicsEngine (bool *_enable*) [inline]

enable/disable physics engine during World::Update.

Parameters

in	<i>_enable</i>	True to enable the physics engine.
----	----------------	------------------------------------

10.187.3.5 void gazebo::physics::World::Fini ()

Finalize the world.

Call this function to tear-down the world.

10.187.3.6 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. 480) to find.
----	--------------	--

Returns

A (p. 109) pointer to the entity, or NULL if no entity was found.

10.187.3.7 `bool gazebo::physics::World::GetEnablePhysicsEngine () [inline]`

check if physics engine is enabled/disabled.

Parameters

	<i>True</i>	if the physics engine is enabled.
--	-------------	-----------------------------------

10.187.3.8 `EntityPtr gazebo::physics::World::GetEntity (const std::string & _name)`

Get a pointer to an **Entity** (p. 277) based on a name.

This function is the same as `GetByName`, but limits the search to only Entities.

Parameters

<i>in</i>	<i>_name</i>	The name of the Entity (p. 277) to find.
-----------	--------------	---

Returns

A (p. 109) pointer to the **Entity** (p. 277), or NULL if no **Entity** (p. 277) was found.

10.187.3.9 `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

<i>in</i>	<i>_pt</i>	The 3D point to search below
-----------	------------	------------------------------

Returns

A (p. 109) pointer to nearest **Entity** (p. 277), NULL if none is found.

10.187.3.10 `ModelPtr gazebo::physics::World::GetModel (unsigned int _index) const`

Get a model based on an index.

Get a **Model** (p. 480) using an index, where index must be greater than zero and less than `World::GetModelCount()` (p. 899)

Parameters

<i>in</i>	<i>_index</i>	The index of the model [0..GetModelCount)
-----------	---------------	---

Returns

A (p. 109) pointer to the **Model** (p. 480). NULL if *_index* is invalid.

10.187.3.11 `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

<code>in</code>	<code>_name</code>	The name of the Model (p. 480) to find.
-----------------	--------------------	--

Returns

A (p. 109) pointer to the **Model** (p. 480), or NULL if no model was found.

10.187.3.12 `ModelPtr gazebo::physics::World::GetModelBelowPoint (const math::Vector3 & _pt)`

Get the nearest model below a point.

This function makes use of `World::GetEntityBelowPoint` (p. 898).

Parameters

<code>in</code>	<code>_pt</code>	The 3D point to search below.
-----------------	------------------	-------------------------------

Returns

A (p. 109) pointer to nearest **Model** (p. 480), NULL if none is found.

10.187.3.13 `unsigned int gazebo::physics::World::GetModelCount () const`

Get the number of models.

Returns

The number of models in the **World** (p. 893).

10.187.3.14 `Model_V gazebo::physics::World::GetModels () const`

Get a list of all the models.

Returns

A (p. 109) list of all the Models in the world.

10.187.3.15 `std::string gazebo::physics::World::GetName () const`

Get the name of the world.

Returns

The name of the world.

10.187.3.16 `common::Time gazebo::physics::World::GetPauseTime () const`

Get the amount of time simulation has been paused.

Returns

The pause time.

10.187.3.17 `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.

10.187.3.18 `common::Time gazebo::physics::World::GetRealTime () const`

Get the real time (elapsed time).

Returns

The real time.

10.187.3.19 `EntityPtr gazebo::physics::World::GetSelectedEntity () const`

Get the selected **Entity** (p. 277).

The selected entity is set via the GUI.

Returns

A (p. 109) point to the **Entity** (p. 277), NULL if nothing is selected.

10.187.3.20 `boost::mutex* gazebo::physics::World::GetSetWorldPoseMutex () const [inline]`

Get the set world pose mutex.

Returns

Pointer to the mutex.

10.187.3.21 `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. 781).

Returns

The current simulation time

10.187.3.22 `common::Time gazebo::physics::World::GetStartTime () const`

Get the wall time simulation was started.

Returns

The start time.

10.187.3.23 `void gazebo::physics::World::Init ()`

Initialize the world.

This is called after Load.

10.187.3.24 `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).
-----------------	---------------------------	--

10.187.3.25 `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 (p. 109) reference to an SDF object.
-----------------	-------------------	---

10.187.3.26 `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 (p. 109) string containing valid SDF markup.
-----------------	-------------------------	---

10.187.3.27 `bool gazebo::physics::World::IsLoaded () const`

Return true if the world has been loaded.

Returns

True if **World::Load** (p. 902) has completed.

10.187.3.28 `bool gazebo::physics::World::IsPaused () const`

Returns the state of the simulation true if paused.

Returns

True if paused.

10.187.3.29 `void gazebo::physics::World::Load (sdf::ElementPtr _sdf)`

Load the world using SDF parameters.

Load a world from and SDF pointer.

Parameters

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

10.187.3.30 `void gazebo::physics::World::LoadPlugin (const std::string & _filename, const std::string & _name, sdf::ElementPtr _sdf)`

Load a plugin.

Parameters

in	<code>_filename</code>	The filename of the plugin.
in	<code>_name</code>	A (p. 109) unique name for the plugin.
in	<code>_sdf</code>	The SDF to pass into the plugin.

10.187.3.31 `void gazebo::physics::World::PrintEntityTree ()`

Print **Entity** (p. 277) tree.

Prints alls the entities to stdout.

10.187.3.32 `void gazebo::physics::World::PublishModelPose (const std::string & _modelName)`

Publish pose updates for a model.

This list of models to publish is processed and cleared once every iteration.

Parameters

in	<code>_modelName</code>	Name of the model to publish.
----	-------------------------	-------------------------------

10.187.3.33 `void gazebo::physics::World::RemovePlugin (const std::string & _name)`

Remove a running plugin.

Parameters

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

10.187.3.34 `void gazebo::physics::World::Reset ()`

Reset time and model poses, configurations in simulation.

10.187.3.35 `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. 138).

Parameters

in	<code>_type</code>	The type of reset.
----	--------------------	--------------------

10.187.3.36 `void gazebo::physics::World::ResetTime ()`

Reset simulation time back to zero.

10.187.3.37 `void gazebo::physics::World::Run ()`

Run the world in a thread.

Run the update loop.

10.187.3.38 `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

in	<code>_filename</code>	Name of the file to save into.
----	------------------------	--------------------------------

10.187.3.39 `void gazebo::physics::World::SetPaused (bool _p)`

Set whether the simulation is paused.

Parameters

in	<code>_p</code>	True pauses the simulation. False runs the simulation.
----	-----------------	--

10.187.3.40 `void gazebo::physics::World::SetSimTime (const common::Time & _t)`

Set the sim time.

Parameters

in	_t	The new simulation time
----	----	-------------------------

10.187.3.41 void gazebo::physics::World::SetState (const WorldState & _state)

Set the current world state.

Parameters

_state	The state to set the World (p. 893) to.
--------	--

10.187.3.42 void gazebo::physics::World::StepWorld (int _steps)

Step callback.

Parameters

in	_steps	The number of steps the World (p. 893) should take.
----	--------	--

10.187.3.43 void gazebo::physics::World::Stop ()

Stop the world.

Stop the update loop.

10.187.3.44 std::string gazebo::physics::World::StripWorldName (const std::string & _name) const

Return a version of the name with "<world_name>:" removed.

Parameters

in	_name	Usually the name of an entity.
----	-------	--------------------------------

Returns

The stripped world name.

10.187.3.45 void gazebo::physics::World::UpdateStateSDF ()

Update the state SDF value from the current state.

10.187.4 Member Data Documentation

10.187.4.1 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. 286) on the **physics::Link** (p. 412) in World::Update.

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

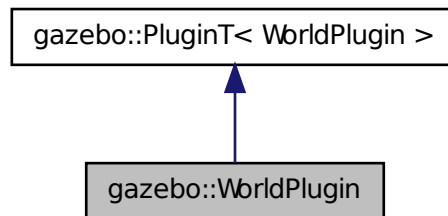
- [World.hh](#)

10.188 gazebo::WorldPlugin Class Reference

A (p. 109) plugin with access to **physics::World** (p. 893).

```
#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

10.188.1 Detailed Description

A (p. 109) plugin with access to **physics::World** (p. 893).

See [reference](#).

10.188.2 Constructor & Destructor Documentation

10.188.2.1 gazebo::WorldPlugin::WorldPlugin () [inline]

Constructor.

References [gazebo::PluginT< WorldPlugin >::type](#), and [gazebo::WORLD_PLUGIN](#).

10.188.2.2 `virtual gazebo::WorldPlugin::~~WorldPlugin () [inline],[virtual]`

Destructor.

10.188.3 Member Function Documentation

10.188.3.1 `virtual void gazebo::WorldPlugin::Init () [inline],[virtual]`

10.188.3.2 `virtual void gazebo::WorldPlugin::Load (physics::WorldPtr _world, sdf::ElementPtr _sdf) [pure virtual]`

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

in	<code>_world</code>	Pointer the World
in	<code>_sdf</code>	Pointer the the SDF element of the plugin.

10.188.3.3 `virtual void gazebo::WorldPlugin::Reset () [inline],[virtual]`

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

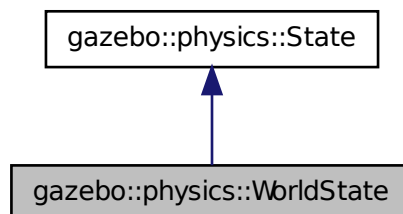
- `common/Plugin.hh`

10.189 gazebo::physics::WorldState Class Reference

Store state information of a `physics::World` (p. 893) 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** (unsigned int _index) const
Get a model state.
- **ModelState GetModelState** (const std::string &_modelName) const
Get a model state by model name.
- unsigned int **GetModelStateCount** () const
Get the number of model states.
- const std::vector< **ModelState** > & **GetModelStates** () const
Get the model states.
- bool **HasModelState** (const std::string &_modelName) const
*Return true if **WorldState** (p. 906) has a **ModelState** (p. 497) with the given name.*
- 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.
- **WorldState operator+** (const **WorldState** &_state) const
Addition operator.
- **WorldState operator-** (const **WorldState** &_state) const
Subtraction operator.
- **WorldState & operator=** (const **WorldState** &_state)
Assignment operator.
- 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

10.189.1 Detailed Description

Store state information of a **physics::World** (p. 893) object.

Instances of this class contain the state of a **World** (p. 893) at a specific time. **World** (p. 893) state includes the state of all models, and their children.

10.189.2 Constructor & Destructor Documentation

10.189.2.1 gazebo::physics::WorldState::WorldState ()

Default constructor.

10.189.2.2 gazebo::physics::WorldState::WorldState (const WorldPtr *_world*) [explicit]

Constructor.

Generate a **WorldState** (p. 906) from an instance of a **World** (p. 893).

Parameters

in	<i>_world</i>	Pointer to a world
----	---------------	--------------------

10.189.2.3 gazebo::physics::WorldState::WorldState (const sdf::ElementPtr *_sdf*) [explicit]

Constructor.

Build a **WorldState** (p. 906) from SDF data

Parameters

in	<i>_sdf</i>	SDF data to load a world state from.
----	-------------	--------------------------------------

10.189.2.4 virtual gazebo::physics::WorldState::~~WorldState () [virtual]

Destructor.

10.189.3 Member Function Documentation

10.189.3.1 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.
-----	-------------	--------------------------

10.189.3.2 ModelState gazebo::physics::WorldState::GetModelState (unsigned int *_index*) const

Get a model state.

Get the state of a **Model** (p. 480) based on an index. The min index is and the max is **WorldState::GetModelState-Count()** (p. 909).

Parameters

in	<i>_index</i>	Index of the model.
----	---------------	---------------------

Returns

State (p. 744) of the requested **Model** (p. 480).

10.189.3.3 **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. 314)	When the <i>_modelName</i> doesn't exist.
---	---

10.189.3.4 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.

10.189.3.5 const std::vector< **ModelState** > & gazebo::physics::WorldState::GetModelStates () const

Get the model states.

Returns

A (p. 109) vector of model states.

10.189.3.6 bool gazebo::physics::WorldState::HasModelState (const std::string & *_modelName*) const

Return true if **WorldState** (p. 906) has a **ModelState** (p. 497) with the given name.

Parameters

in	<i>_modelName</i>	Name of the model to search for.
----	-------------------	----------------------------------

Returns

True if the **ModelState** (p. 497) exists.

10.189.3.7 `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.

10.189.3.8 `virtual void gazebo::physics::WorldState::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Set a **WorldState** (p. 906) from an SDF element containing **WorldState** (p. 906) info.

Parameters

in	_elem	Pointer to the WorldState (p. 906) SDF element.
----	-------	--

Reimplemented from **gazebo::physics::State** (p. 747).

10.189.3.9 `WorldState gazebo::physics::WorldState::operator+ (const WorldState & _state) const`

Addition operator.

Parameters

in	_pt	A (p. 109) state to add.
----	-----	---------------------------------

Returns

The resulting state.

10.189.3.10 `WorldState gazebo::physics::WorldState::operator- (const WorldState & _state) const`

Subtraction operator.

Parameters

in	_pt	A (p. 109) state to subtract.
----	-----	--------------------------------------

Returns

The resulting state.

10.189.3.11 `WorldState& gazebo::physics::WorldState::operator= (const WorldState & _state)`

Assignment operator.

Parameters

in	<i>_state</i>	State (p. 744) value
----	---------------	-----------------------------

Returns

Reference to this

10.189.3.12 void gazebo::physics::WorldState::SetWorld (const WorldPtr *_world*)

Set the world.

Parameters

in	<i>_world</i>	Pointer to the world.
----	---------------	-----------------------

10.189.4 Friends And Related Function Documentation

10.189.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::physics::WorldState & *_state*) [friend]

Stream insertion operator.

Parameters

in	<i>_out</i>	output stream
in	<i>_state</i>	World (p. 893) state to output

Returns

the stream

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

- **WorldState.hh**

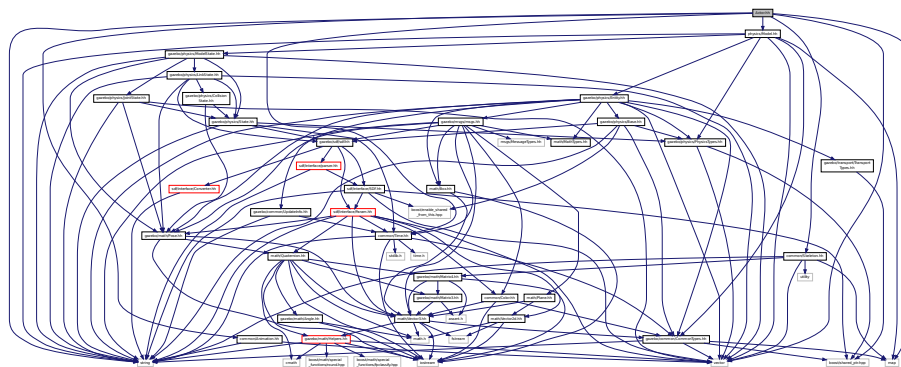
Chapter 11

File Documentation

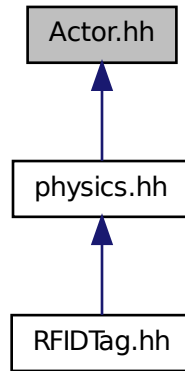
11.1 Actor.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "physics/Model.hh"
#include "common/Time.hh"
#include "common/Skeleton.hh"
#include "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. 109) class enables GPU based mesh model / skeleton scriptable animation.
- struct **gazebo::physics::TrajectoryInfo**

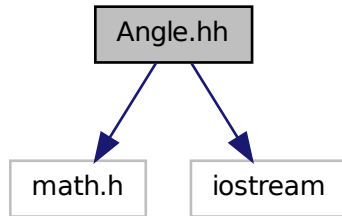
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::physics**
namespace for physics

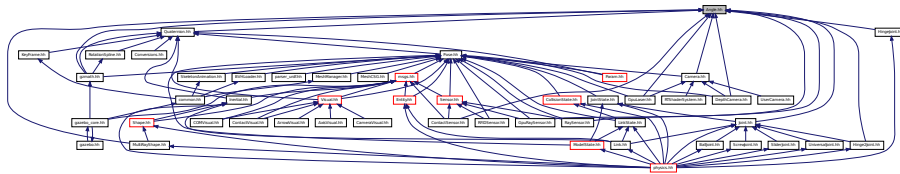
11.2 Angle.hh File Reference

```
#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

<code>in</code>	<code>degrees</code>	
-----------------	----------------------	--

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

<code>in</code>	<code>angle</code>	
-----------------	--------------------	--

Returns

the angle, in range

11.2.1.3 `#define GZ_RTOD(r) ((r) * 180 / M_PI)`

Macro that converts radians to degrees.

Parameters

<code>in</code>	<code>radians</code>	
-----------------	----------------------	--

Returns

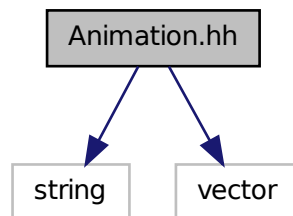
degrees

11.3 Animation.hh File Reference

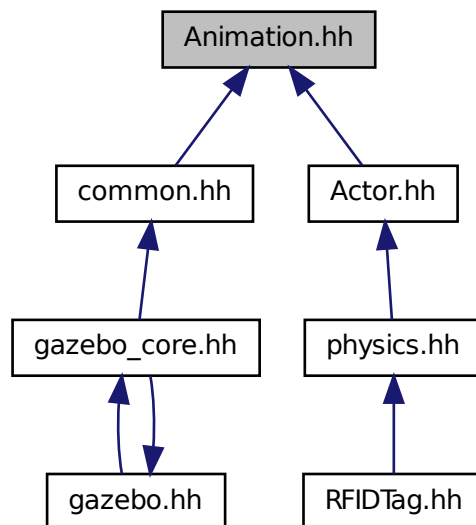
```
#include <string>
```

```
#include <vector>
```

Include dependency graph for Animation.hh:



This graph shows which files directly or indirectly include this file:



Classes

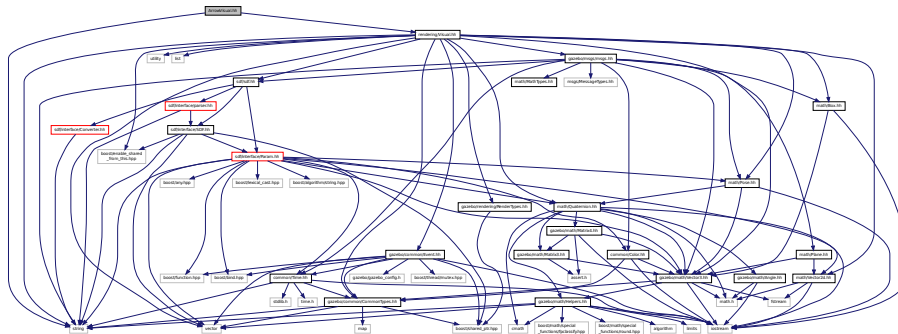
- class **gazebo::common::Animation**
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.
- class **gazebo::common::NumericAnimation**
A (p. 109) numeric animation.
- class **gazebo::common::PoseAnimation**
A (p. 109) pose animation.

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 "rendering/Visual.hh"
Include dependency graph for ArrowVisual.hh:
```



Classes

- class **gazebo::rendering::ArrowVisual**
Basic arrow visualization.

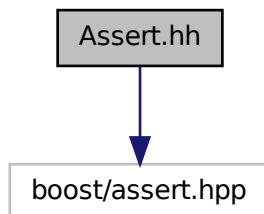
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **ogre**

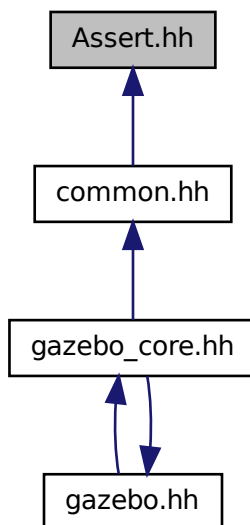
11.5 Assert.hh File Reference

```
#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.

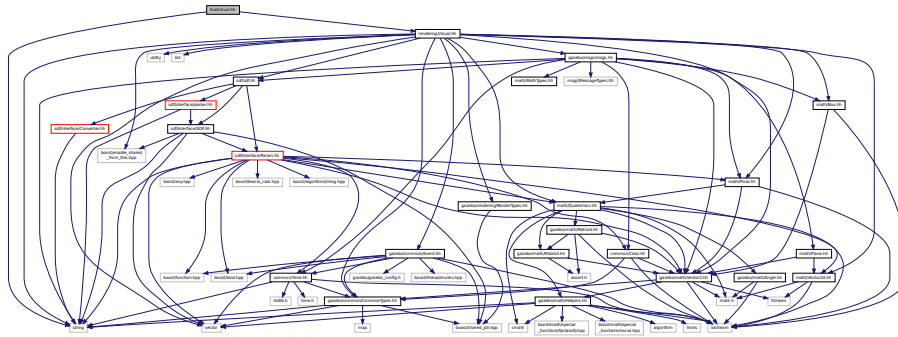
11.5.1 Macro Definition Documentation

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.

11.6 AxisVisual.hh File Reference

```
#include <string>
#include "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.

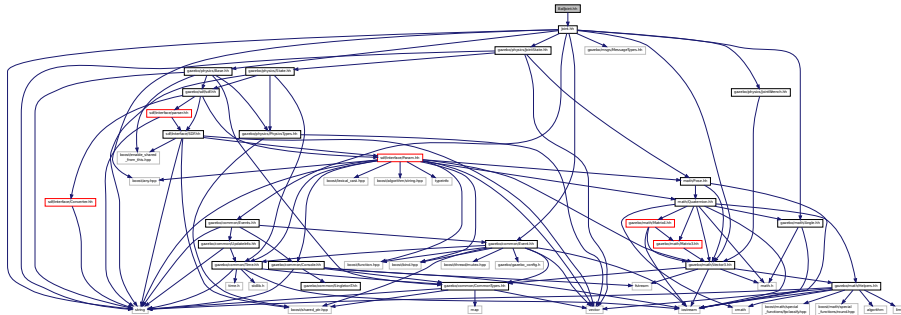
- namespace **gazebo::rendering**

Rendering namespace.

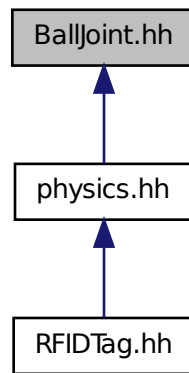
11.7 BallJoint.hh File Reference

```
#include "Joint.hh"
```

Include dependency graph for BallJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

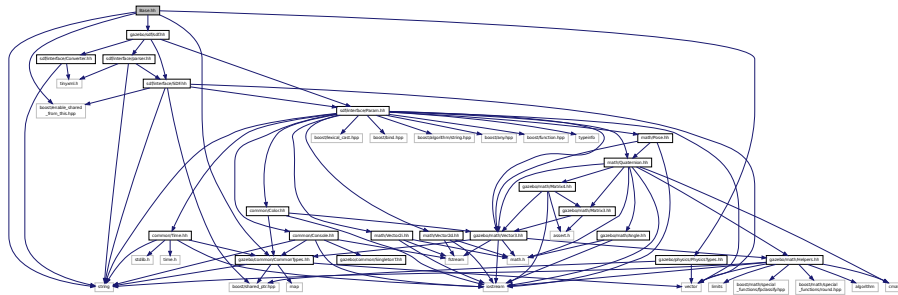
- class **gazebo::physics::BallJoint**< T >
Base (p. 135) class for a ball joint.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.8 Base.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include "gazebo/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. 135) class for most physics classes.

Namespaces

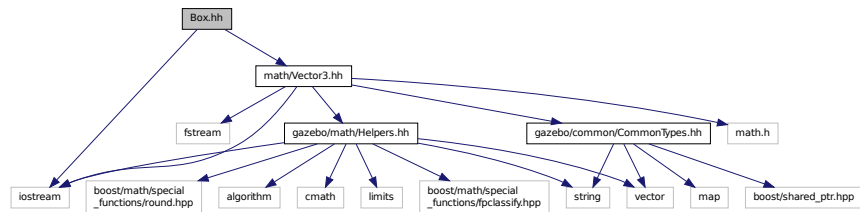
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Variables

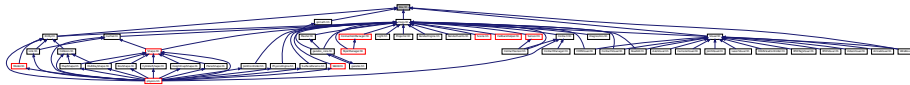
- static std::string **gazebo::physics::EntityTypeName []**
String names for the different entity types.

11.9 Box.hh File Reference

```
#include <iostream>
#include "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.

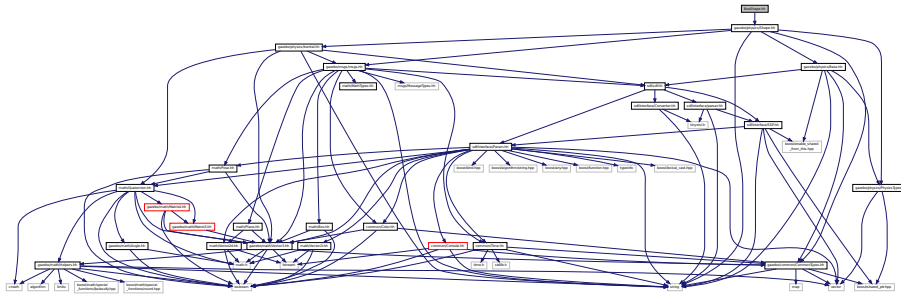
- namespace **gazebo::math**

Math namespace.

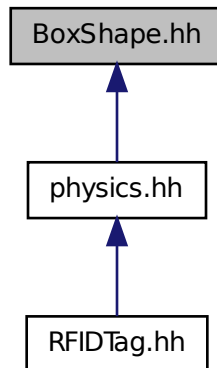
11.10 BoxShape.hh File Reference

```
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for BoxShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

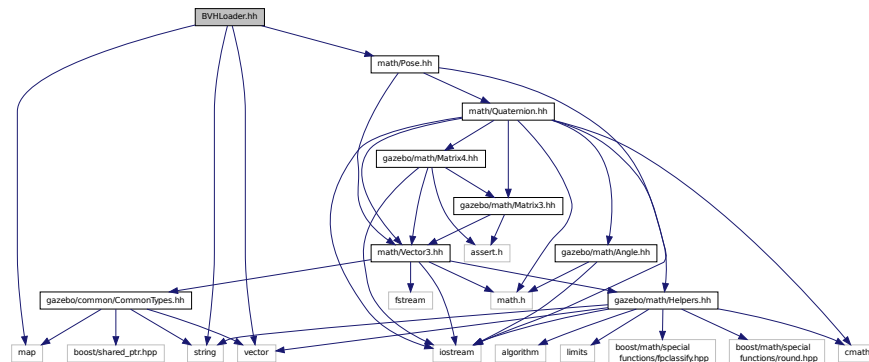
- class **gazebo::physics::BoxShape**
Box geometry primitive.

Namespaces

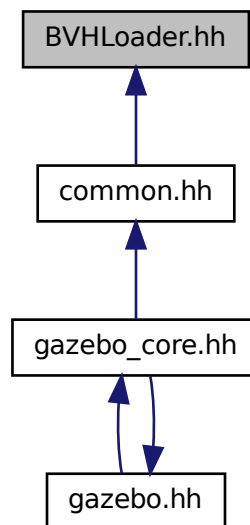
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.11 BVHLoader.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include "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`

11.11.1 Macro Definition Documentation

11.11.1.1 `#define X_POSITION 0`

11.11.1.2 `#define X_ROTATION 3`

11.11.1.3 `#define Y_POSITION 1`

11.11.1.4 `#define Y_ROTATION 4`

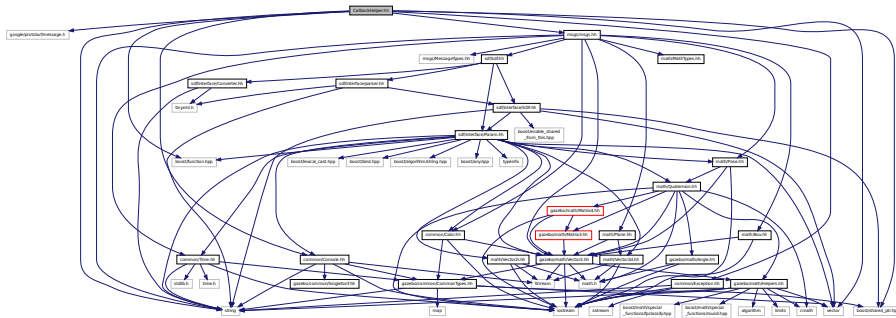
11.11.1.5 `#define Z_POSITION 2`

11.11.1.6 `#define Z_ROTATION 5`

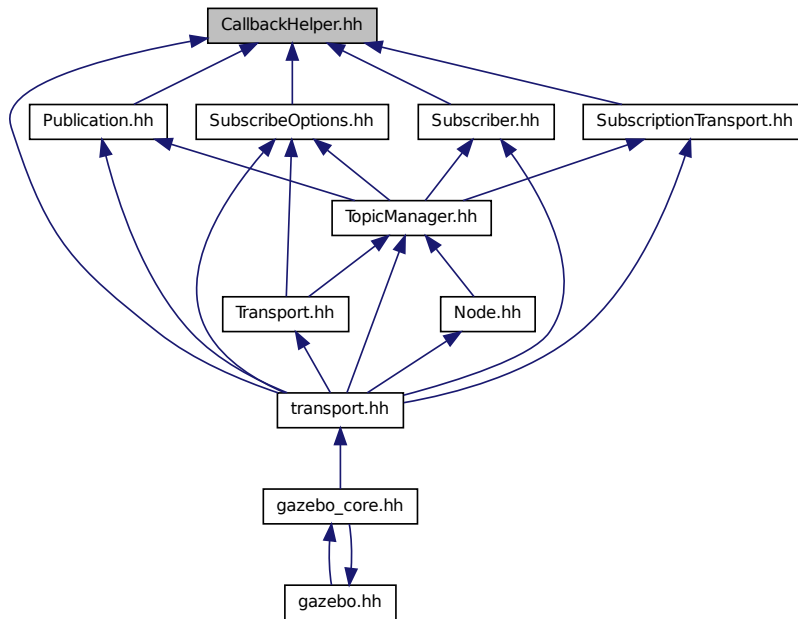
11.12 CallbackHelper.hh File Reference

```
#include <google/protobuf/message.h>
#include <boost/function.hpp>
#include <boost/shared_ptr.hpp>
#include <vector>
#include <string>
#include "common/Console.hh"
#include "msgs/msgs.hh"
#include "common/Exception.hh"
```

Include dependency graph for CallbackHelper.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::CallbackHelper**
A (p. 109) 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

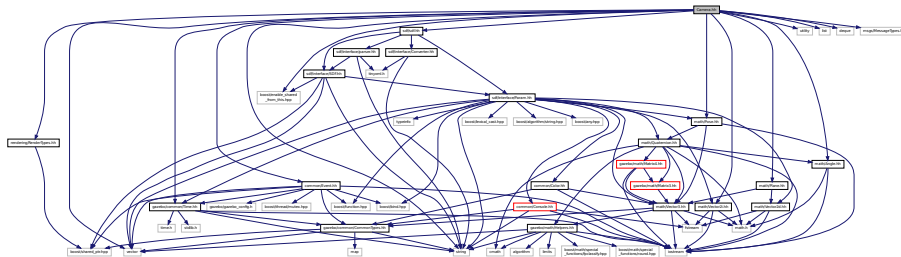
- typedef CallbackHelper * **gazebo::transport::CallbackHelperPtr**

*boost shared pointer to **transport::CallbackHelper** (p. 155)*

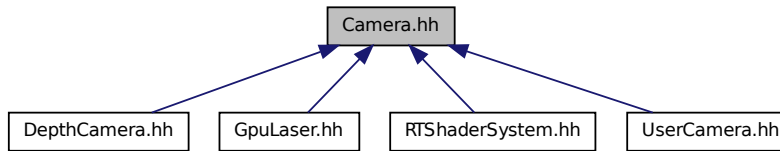
11.13 Camera.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include <utility>
#include <list>
#include <vector>
#include <deque>
#include "common/Event.hh"
#include "common/Time.hh"
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "math/Plane.hh"
#include "math/Vector2i.hh"
#include "msgs/MessageTypes.hh"
#include "rendering/RenderTypes.hh"
#include "sdf/sdf.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**

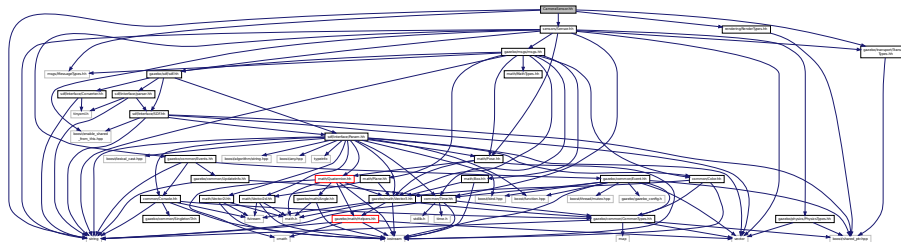
11.14 CameraSensor.hh File Reference

```

#include <string>
#include "sensors/Sensor.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
#include "rendering/RenderTypes.hh"

```

Include dependency graph for CameraSensor.hh:



Classes

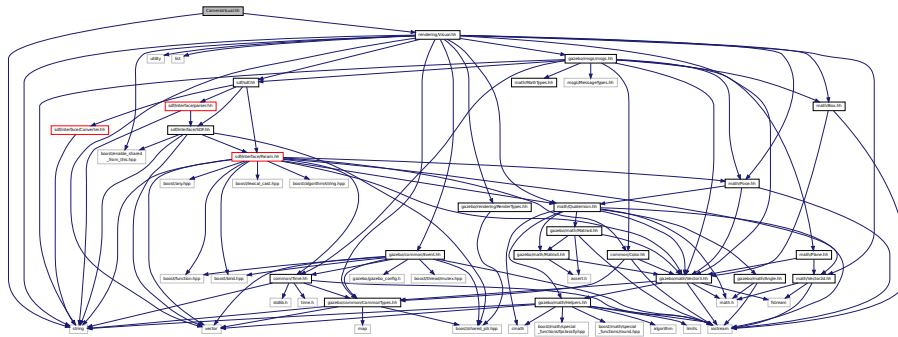
- class **gazebo::sensors::CameraSensor**
Basic camera sensor.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.15 CameraVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
Include dependency graph for CameraVisual.hh:
```



Classes

- class **gazebo::rendering::CameraVisual**
Basic camera visualization.

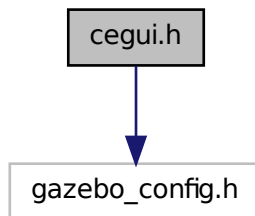
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

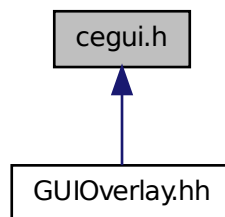
11.16 cegui.h File Reference

```
#include "gazebo_config.h"
```

Include dependency graph for cegui.h:



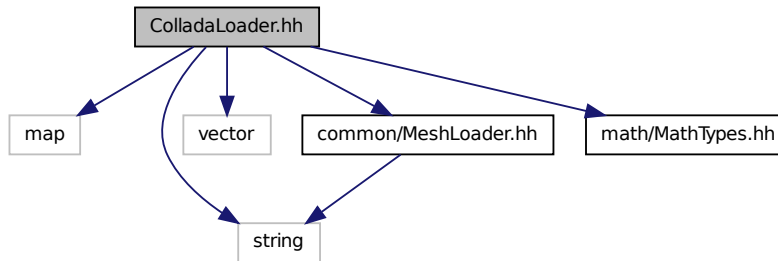
This graph shows which files directly or indirectly include this file:



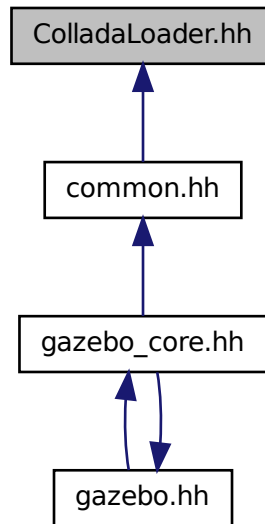
11.17 ColladaLoader.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include "common/MeshLoader.hh"
#include "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.

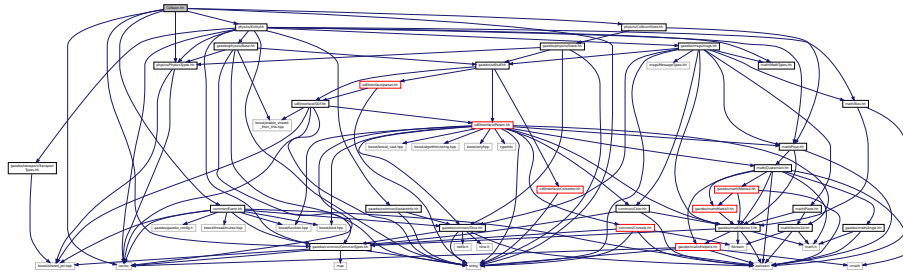
- namespace **gazebo::common**

Common namespace.

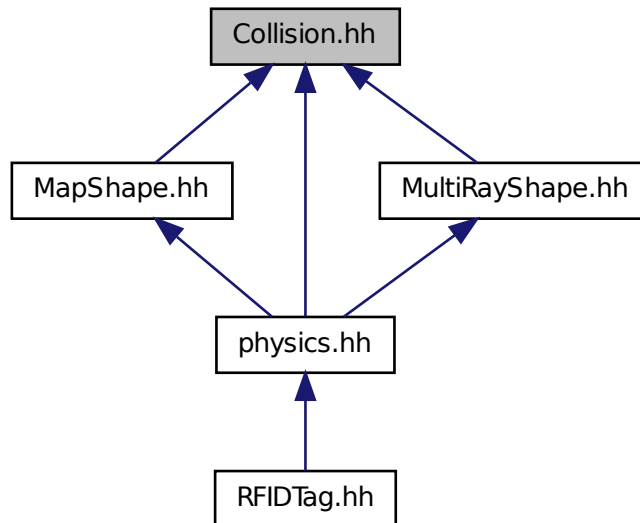
11.18 Collision.hh File Reference

```
#include <string>
#include <vector>
#include "common/Event.hh"
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/CollisionState.hh"
#include "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. 135) class for all collision entities.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

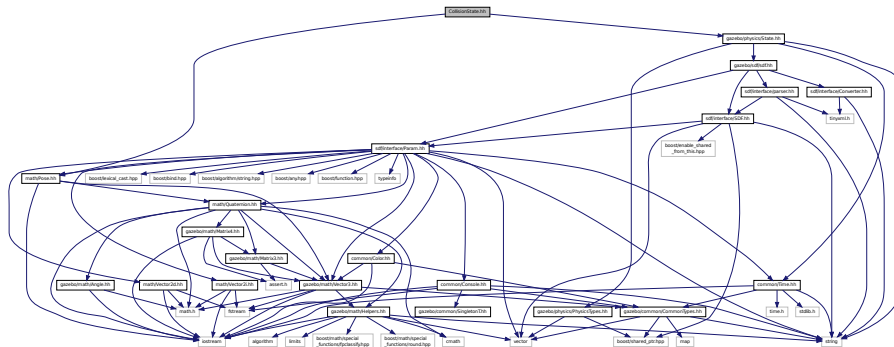
namespace for physics

11.19 CollisionState.hh File Reference

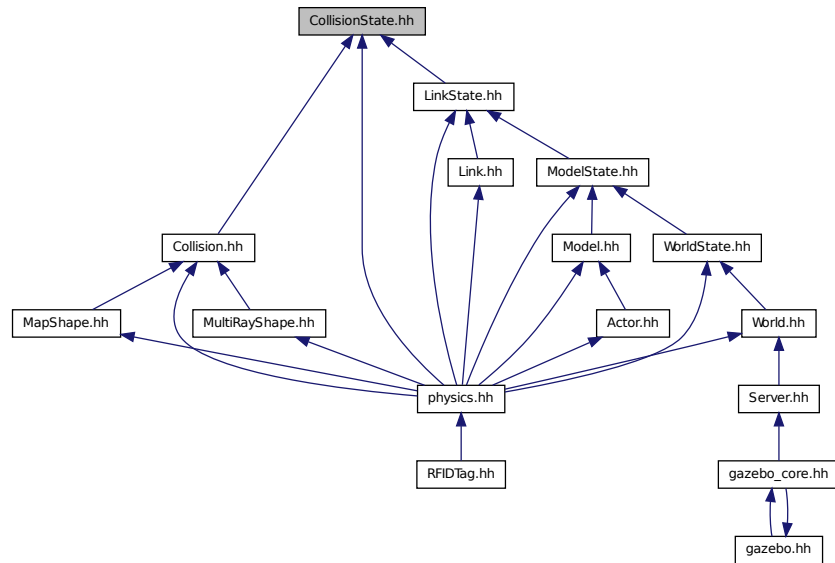
```
#include "gazebo/physics/State.hh"
```

```
#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. 192) object.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

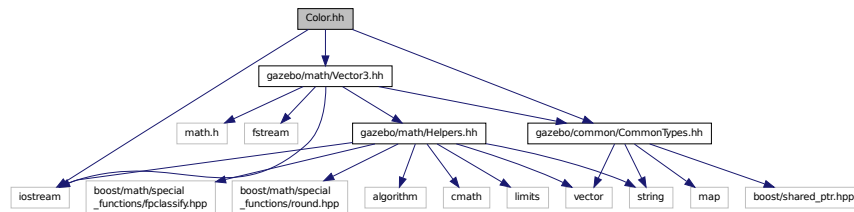
11.20 Color.hh File Reference

```

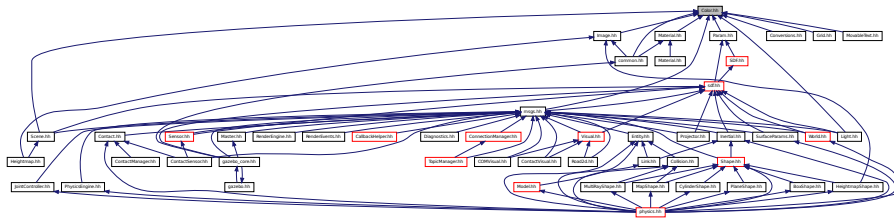
#include <iostream>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/math/Vector3.hh"

```

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.

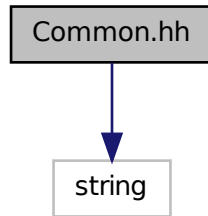
- namespace **gazebo::common**

Common namespace.

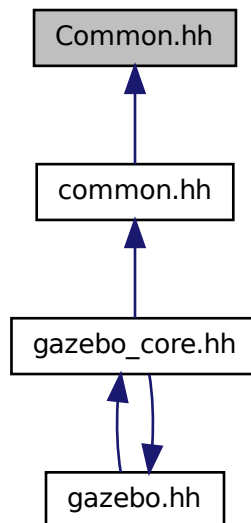
11.21 Common.hh File Reference

```
#include <string>
```


Include dependency graph for Common.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

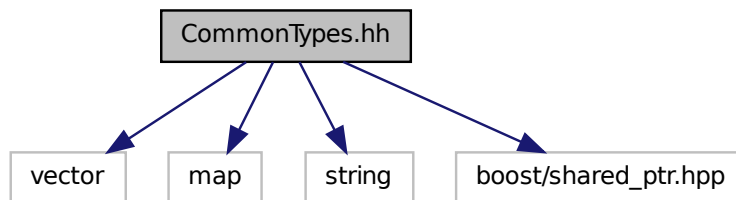
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

Functions

- void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)
*add path prefix to **common::SystemPaths** (p. 769)*
- std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath=true)
*search for file in **common::SystemPaths** (p. 769)*
- std::string **gazebo::common::find_file_path** (const std::string &_file)
*search for a file in **common::SystemPaths** (p. 769)*

11.22 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 **ParamT**< T >

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

- namespace **gazebo::event**
Event (p. 288) namespace.

Macros

- #define **GAZEBO_DEPRECATED**
- #define **GAZEBO_FORCEINLINE**
- #define **NULL** 0

Typedefs

- typedef Animation * **gazebo::common::AnimationPtr**
- typedef std::vector
< ConnectionPtr > **gazebo::event::Connection_V**
- typedef Connection * **gazebo::event::ConnectionPtr**
- typedef DiagnosticTimer * **gazebo::common::DiagnosticTimerPtr**
- typedef GUIPlugin * **gazebo::GUIPluginPtr**
- typedef ModelPlugin * **gazebo::ModelPluginPtr**
- typedef NumericAnimation * **gazebo::common::NumericAnimationPtr**
- typedef std::vector
< common::Param * > **gazebo::common::Param_V**
- typedef PoseAnimation * **gazebo::common::PoseAnimationPtr**
- typedef SensorPlugin * **gazebo::SensorPluginPtr**
- typedef std::map< std::string,
std::string > **gazebo::common::StrStr_M**
- typedef SystemPlugin * **gazebo::SystemPluginPtr**
- typedef VisualPlugin * **gazebo::VisualPluginPtr**
- typedef WorldPlugin * **gazebo::WorldPluginPtr**

11.22.1 Macro Definition Documentation

11.22.1.1 #define **GAZEBO_DEPRECATED**

11.22.1.2 #define **GAZEBO_FORCEINLINE**

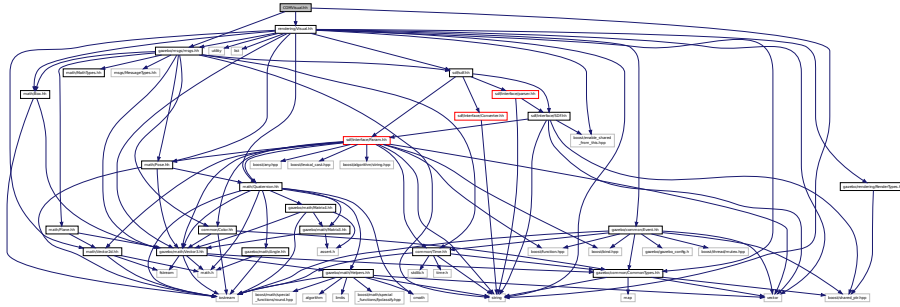
11.22.1.3 #define **NULL** 0

Referenced by gazebo::transport::TopicManager::Advertise(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::event::EventT< T >::Disconnect(), gazebo::transport::ConnectionReadTask::execute(), gazebo::transport::CallbackHelperT< M >::GetMsgType(), and gazebo::transport::SubscribeOptions::Init().

11.23 COMVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
#include "msgs/msgs.hh"
```

Include dependency graph for COMVisual.hh:



Classes

- class **gazebo::rendering::COMVisual**
Basic Center of Mass visualization.

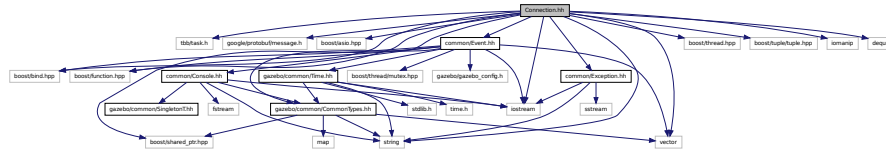
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **ogre**

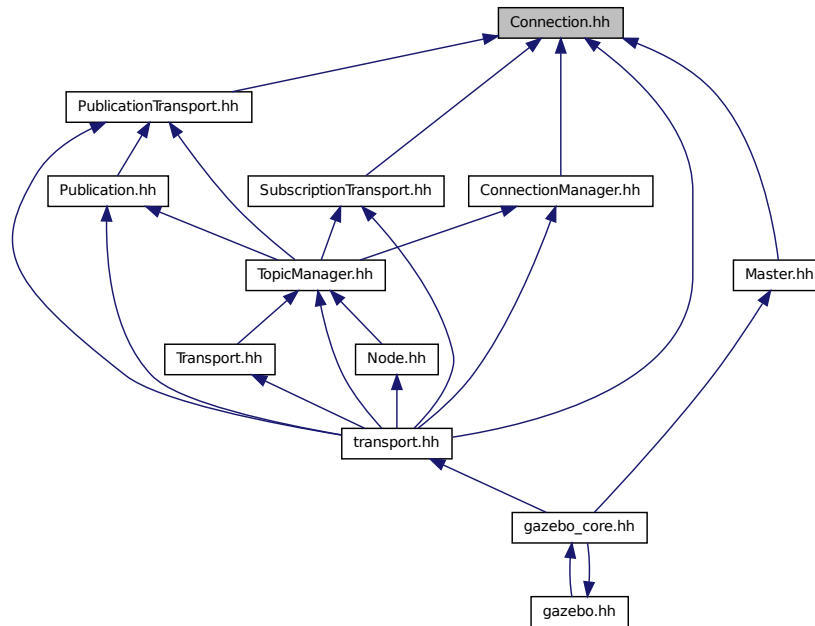
11.24 Connection.hh File Reference

```
#include <tbb/task.h>
#include <google/protobuf/message.h>
#include <boost/asio.hpp>
#include <boost/bind.hpp>
#include <boost/function.hpp>
#include <boost/thread.hpp>
#include <boost/tuple/tuple.hpp>
#include <string>
#include <vector>
#include <iostream>
#include <iomanip>
#include <deque>
#include "common/Event.hh"
#include "common/Console.hh"
#include "common/Exception.hh"
```

Include dependency graph for Connection.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Connection**
Single TCP/IP connection manager.
- class **gazebo::transport::ConnectionReadTask**

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

Macros

- #define **HEADER_LENGTH** 8

Typedefs

- typedef Connection * **gazebo::transport::ConnectionPtr**

Functions

- bool **gazebo::transport::is_stopped** ()

Is the transport system stopped?

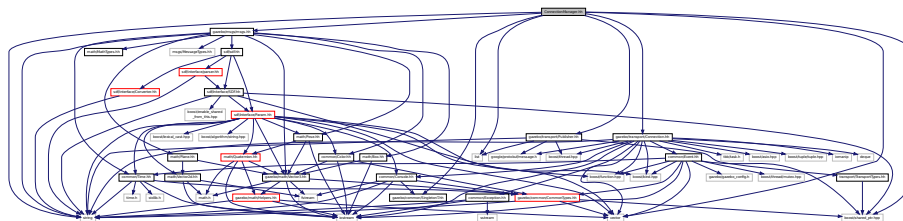
11.24.1 Macro Definition Documentation

11.24.1.1 #define HEADER_LENGTH 8

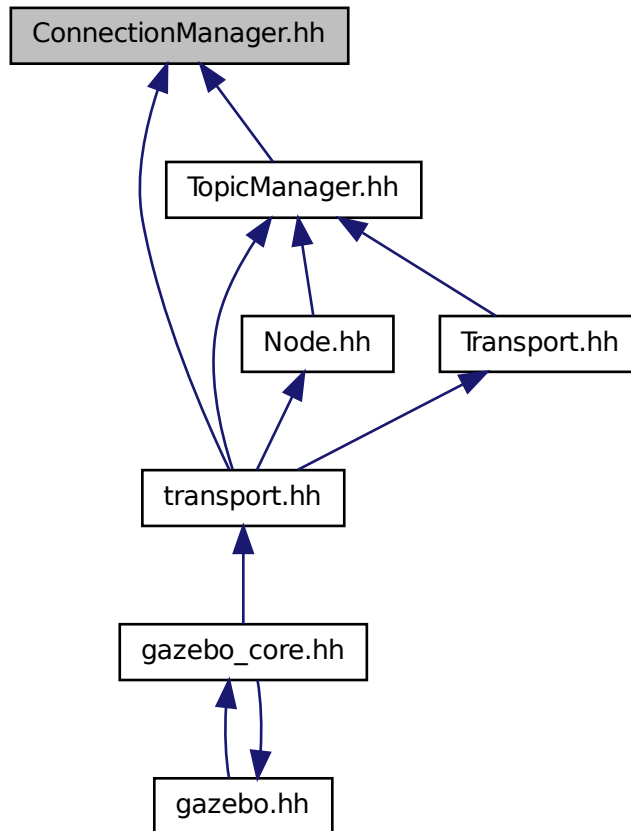
Referenced by gazebo::transport::Connection::AsyncRead().

11.25 ConnectionManager.hh File Reference

```
#include <boost/shared_ptr.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.

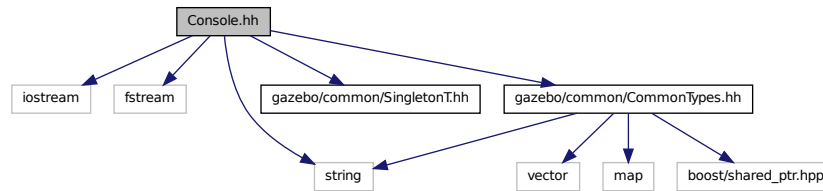
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

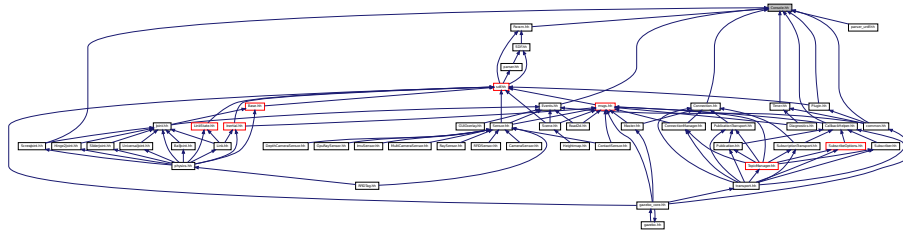
11.26 Console.hh File Reference

```
#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

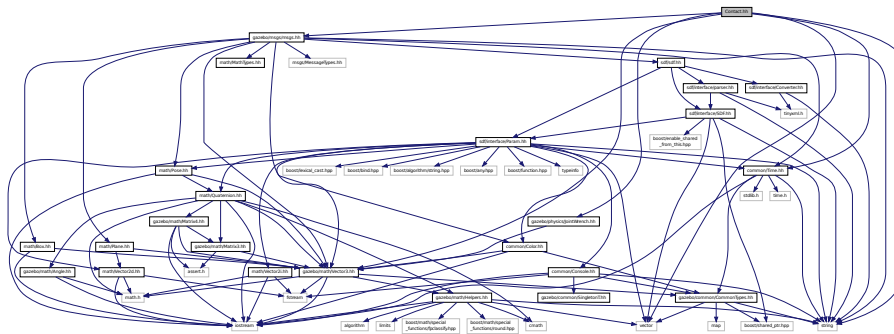
- #define **gzclr_end** "\033[0m"
End marker.
- #define **gzclr_start**(clr) "\033[1;33m"
Start marker.
- #define **gzdbg** (gazebo::common::Console::Instance()->ColorMsg("Dbg", 36))
Output a debug message.

- **#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.

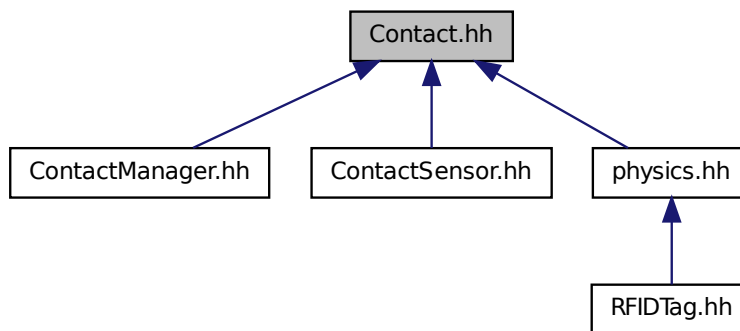
11.27 Contact.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/msgs/msgs.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 (p. 109) 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

11.27.1 Macro Definition Documentation

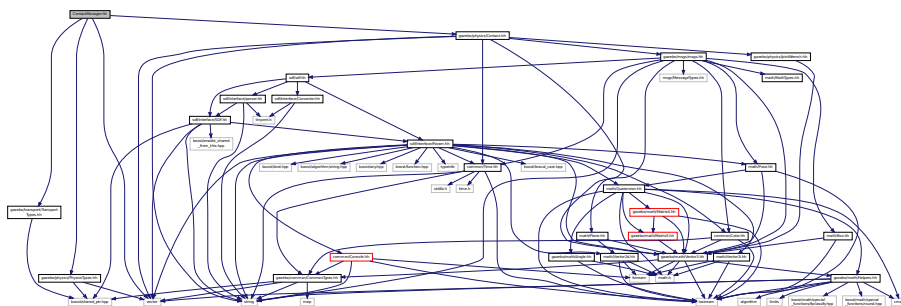
11.27.1.1 #define MAX_COLLIDE_RETURNS 250

11.27.1.2 #define MAX_CONTACT_JOINTS 32

11.28 ContactManager.hh File Reference

```
#include <vector>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Contact.hh"
```

Include dependency graph for ContactManager.hh:



Classes

- class **gazebo::physics::ContactManager**
Aggregates all the contact information generated by the collision detection engine.

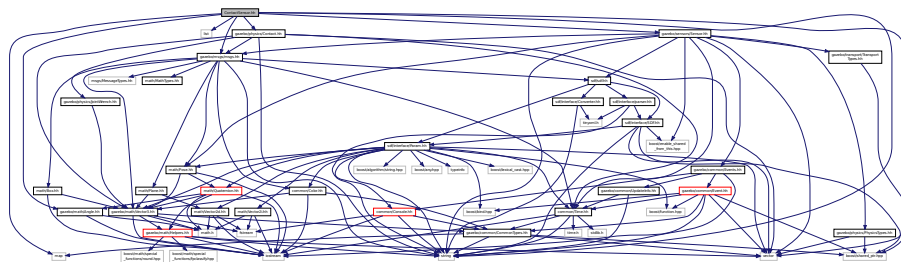
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.29 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.

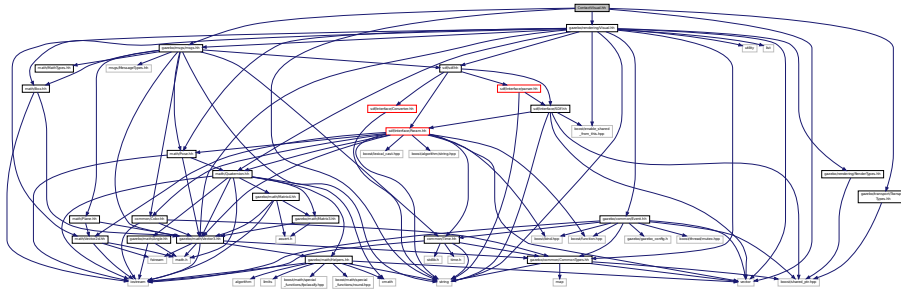
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.30 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**

11.31 Conversions.hh File Reference

```
#include "rendering/ogre_gazebo.h"
#include "common/Color.hh"
#include "math/Vector3.hh"
#include "math/Quaternion.hh"
```

Include dependency graph for Conversions.hh:



Classes

- class **gazebo::rendering::Conversions**
Conversions (p. 243) *Conversions.hh* (p. 948) *rendering/Conversions.hh* (p. 948).

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.

- namespace **gazebo::rendering**

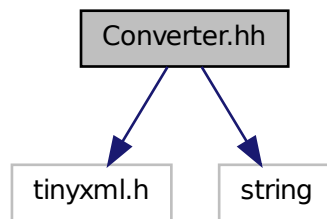
Rendering namespace.

11.32 Converter.hh File Reference

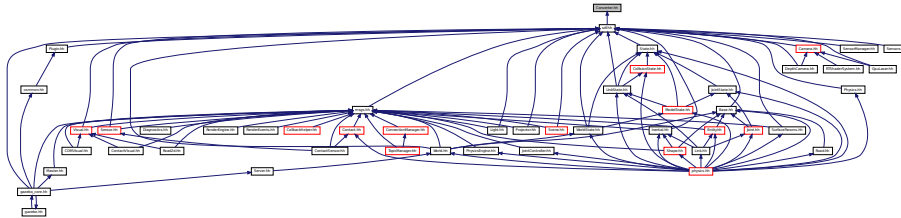
```
#include <tinyxml.h>
```

```
#include <string>
```

Include dependency graph for Converter.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **sdf::Converter**

*Convert from one version of **SDF** (p. 680) to another.*

Namespaces

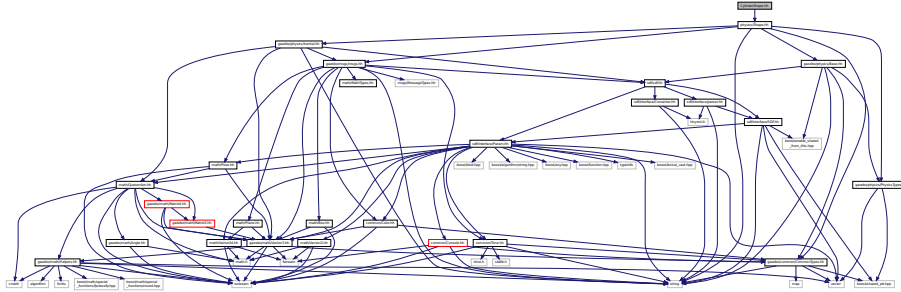
- namespace **sdf**

namespace for Simulation Description Format parser

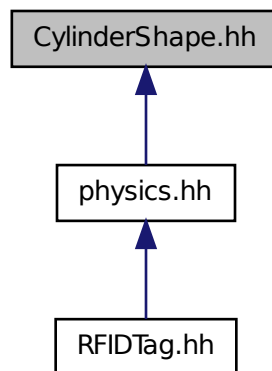
11.33 CylinderShape.hh File Reference

```
#include "physics/Shape.hh"
```

Include dependency graph for CylinderShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

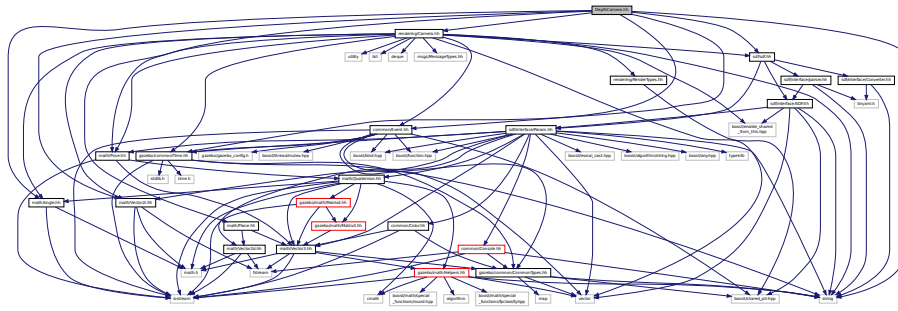
- class **gazebo::physics::CylinderShape**
Cylinder collision.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.34 DepthCamera.hh File Reference

```
#include <string>
#include "common/Event.hh"
#include "common/Time.hh"
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "math/Vector2i.hh"
#include "sdf/sdf.hh"
#include "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.

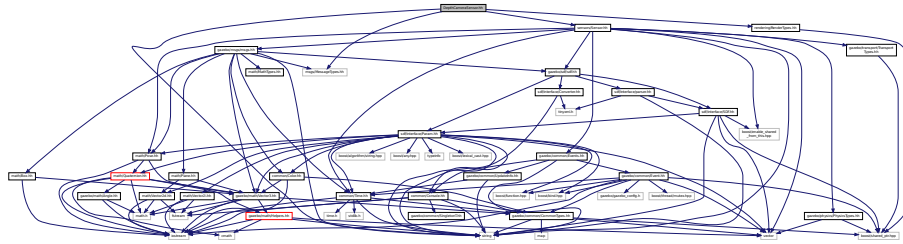
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.35 DepthCameraSensor.hh File Reference

```
#include <string>
#include "sensors/Sensor.hh"
#include "msgs/MessageTypes.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for DepthCameraSensor.hh:



Classes

- class **gazebo::sensors::DepthCameraSensor**

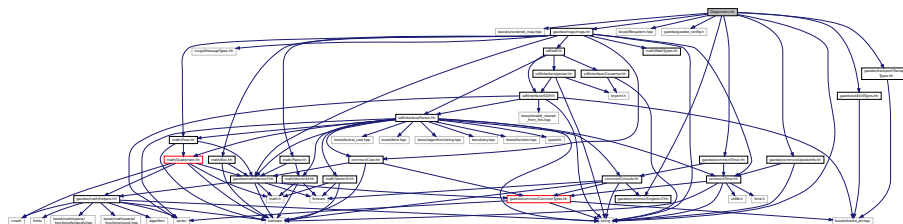
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.36 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**

- class **gazebo::util::DiagnosticTimer**
A (p. 109) diagnostic manager class.
- class **gazebo::util::DiagnosticTimer**
A (p. 109) timer designed for diagnostics.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::util**

Macros

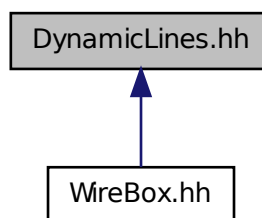
- #define **DIAG_TIMER_LAP**(_name, _prefix) ((void)0)
- #define **DIAG_TIMER_START**(_name) ((void) 0)
- #define **DIAG_TIMER_STOP**(_name) ((void) 0)

11.37 DynamicLines.hh File Reference

```
#include <vector>
#include <string>
#include "math/Vector3.hh"
#include "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

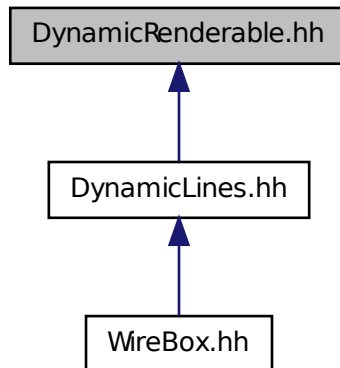
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.38 DynamicRenderable.hh File Reference

```
#include "rendering/ogre_gazebo.h"
#include "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.

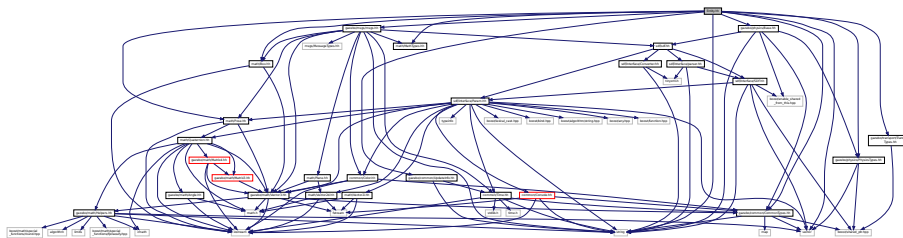
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

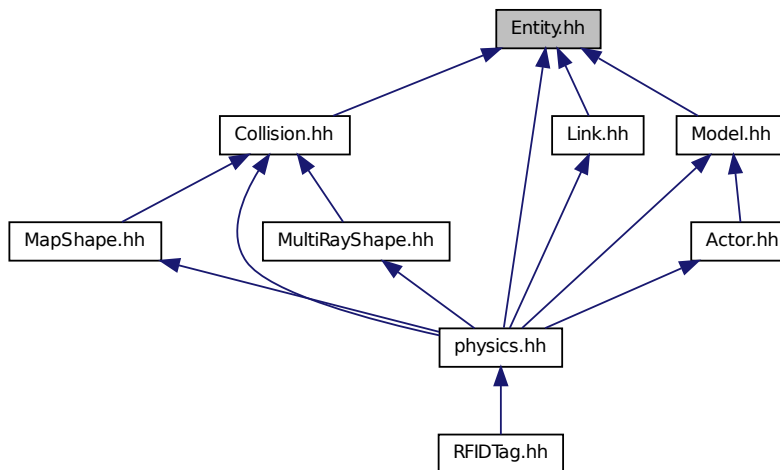
11.39 Entity.hh File Reference

```
#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. 135) class for all physics objects in Gazebo.

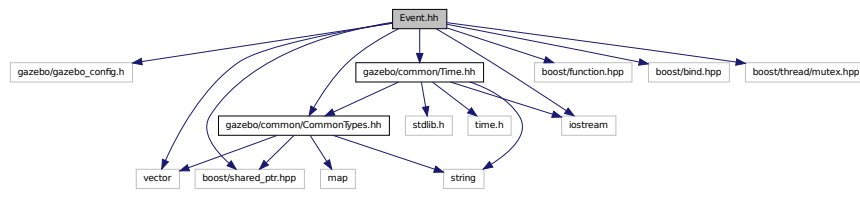
Namespaces

- namespace **boost**
- namespace **gazebo**
 - Forward declarations for the common classes.*
- namespace **gazebo::physics**
 - namespace for physics*

11.40 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 (p. 109) class that encapsulates a connection.*
- class **gazebo::event::Event**
 - Base class for all events.*
- class **gazebo::event::EventT < T >**
 - A (p. 109) class for event processing.*

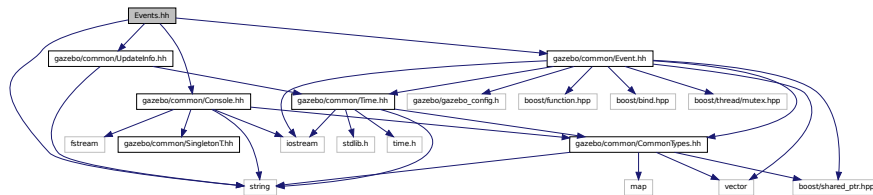
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::event**
Event (p. 288) namespace.

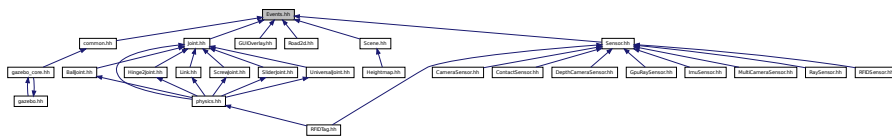
11.41 Events.hh File Reference

```
#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

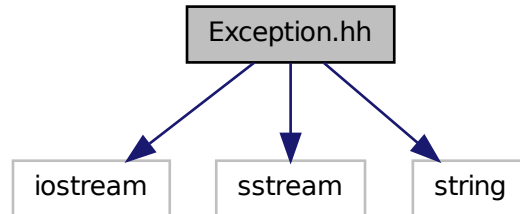
- class **gazebo::event::Events**
*An **Event** (p. 288) class to get notifications for simulator events.*

Namespaces

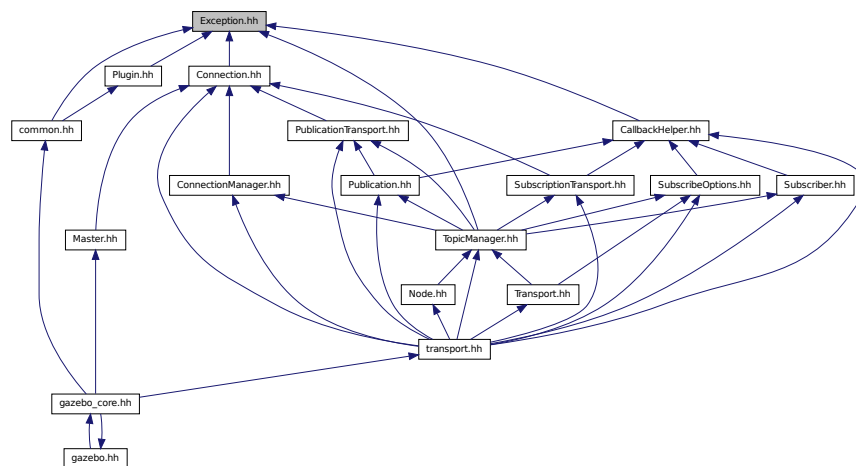
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::event**
Event (p. 288) namespace.

11.42 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

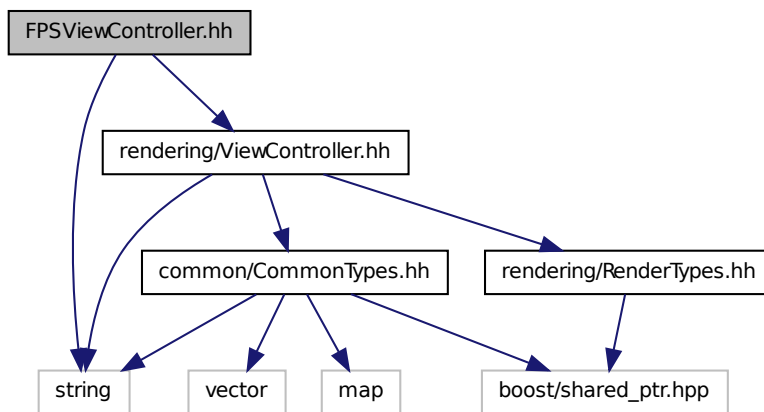
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

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.

11.43 FPSViewController.hh File Reference

```
#include <string>
#include "rendering/ViewController.hh"
Include dependency graph for FPSViewController.hh:
```



Classes

- class **gazebo::rendering::FPSViewController**
First Person Shooter style view controller.

Namespaces

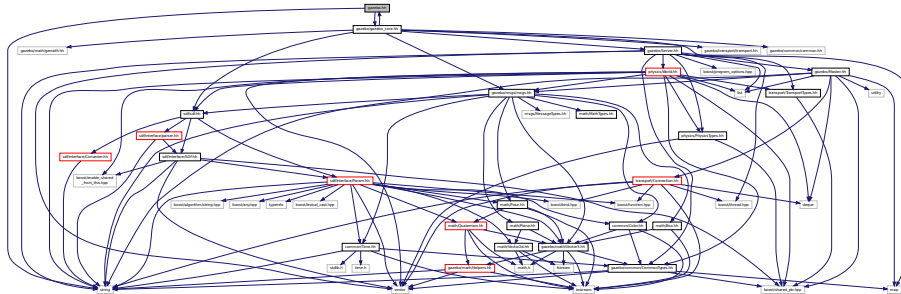
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.44 gazebo.hh File Reference

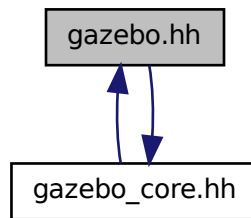
```
#include <gazebo/gazebo_core.hh>
```

```
#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.

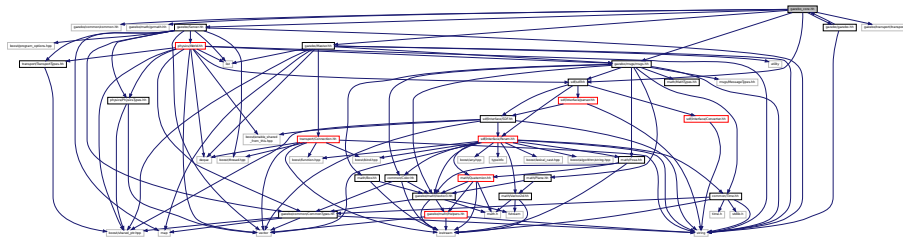
Functions

- 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** ()

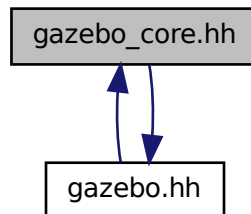
11.45 gazebo_core.hh File Reference

```
#include <gazebo/common/common.hh>
#include <gazebo/math/gzmath.hh>
#include <gazebo/msgs/msgs.hh>
#include <gazebo/sdf/sdf.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:



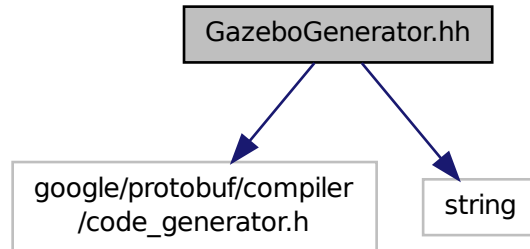
This graph shows which files directly or indirectly include this file:



11.46 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. 89).*

Namespaces

- namespace **google**
- namespace **google::protobuf**
- namespace **google::protobuf::compiler**
- namespace **google::protobuf::compiler::cpp**

11.47 GpuLaser.hh File Reference

```

#include <string>
#include <vector>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/rendering/Camera.hh"
#include "gazebo/sensors/SensorTypes.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/sdf/sdf.hh"

```

Include dependency graph for GpuLaser.hh:



Classes

- class **gazebo::rendering::GpuLaser**
GPU based laser distance sensor.

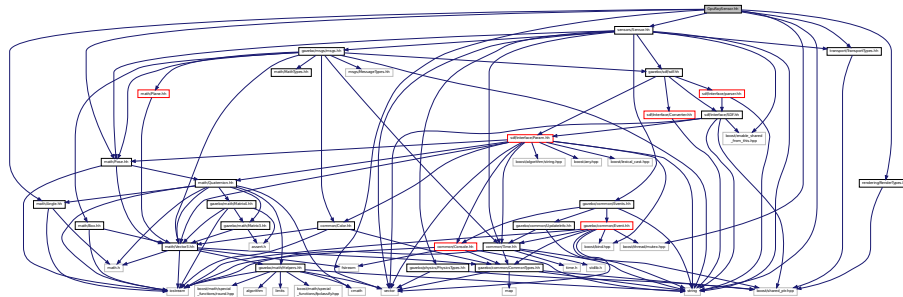
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.48 GpuRaySensor.hh File Reference

```
#include <vector>
#include <string>
#include <boost/thread/mutex.hpp>
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "transport/TransportTypes.hh"
#include "sensors/Sensor.hh"
#include "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.

11.49 Grid.hh File Reference

```
#include <stdint.h>
#include <vector>
#include <string>
#include "rendering/ogre_gazebo.h"
#include "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**

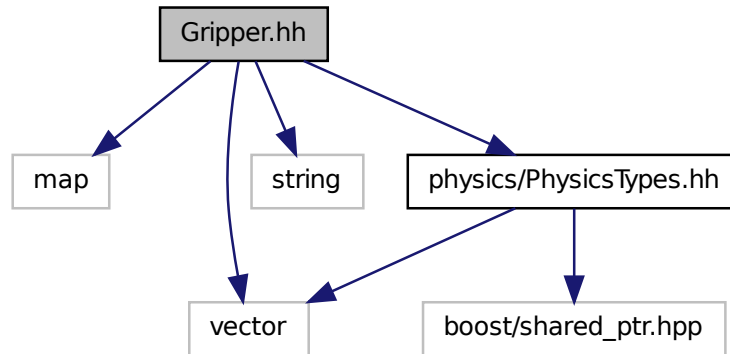
Rendering namespace.

- namespace **Ogre**

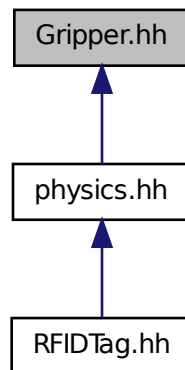
11.50 Gripper.hh File Reference

```
#include <map>
#include <vector>
#include <string>
#include "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 (p. 109) gripper abstraction.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

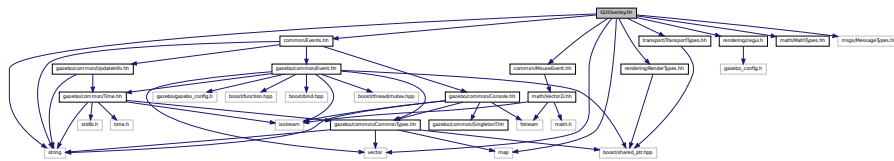
- namespace **gazebo::physics**

namespace for physics

11.51 GUIOverlay.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "rendering/cegui.h"
#include "common/MouseEvent.hh"
#include "common/Events.hh"
#include "math/MathTypes.hh"
#include "rendering/RenderTypes.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
```

Include dependency graph for GUIOverlay.hh:



Classes

- class **gazebo::rendering::GUIOverlay**

A (p. 109) 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**

11.52 Heightmap.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/ogre_gazebo.h"
#include "common/Image.hh"
#include "math/Vector3.hh"
#include "math/Vector2d.hh"
#include "rendering/Scene.hh"
```

Include dependency graph for Heightmap.hh:



Classes

- 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.

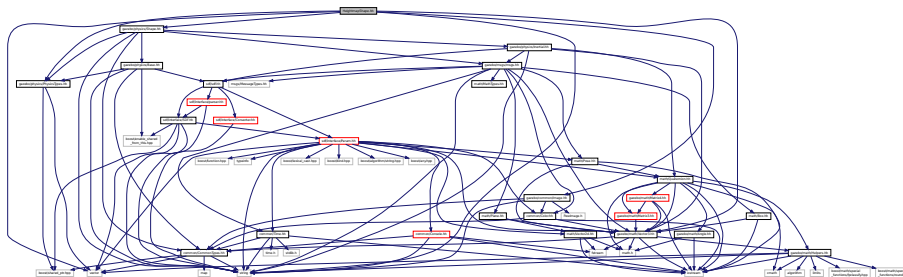
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

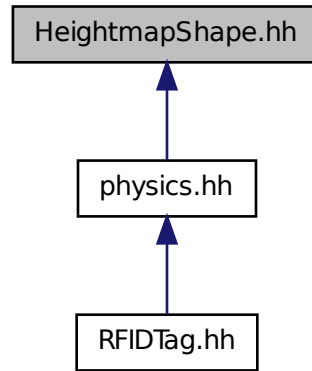
11.53 HeightmapShape.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/common/Image.hh"
#include "gazebo/math/Vector3.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. 348) collision shape builds a heightmap from an image.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

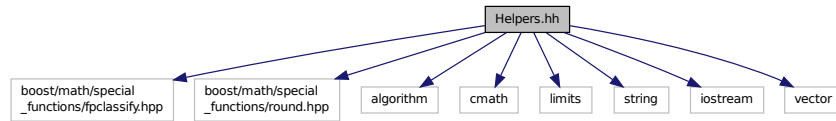
- namespace **gazebo::physics**

namespace for physics

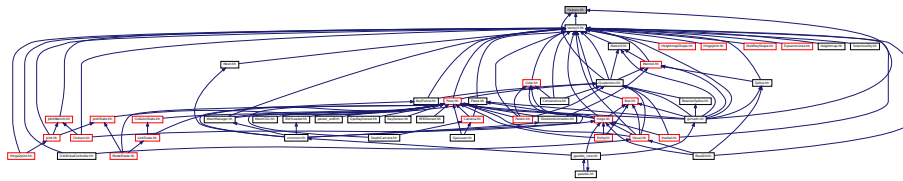
11.54 Helpers.hh File Reference

```
#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()`
- `#define GZ_DBL_MIN std::numeric_limits<double>::min()`
- `#define GZ_FLT_MAX std::numeric_limits<float>::max()`
- `#define GZ_FLT_MIN std::numeric_limits<float>::min()`

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)

11.54.1 Macro Definition Documentation

11.54.1.1 `#define GZ_DBL_MAX std::numeric_limits<double>::max()`

11.54.1.2 `#define GZ_DBL_MIN std::numeric_limits<double>::min()`

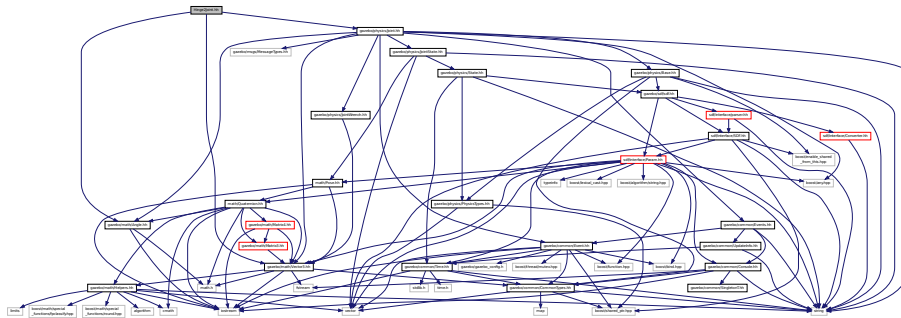
11.54.1.3 `#define GZ_FLT_MAX std::numeric_limits<float>::max()`

11.54.1.4 `#define GZ_FLT_MIN std::numeric_limits<float>::min()`

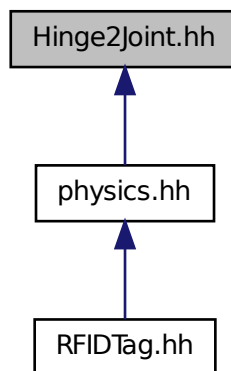
11.55 Hinge2Joint.hh File Reference

```
#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

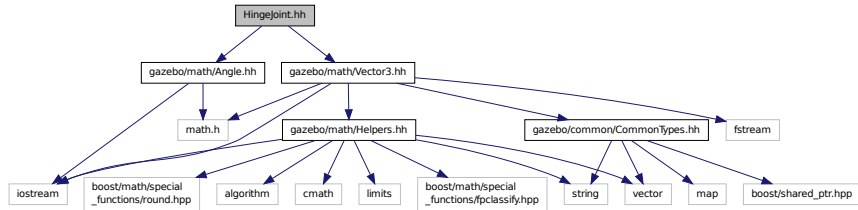
- class `gazebo::physics::Hinge2Joint< T >`
A (p. 109) two axis hinge joint.

Namespaces

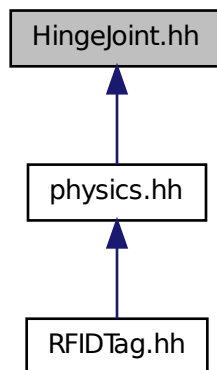
- namespace `gazebo`
Forward declarations for the common classes.
- namespace `gazebo::physics`
namespace for physics

11.56 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 (p. 109) single axis hinge joint.

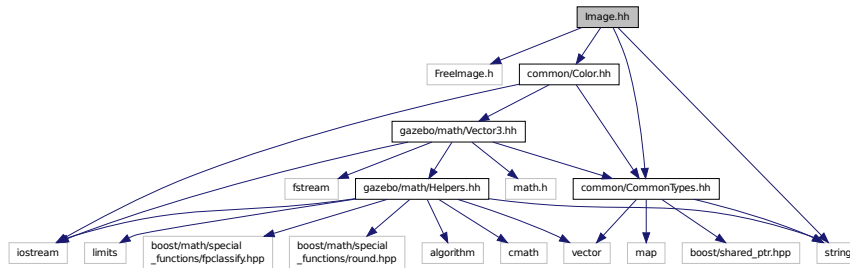
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

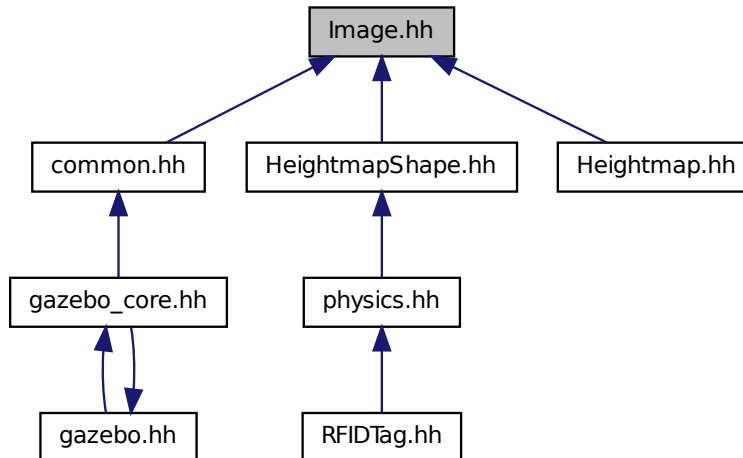
11.57 Image.hh File Reference

```
#include <FreeImage.h>
#include <string>
#include "common/CommonTypes.hh"
#include "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.

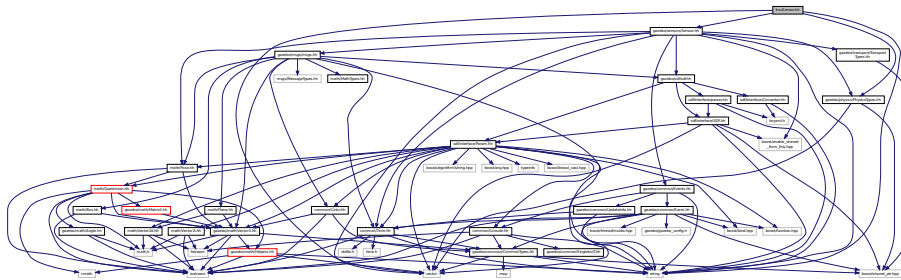
Variables

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

11.58 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.

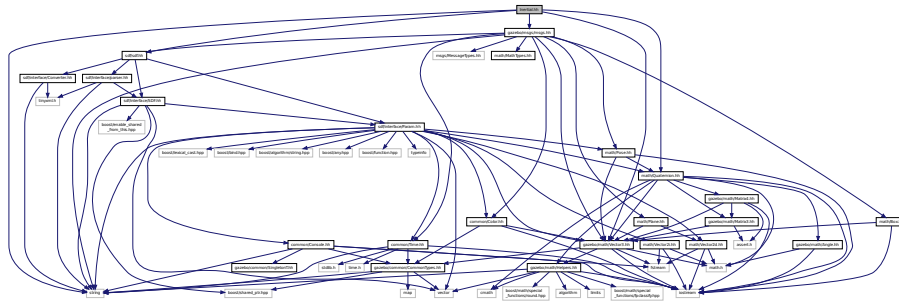
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

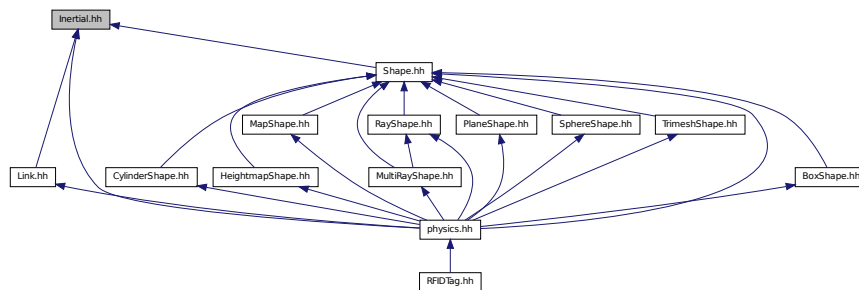
11.59 Inertial.hh File Reference

```
#include <string>
#include "gazebo_msgs/msgs.hh"
#include "gazebo/sdf/sdf.hh"
#include "gazebo/math/Quaternion.hh"
#include "gazebo/math/Vector3.hh"
```

Include dependency graph for Inertial.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Inertial**
A (p. 109) class for inertial information about a link.

Namespaces

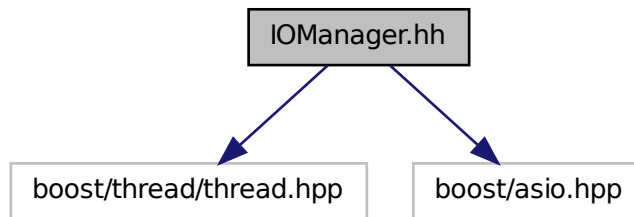
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.60 IOManager.hh File Reference

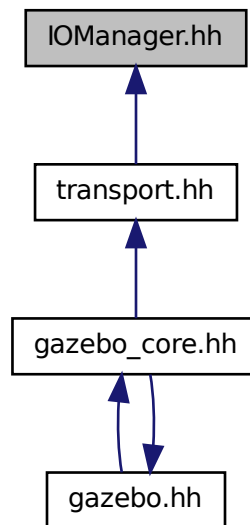
```
#include <boost/thread/thread.hpp>
```

```
#include <boost/asio.hpp>
```

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.

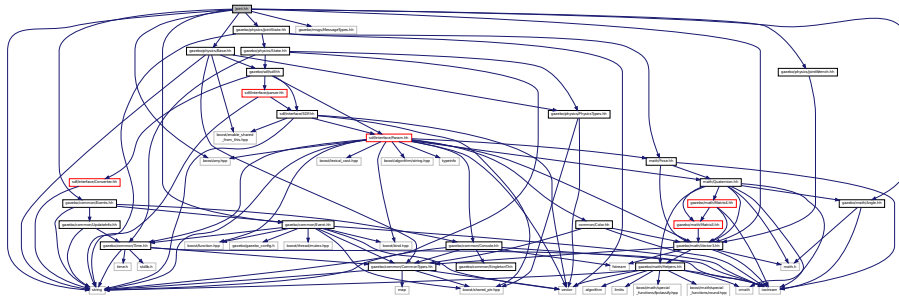
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

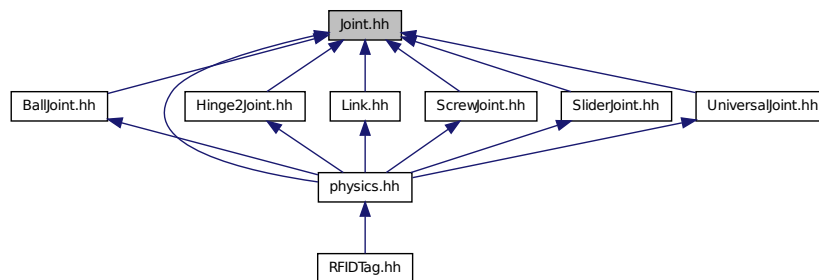
11.61 Joint.hh File Reference

```
#include <string>
#include <boost/any.hpp>
#include "gazebo/common/Event.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo msgs/MessageTypes.hh"
#include "gazebo/physics/JointState.hh"
#include "gazebo/physics/Base.hh"
#include "gazebo/physics/JointWrench.hh"
```

Include dependency graph for Joint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Joint**

Base (p. 135) 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.

11.61.1 Macro Definition Documentation

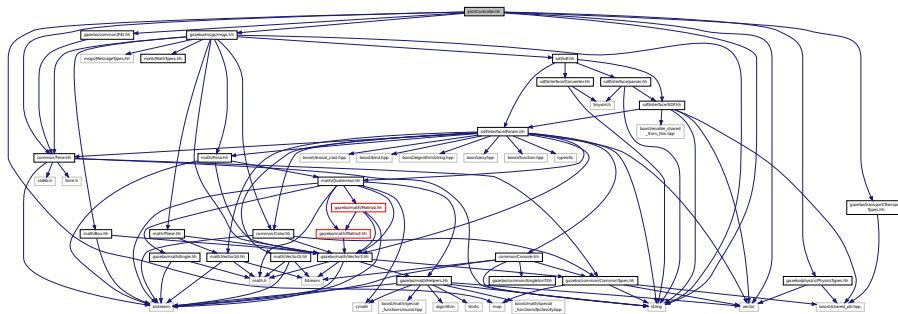
11.61.1.1 `#define MAX_JOINT_AXIS 2`

maximum number of axis per joint anticipated.

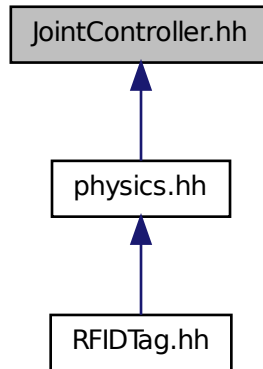
Currently, this is 2 as 3-axis joints (e.g. ball) actuation, control is not there yet.

11.62 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/messages/messages.hh"
Include dependency graph for JointController.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::JointController**

A (p. 109) class for manipulating **physics::Joint** (p. 375).

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

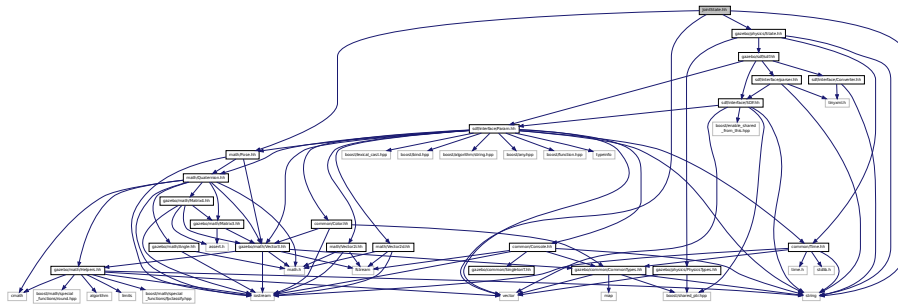
- namespace **gazebo::physics**

namespace for physics

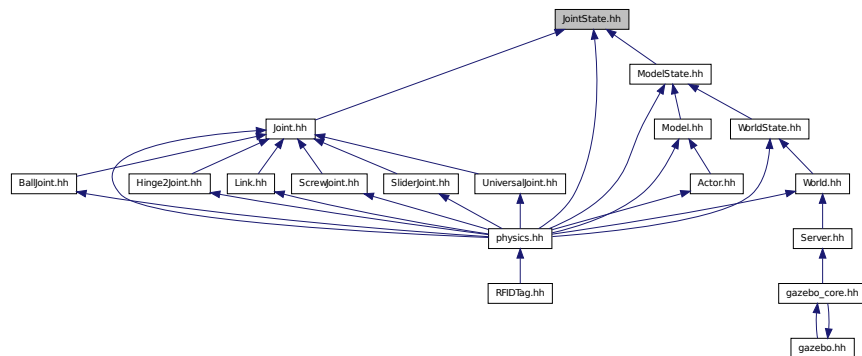
11.63 JointState.hh File Reference

```
#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. 375)

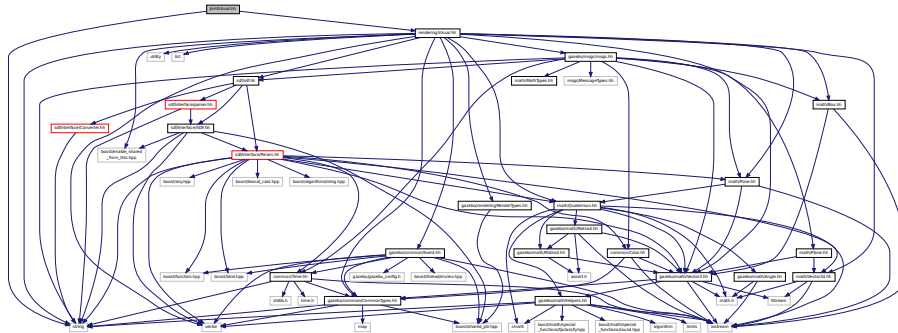
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.64 JointVisual.hh File Reference

```
#include <string>
#include "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.

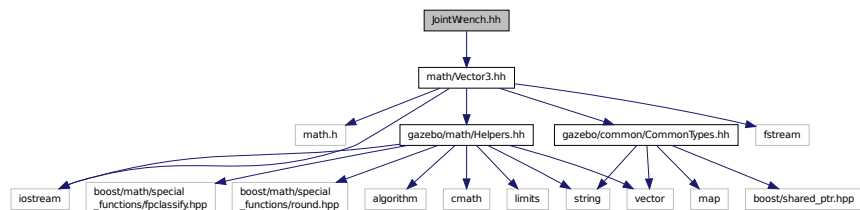
- namespace **gazebo::rendering**

Rendering namespace.

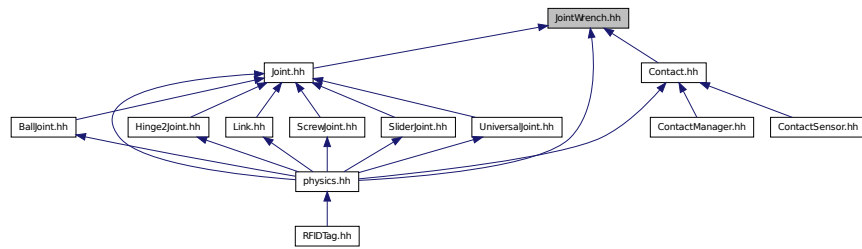
11.65 JointWrench.hh File Reference

```
#include "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

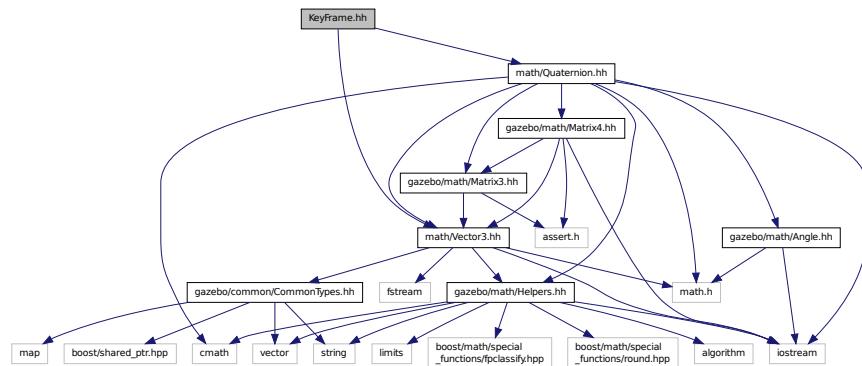
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.66 KeyFrame.hh File Reference

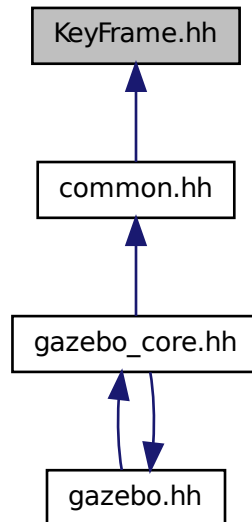
```
#include "math/Vector3.hh"
```

```
#include "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 (p. 109) key frame in an animation.
- class **gazebo::common::NumericKeyFrame**
*A (p. 109) keyframe for a **NumericAnimation** (p. 542).*
- class **gazebo::common::PoseKeyFrame**
*A (p. 109) keyframe for a **PoseAnimation** (p. 593).*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

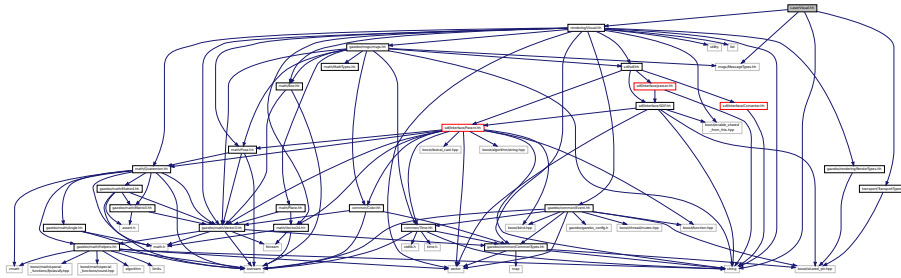
11.67 LaserVisual.hh File Reference

```

#include <string>
#include "rendering/Visual.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"

```

Include dependency graph for LaserVisual.hh:



Classes

- class **gazebo::rendering::LaserVisual**
Visualization for laser data.

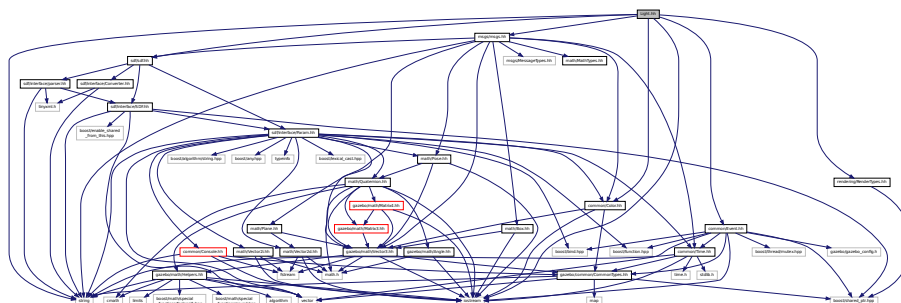
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.68 Light.hh File Reference

```
#include <string>
#include <iostream>
#include "msgs/msgs.hh"
#include "rendering/RenderTypes.hh"
#include "common/Event.hh"
#include "common/Color.hh"
#include "sdf/sdf.hh"
```

Include dependency graph for Light.hh:



Classes

- class **gazebo::rendering::Light**

A (p. 109) light source.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

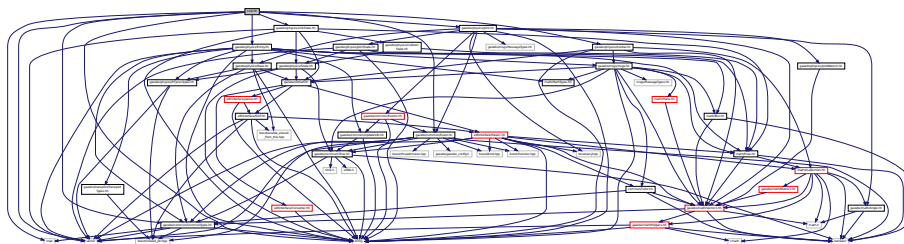
Rendering namespace.

- namespace **Ogre**

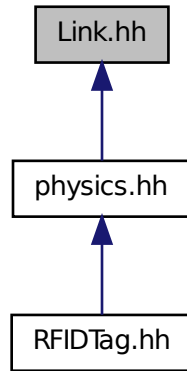
11.69 Link.hh File Reference

```
#include <map>
#include <vector>
#include <string>
#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. 412) 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.

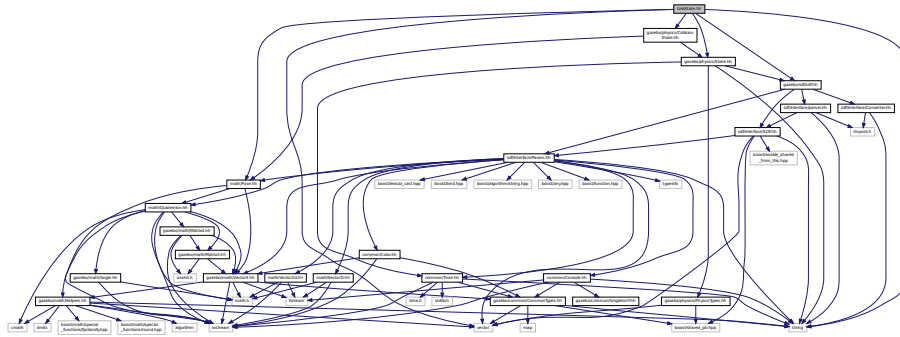
- namespace **gazebo::physics**

namespace for physics

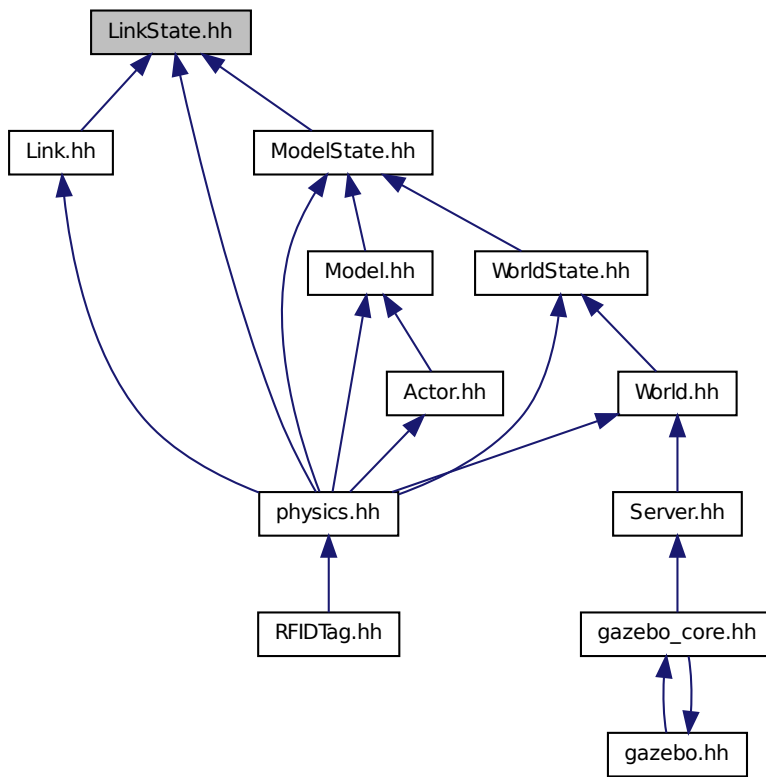
11.70 LinkState.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/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. 412) object.

Namespaces

- namespace **gazebo**

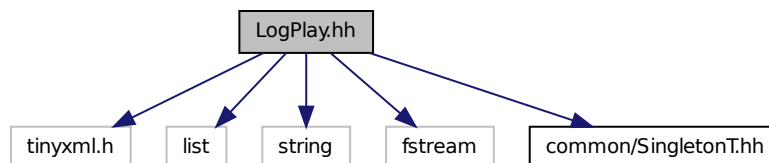
Forward declarations for the common classes.

- namespace **gazebo::physics**

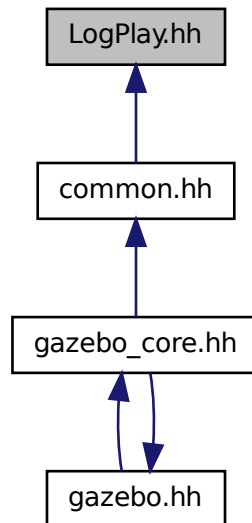
namespace for physics

11.71 LogPlay.hh File Reference

```
#include <tinyxml.h>
#include <list>
#include <string>
#include <fstream>
#include "common/SingletonT.hh"
Include dependency graph for LogPlay.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::LogPlay**

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.72 LogRecord.hh File Reference

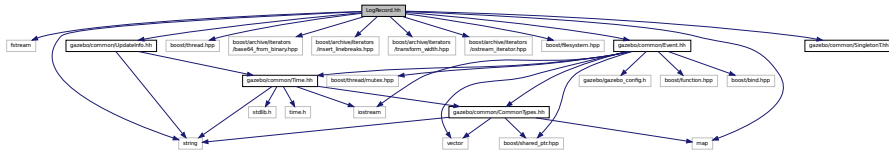
```
#include <fstream>
```

```

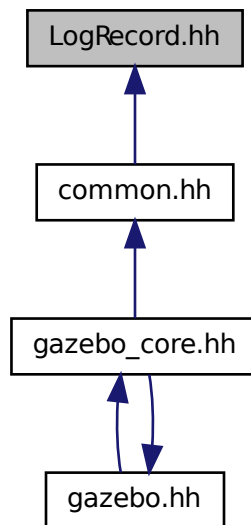
#include <string>
#include <map>
#include <boost/thread.hpp>
#include <boost/archive/iterators/base64_from_binary.hpp>
#include <boost/archive/iterators/insert_linebreaks.hpp>
#include <boost/archive/iterators/transform_width.hpp>
#include <boost/archive/iterators/ostream_iterator.hpp>
#include <boost/filesystem.hpp>
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/common/SingletonT.hh"

```

Include dependency graph for LogRecord.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::LogRecord**
addtogroup gazebo_common

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

Common namespace.

Macros

- `#define GZ_LOG_VERSION "1.0"`

11.72.1 Macro Definition Documentation

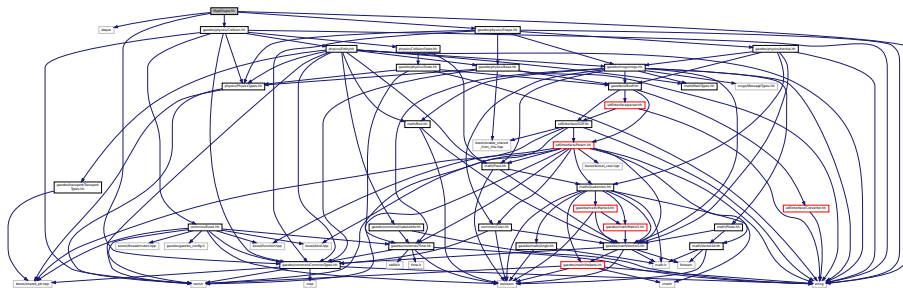
11.72.1.1 `#define GZ_LOG_VERSION "1.0"`

11.73 mainpage.html File Reference

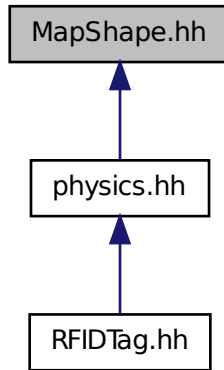
11.74 MapShape.hh File Reference

```
#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:



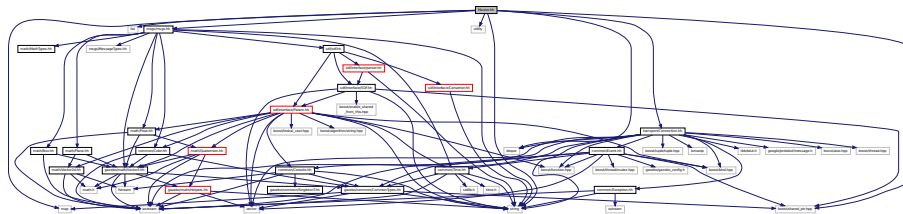
11.75 Master.hh File Reference

```

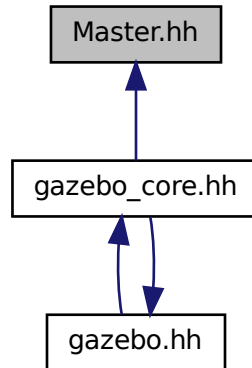
#include <string>
#include <list>
#include <deque>
#include <utility>
#include <map>
#include <boost/shared_ptr.hpp>
#include "msgs/msgs.hh"
#include "transport/Connection.hh"

```

Include dependency graph for Master.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::Master**

A (p. 109) ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

Namespaces

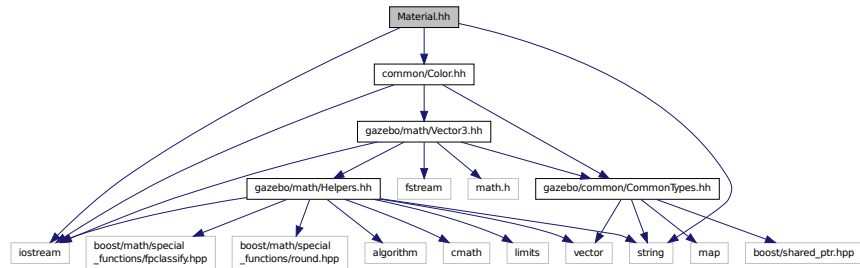
- namespace **gazebo**

Forward declarations for the common classes.

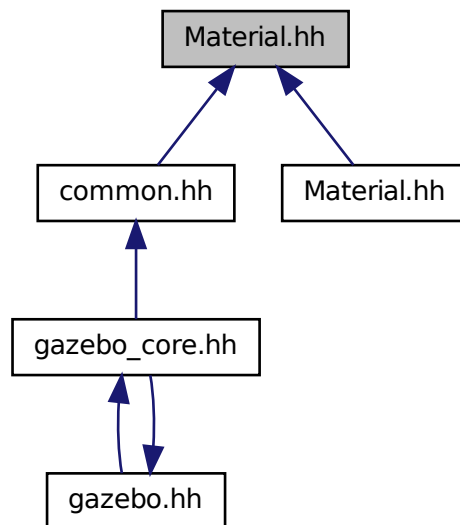
11.76 Material.hh File Reference

```
#include <string>
#include <iostream>
#include "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

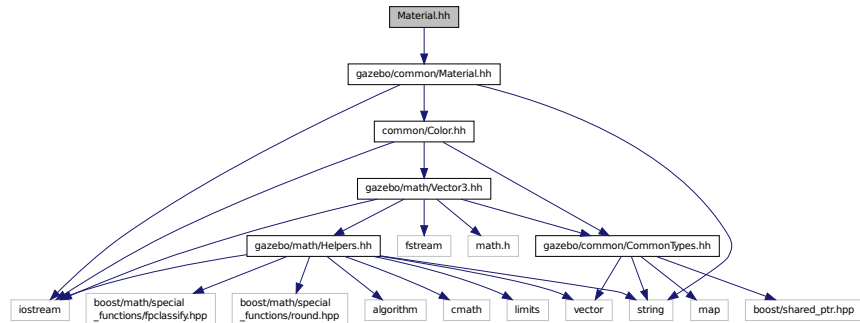
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**

Common namespace.

11.77 Material.hh File Reference

```
#include "gazebo/common/Material.hh"
```

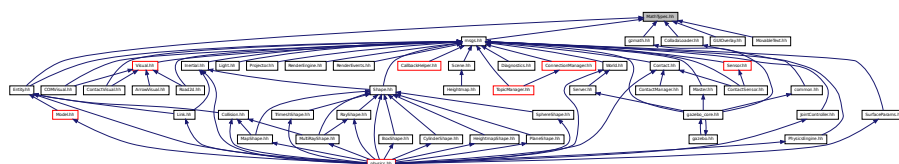
Include dependency graph for rendering/Material.hh:



11.78 MathTypes.hh File Reference

Forward declarations for the math classes.

This graph shows which files directly or indirectly include this file:



Namespaces

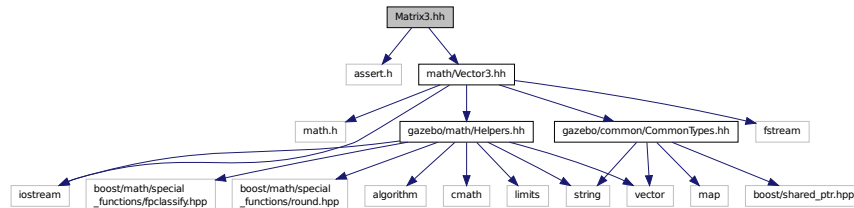
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

11.78.1 Detailed Description

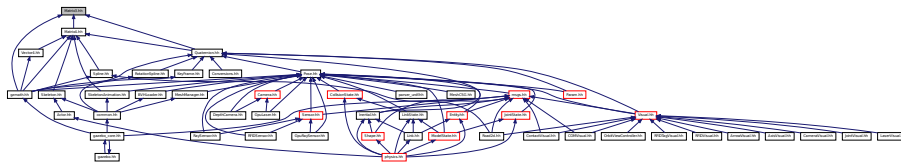
Forward declarations for the math classes.

11.79 Matrix3.hh File Reference

```
#include <assert.h>
#include "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 (p. 109) 3x3 matrix class.

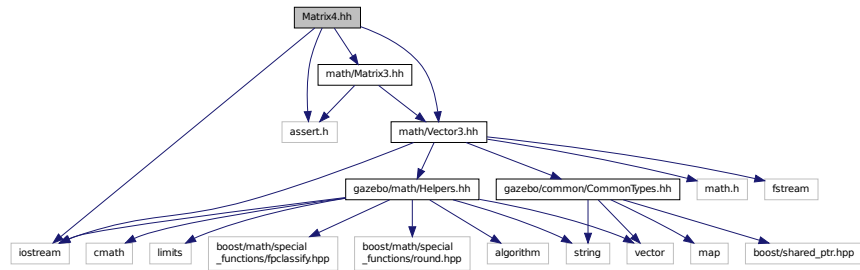
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

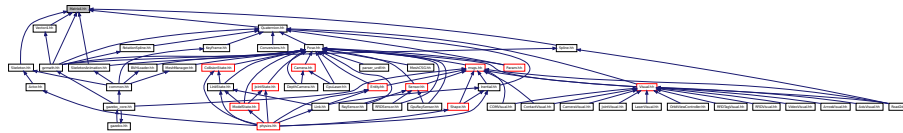
11.80 Matrix4.hh File Reference

```
#include <assert.h>
#include <iostream>
#include "math/Vector3.hh"
#include "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 (p. 109) 3x3 matrix class.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::math**

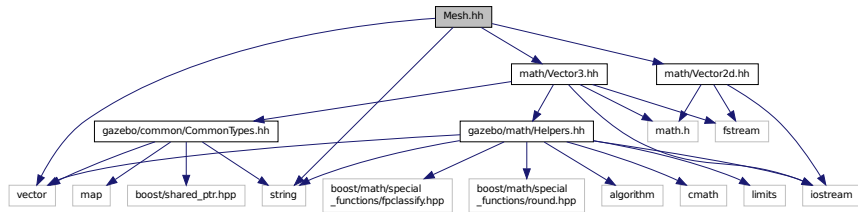
Math namespace.

11.81 Mesh.hh File Reference

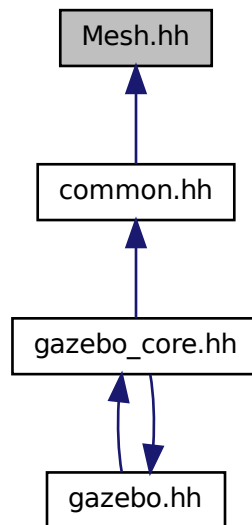
```

#include <vector>
#include <string>
#include "math/Vector3.hh"
#include "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 (p. 109) 3D mesh.
- struct **gazebo::common::NodeAssignment**
Vertex to node weighted assignment for skeleton animation visualization.
- class **gazebo::common::SubMesh**
A (p. 109) child mesh.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

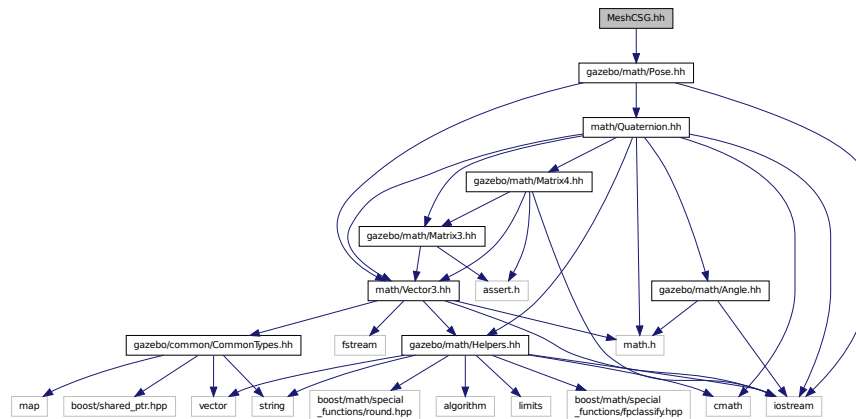
- namespace **gazebo::common**

Common namespace.

11.82 MeshCSG.hh File Reference

```
#include "gazebo/math/Pose.hh"
```

Include dependency graph for MeshCSG.hh:



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` **GPtArray**
- typedef `_GtsSurface` **GtsSurface**

11.82.1 Typedef Documentation

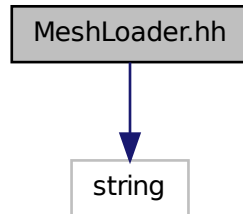
11.82.1.1 typedef `_GPtrArray` **GPtArray**

11.82.1.2 typedef `_GtsSurface` `GtsSurface`

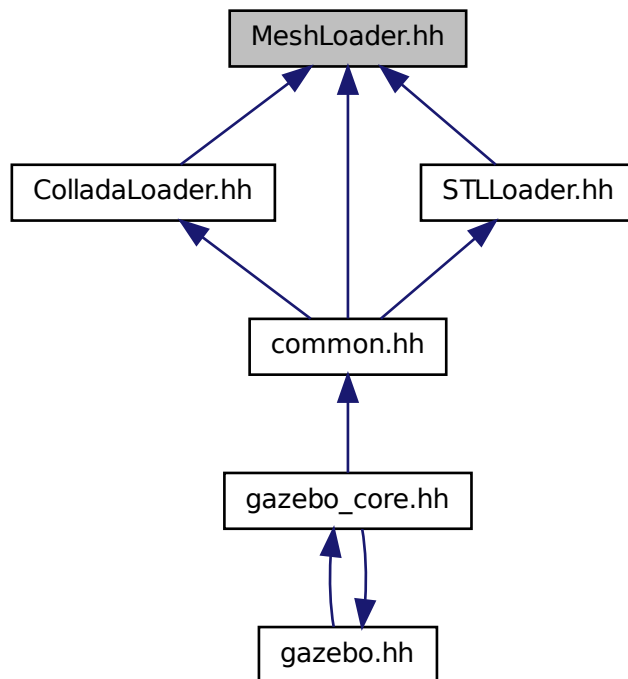
11.83 MeshLoader.hh File Reference

```
#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.

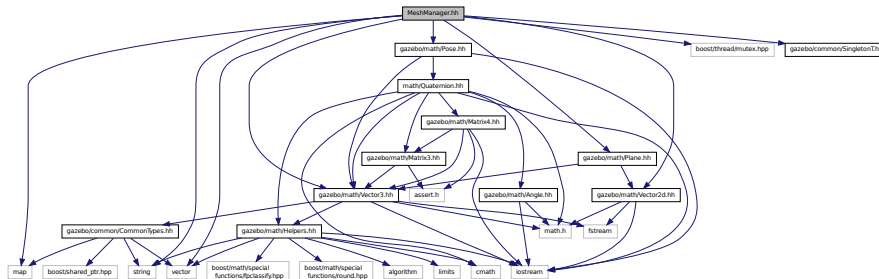
- namespace `gazebo::common`

Common namespace.

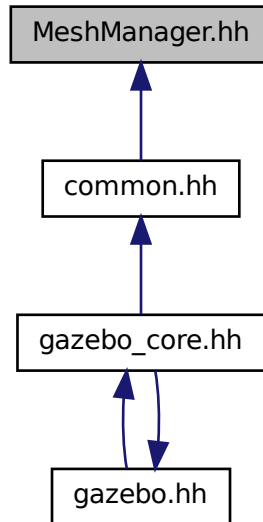
11.84 MeshManager.hh File Reference

```
#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.

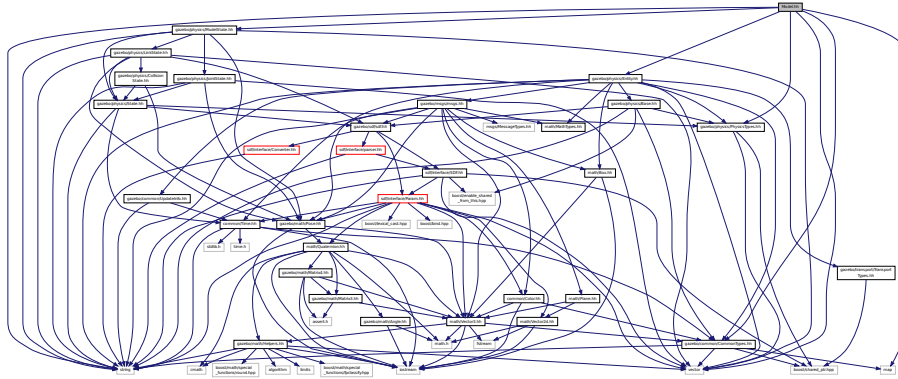
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

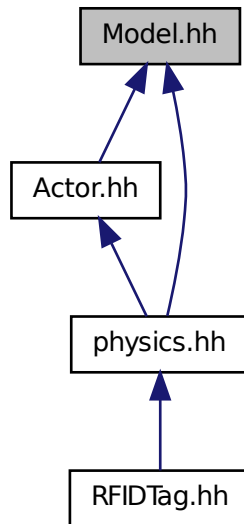
11.85 Model.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#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 (p. 109) model is a collection of links, joints, and plugins.

Namespaces

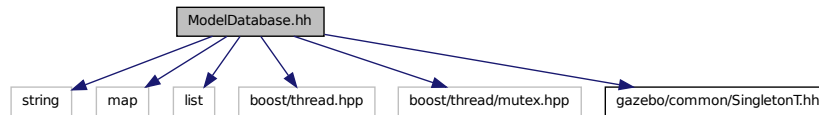
- namespace **boost**

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.86 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 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.

11.86.1 Macro Definition Documentation

11.86.1.1 #define GZ_MODEL_DB_MANIFEST_FILENAME "database.config"

The file name of model database XML configuration.

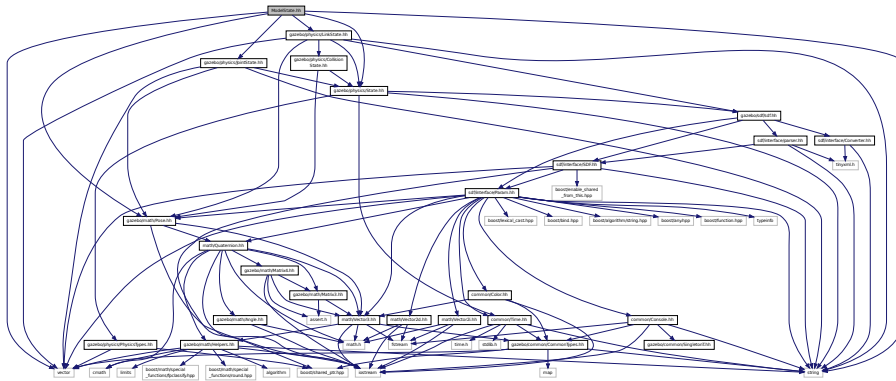
11.86.1.2 `#define GZ_MODEL_MANIFEST_FILENAME "model.config"`

The file name of model XML configuration.

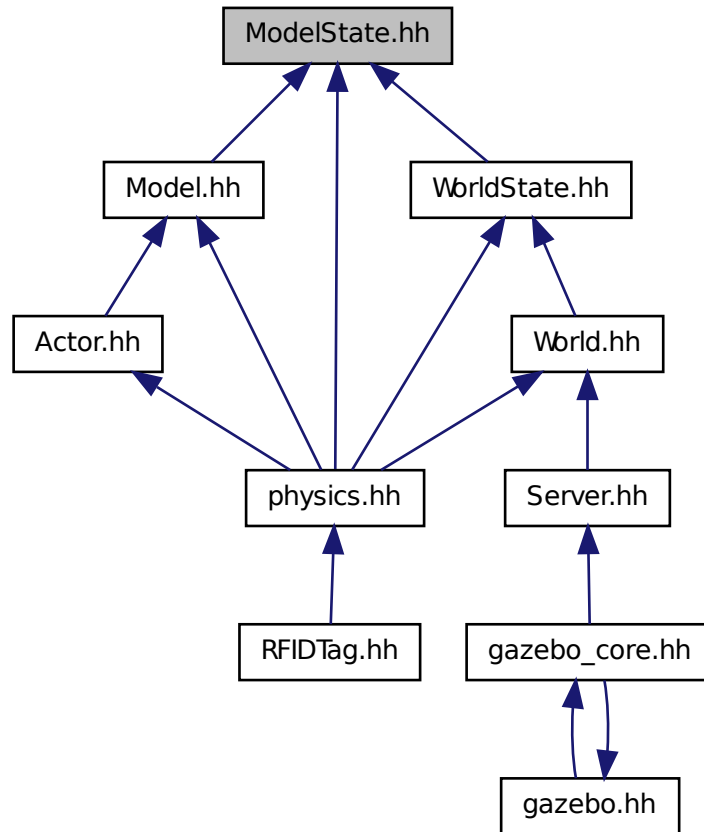
11.87 ModelState.hh File Reference

```
#include <vector>
#include <string>
#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. 480) object.

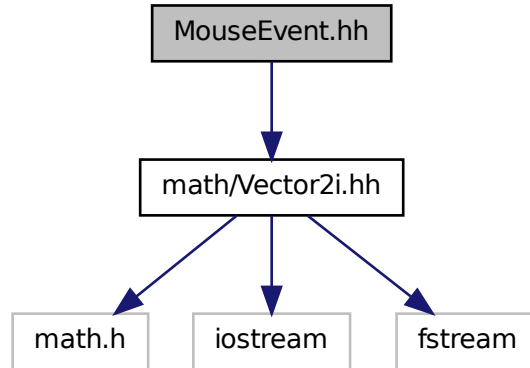
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

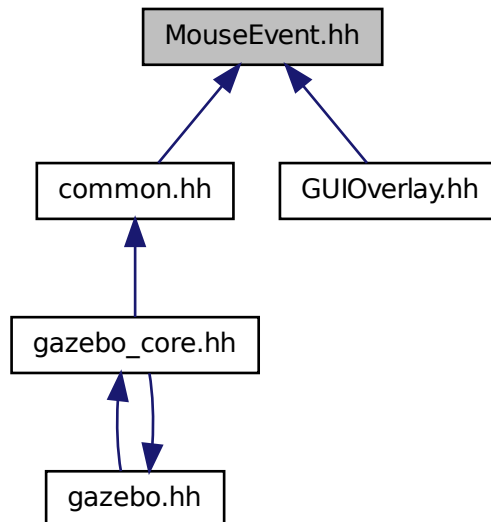
11.88 MouseEvent.hh File Reference

```
#include "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.

11.89 MovableText.hh File Reference

```
#include <string>
#include "rendering/ogre_gazebo.h"
#include "common/CommonTypes.hh"
#include "common/Color.hh"
#include "math/MathTypes.hh"
```

Include dependency graph for MovableText.hh:



Classes

- class **gazebo::rendering::MovableText**

Movable text.

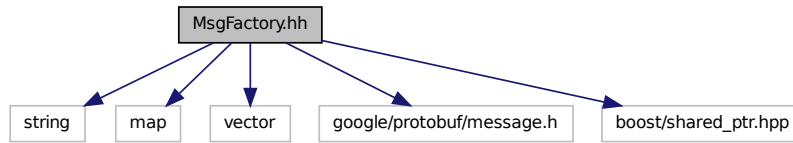
Namespaces

- namespace **boost**
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.90 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:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::msgs::MsgFactory**

A (p. 109) 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
`google::protobuf::Message *(* gazebo::msgs::MsgFactoryFn)()`

11.91 msgs.hh File Reference

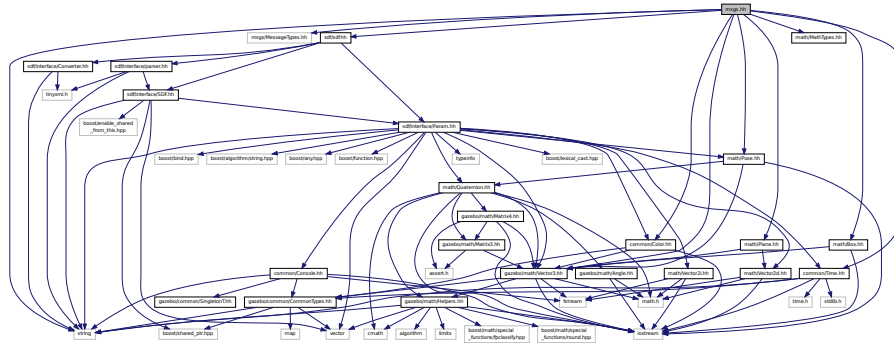
```
#include <string>
```

```

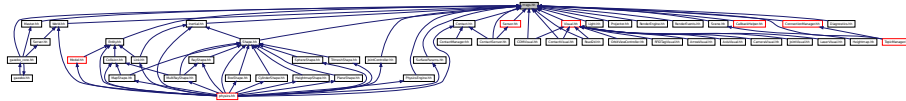
#include "msgs/MessageTypes.hh"
#include "sdf/sdf.hh"
#include "math/MathTypes.hh"
#include "math/Vector3.hh"
#include "math/Pose.hh"
#include "math/Plane.hh"
#include "math/Box.hh"
#include "common/Color.hh"
#include "common/Time.hh"

```

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. 839) to a msgs::Vector3d.*
- msgs::Quaternion **gazebo::msgs::Convert** (const math::Quaternion &_q)
*Convert a **math::Quaternion** (p. 609) to a msgs::Quaternion.*
- msgs::Pose **gazebo::msgs::Convert** (const math::Pose &_p)
*Convert a **math::Pose** (p. 584) to a msgs::Pose.*
- msgs::Color **gazebo::msgs::Convert** (const common::Color &_c)
*Convert a **common::Color** (p. 205) to a msgs::Color.*

- `msgs::Time gazebo::msgs::Convert` (const common::Time &_t)
*Convert a **common::Time** (p. 776) to a `msgs::Time`.*
- `msgs::PlaneGeom gazebo::msgs::Convert` (const math::Plane &_p)
*Convert a **math::Plane** (p. 575) 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. 609).*
- `math::Pose gazebo::msgs::Convert` (const msgs::Pose &_p)
*Convert a `msgs::Pose` to a **math::Pose** (p. 584).*
- `common::Color gazebo::msgs::Convert` (const msgs::Color &_c)
*Convert a `msgs::Color` to a **common::Color** (p. 205).*
- `common::Time gazebo::msgs::Convert` (const msgs::Time &_t)
*Convert a `msgs::Time` to a **common::Time** (p. 776).*
- `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::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::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. 356).*
- `void gazebo::msgs::Set` (`msgs::Image` *_msg, const common::Image &_i)
*Set a `msgs::Image` from a **common::Image** (p. 356).*
- `void gazebo::msgs::Set` (`msgs::Vector3d` *_pt, const math::Vector3 &_v)
*Set a `msgs::Vector3d` from a **math::Vector3** (p. 839).*
- `void gazebo::msgs::Set` (`msgs::Vector2d` *_pt, const math::Vector2d &_v)
*Set a `msgs::Vector2d` from a **math::Vector3** (p. 839).*
- `void gazebo::msgs::Set` (`msgs::Quaternion` *_q, const math::Quaternion &_v)
*Set a `msgs::Quaternion` from a **math::Quaternion** (p. 609).*
- `void gazebo::msgs::Set` (`msgs::Pose` *_p, const math::Pose &_v)
*Set a `msgs::Pose` from a **math::Pose** (p. 584).*
- `void gazebo::msgs::Set` (`msgs::Color` *_c, const common::Color &_v)
*Set a `msgs::Color` from a **common::Color** (p. 205).*
- `void gazebo::msgs::Set` (`msgs::Time` *_t, const common::Time &_v)
*Set a `msgs::Time` from a **common::Time** (p. 776).*
- `void gazebo::msgs::Set` (`msgs::PlaneGeom` *_p, const math::Plane &_v)

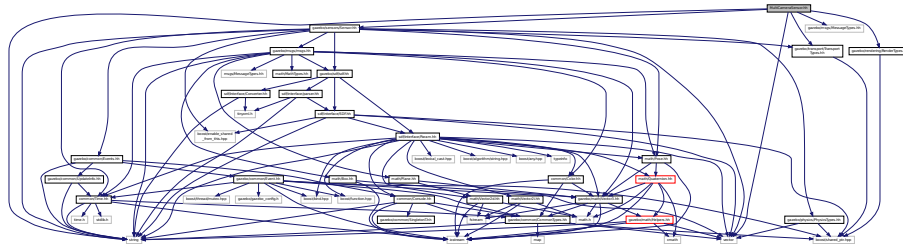
Set a `msgs::Plane` from a `math::Plane` (p. 575).

- 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.

11.92 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.

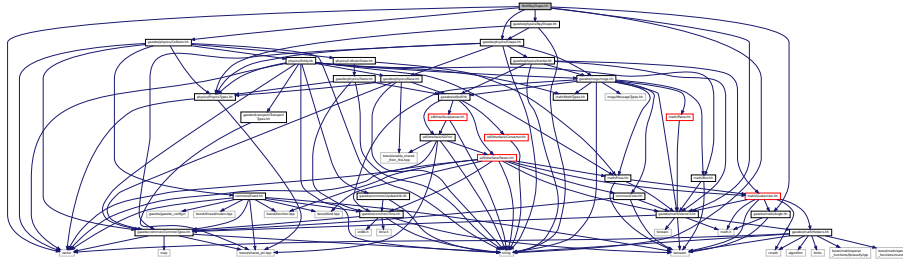
Namespaces

- namespace `gazebo`
Forward declarations for the common classes.
- namespace `gazebo::sensors`
Sensors namespace.

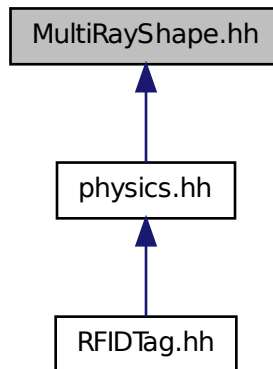
11.93 MultiRayShape.hh File Reference

```
#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

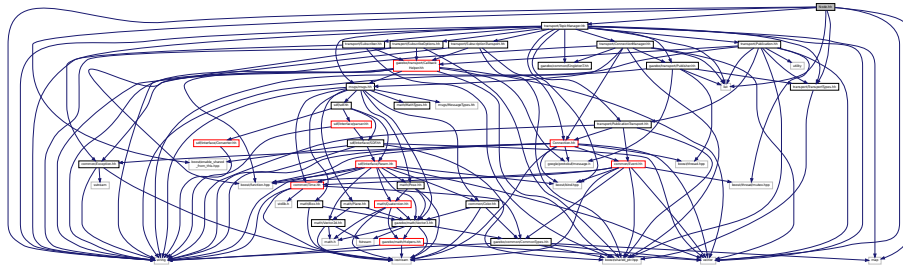
- class **gazebo::physics::MultiRayShape**
Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

Namespaces

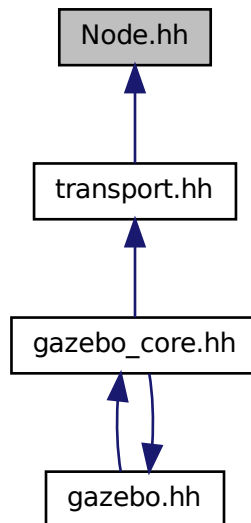
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.94 Node.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <map>
#include <list>
#include <string>
#include <vector>
#include "transport/TransportTypes.hh"
#include "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 (p. 109) 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**

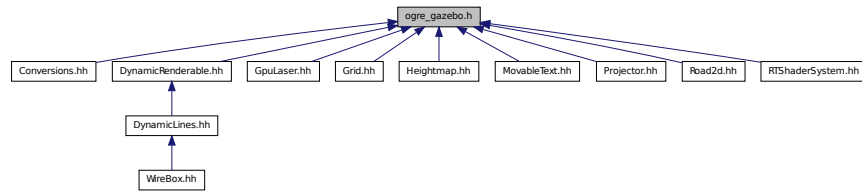
11.95 ogre_gazebo.h File Reference

```
#include <Ogre.h>
#include <OgreImageCodec.h>
#include <OGRE/OgreMovableObject.h>
#include <OGRE/OgreRenderable.h>
#include <OgrePlugin.h>
#include <OgreDataStream.h>
#include <OgreLogManager.h>
#include <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/Terrain/OgreTerrainMaterialGeneratorA.h>
#include <OGRE/Terrain/OgreTerrain.h>
#include <OGRE/Terrain/OgreTerrainGroup.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 dependency graph for ogre_gazebo.h:



This graph shows which files directly or indirectly include this file:



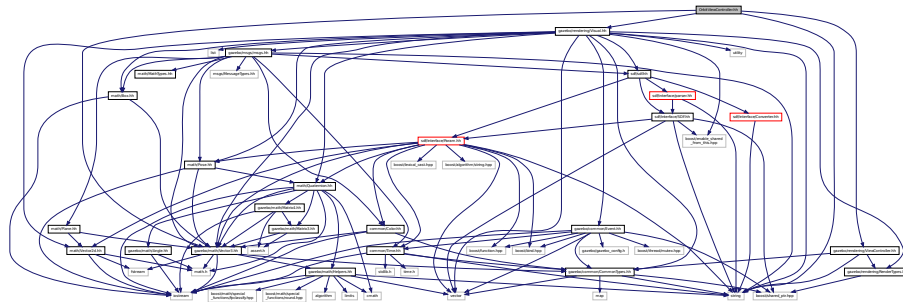
11.96 OrbitViewController.hh File Reference

```

#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.

11.97 Param.hh File Reference

```

#include <boost/lexical_cast.hpp>

```

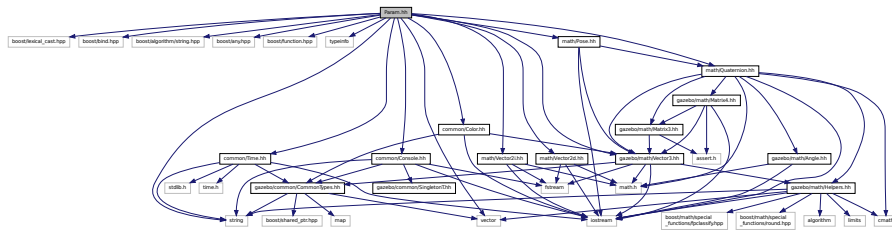


```

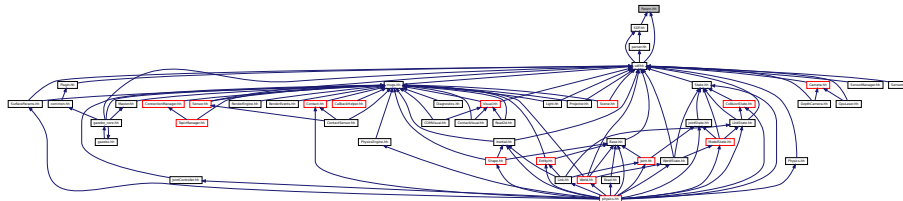
#include <boost/bind.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/any.hpp>
#include <boost/function.hpp>
#include <typeinfo>
#include <string>
#include <vector>
#include "common/Console.hh"
#include "common/Color.hh"
#include "common/Time.hh"
#include "math/Vector3.hh"
#include "math/Vector2i.hh"
#include "math/Vector2d.hh"
#include "math/Pose.hh"
#include "math/Quaternion.hh"

```

Include dependency graph for Param.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **sdf::Param**
A (p. 109) parameter class.
- class **sdf::ParamT< T >**
Templatized parameter class.

Namespaces

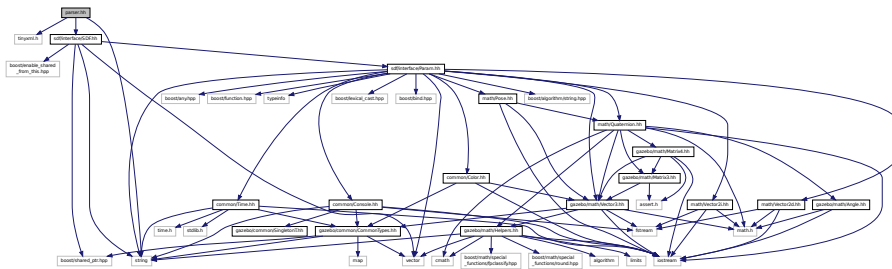
- namespace **sdf**
namespace for Simulation Description Format parser

Typedefs

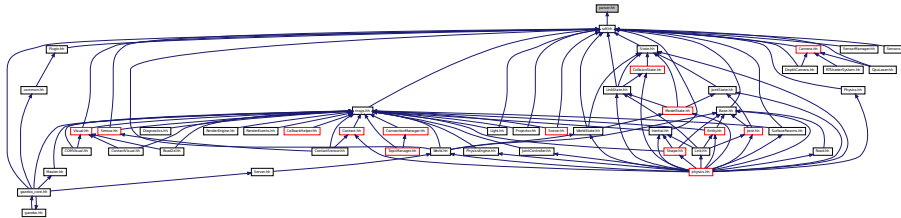
- typedef std::vector< ParamPtr > **sdf::Param_V**
- typedef Param * **sdf::ParamPtr**

11.98 parser.hh File Reference

```
#include <tinyxml.h>
#include <string>
#include "sdf/interface/SDF.hh"
Include dependency graph for parser.hh:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **sdf**
namespace for Simulation Description Format parser

Functions

- void **sdf::addNestedModel** (ElementPtr _sdf, ElementPtr _includeSDF)
- void **sdf::copyChildren** (ElementPtr _sdf, TiXmlElement * _xml)
- bool **sdf::init** (SDFPtr _sdf)
Init based on the installed sdf_format.xml file.
- bool **sdf::initDoc** (TiXmlDocument * _xmlDoc, SDFPtr _sdf)
- bool **sdf::initDoc** (TiXmlDocument * _xmlDoc, ElementPtr _sdf)
- bool **sdf::initFile** (const std::string & _filename, SDFPtr _sdf)
- bool **sdf::initFile** (const std::string & _filename, ElementPtr _sdf)

- bool **sdf::initString** (const std::string &_xmlString, SDFPtr _sdf)
- bool **sdf::initXml** (TiXmlElement * _xml, ElementPtr _sdf)
- bool **sdf::readDoc** (TiXmlDocument * _xmlDoc, SDFPtr _sdf, const std::string &_source)

Populate the **SDF** (p. 680) values from a TinyXML document.
- bool **sdf::readDoc** (TiXmlDocument * _xmlDoc, ElementPtr _sdf, const std::string &_source)
- bool **sdf::readFile** (const std::string &_filename, SDFPtr _sdf)

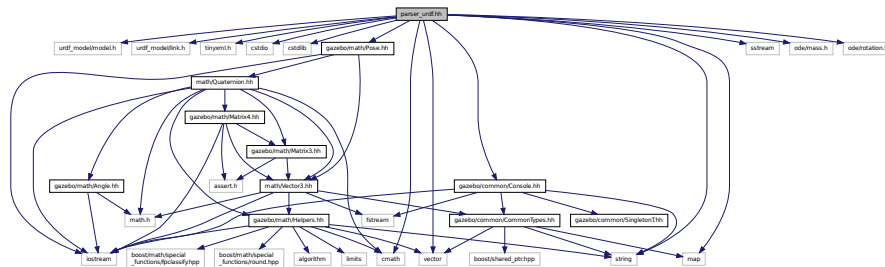
Populate the **SDF** (p. 680) values from a file.
- bool **sdf::readString** (const std::string &_xmlString, SDFPtr _sdf)

Populate the **SDF** (p. 680) values from a string.
- bool **sdf::readString** (const std::string &_xmlString, ElementPtr _sdf)
- bool **sdf::readXml** (TiXmlElement * _xml, ElementPtr _sdf)

11.99 parser_urdf.hh File Reference

```
#include <urdf_model/model.h>
#include <urdf_model/link.h>
#include <tinyxml.h>
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <vector>
#include <string>
#include <sstream>
#include <map>
#include "ode/mass.h"
#include "ode/rotation.h"
#include "gazebo/math/Pose.hh"
#include "gazebo/common/Console.hh"
```

Include dependency graph for parser_urdf.hh:



Classes

- class **urdf2gazebo::GazeboExtension**
- class **urdf2gazebo::URDF2Gazebo**

Namespaces

- namespace **urdf2gazebo**

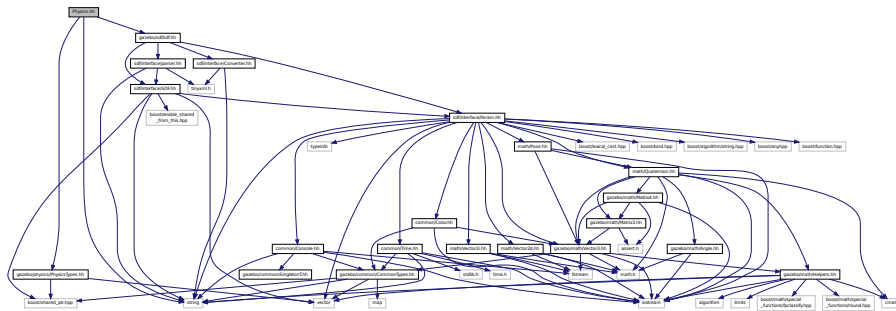
namespace for URDF to SDF parser

Typedefs

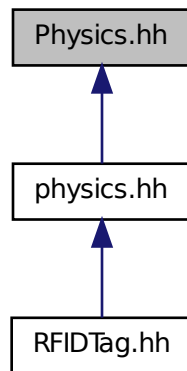
- typedef const urdf::Link * **urdf2gazebo::ConstUrdfLinkPtr**
- typedef urdf::Collision * **urdf2gazebo::UrdfCollisionPtr**
- typedef urdf::Link * **urdf2gazebo::UrdfLinkPtr**
- typedef urdf::Visual * **urdf2gazebo::UrdfVisualPtr**

11.100 Physics.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sdf/sdf.hh"
Include dependency graph for Physics.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

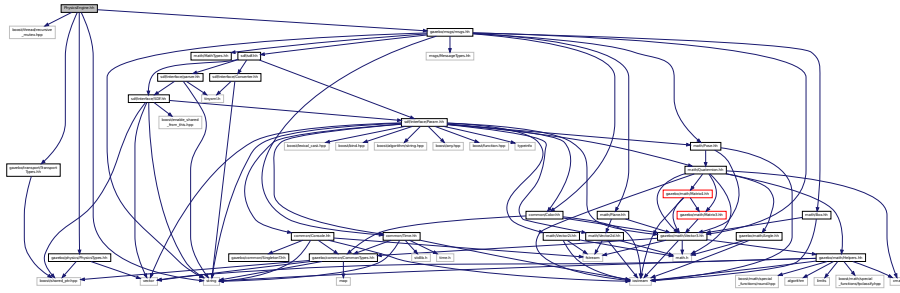
Functions

- 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. 76).*
- WorldPtr **gazebo::physics::get_world** (const std::string &_name="")
Returns a pointer to a world by name.
- 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. 774)'s and call **gazebo::transport::init** (p. 77).*
- void **gazebo::physics::load_world** (WorldPtr _world, sdf::ElementPtr _sdf)
*Load world from **sdf::Element** (p. 269) pointer.*
- void **gazebo::physics::load_worlds** (sdf::ElementPtr _sdf)
*load multiple worlds from single **sdf::Element** (p. 269) pointer*
- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)
*Pause world by calling **World::SetPaused** (p. 903).*
- 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)
*Run world by calling **World::Run()** (p. 903) given a pointer to it.*
- void **gazebo::physics::run_worlds** ()
run multiple worlds stored in static variable gazebo::g_worlds
- void **gazebo::physics::stop_world** (WorldPtr _world)
*Stop world by calling **World::Stop()** (p. 904) given a pointer to it.*
- void **gazebo::physics::stop_worlds** ()
stop multiple worlds stored in static variable gazebo::g_worlds

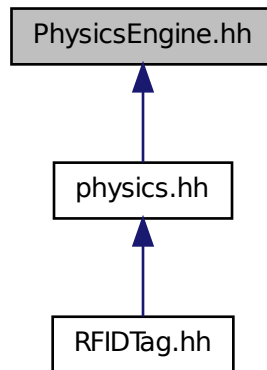
11.101 PhysicsEngine.hh File Reference

```
#include <boost/thread/recursive_mutex.hpp>
#include <string>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/messages/messages.hh"
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for PhysicsEngine.hh:



This graph shows which files directly or indirectly include this file:



Classes

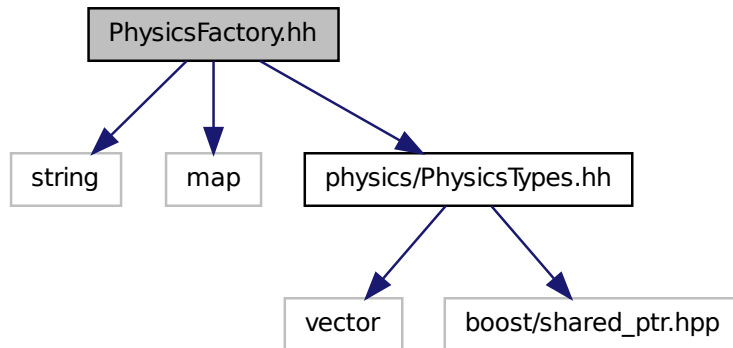
- class **gazebo::physics::PhysicsEngine**
Base (p. 135) class for a physics engine.

Namespaces

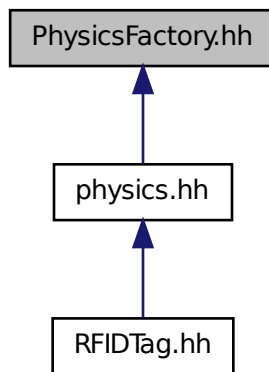
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.102 PhysicsFactory.hh File Reference

```
#include <string>
#include <map>
#include "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.

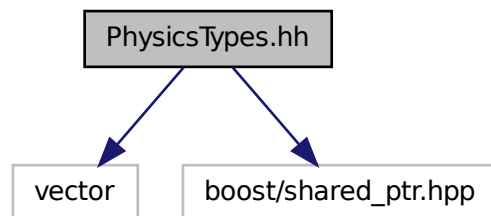
Typedefs

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

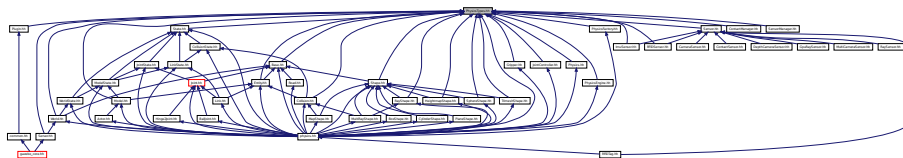
11.103 PhysicsTypes.hh File Reference

default namespace for gazebo

```
#include <vector>
#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 Actor * **gazebo::physics::ActorPtr**
- typedef std::vector< BasePtr > **gazebo::physics::Base_V**
- typedef Base * **gazebo::physics::BasePtr**
- typedef BoxShape * **gazebo::physics::BoxShapePtr**
- typedef std::vector< CollisionPtr > **gazebo::physics::Collision_V**
- typedef Collision * **gazebo::physics::CollisionPtr**
- typedef Contact * **gazebo::physics::ContactPtr**
- typedef CylinderShape * **gazebo::physics::CylinderShapePtr**
- typedef Entity * **gazebo::physics::EntityPtr**
- typedef HeightmapShape * **gazebo::physics::HeightmapShapePtr**
- typedef Inertial * **gazebo::physics::InertialPtr**
- typedef std::vector< JointPtr > **gazebo::physics::Joint_V**
- typedef std::vector
< JointControllerPtr > **gazebo::physics::JointController_V**
- typedef JointController * **gazebo::physics::JointControllerPtr**
- typedef Joint * **gazebo::physics::JointPtr**
- typedef std::vector< LinkPtr > **gazebo::physics::Link_V**
- typedef Link * **gazebo::physics::LinkPtr**
- typedef MeshShape * **gazebo::physics::MeshShapePtr**
- typedef std::vector< ModelPtr > **gazebo::physics::Model_V**
- typedef Model * **gazebo::physics::ModelPtr**
- typedef MultiRayShape * **gazebo::physics::MultiRayShapePtr**
- typedef PhysicsEngine * **gazebo::physics::PhysicsEnginePtr**
- typedef RayShape * **gazebo::physics::RayShapePtr**
- typedef **Road** * **gazebo::physics::RoadPtr**
- typedef Shape * **gazebo::physics::ShapePtr**
- typedef SphereShape * **gazebo::physics::SphereShapePtr**
- typedef SurfaceParams * **gazebo::physics::SurfaceParamsPtr**
- typedef World * **gazebo::physics::WorldPtr**

11.103.1 Detailed Description

default namespace for gazebo

11.103.2 Macro Definition Documentation

11.103.2.1 #define GZ_ALL_COLLIDE 0x0FFFFFFF

Default collision bitmask.

Collision objects will collide with everything.

11.103.2.2 #define GZ_FIXED_COLLIDE 0x00000001

Collision object will collide only with fixed objects.

11.103.2.3 #define GZ_GHOST_COLLIDE 0x10000000

Collides with everything else but other ghost.

11.103.2.4 #define GZ_NONE_COLLIDE 0x00000000

Collision object will collide with nothing.

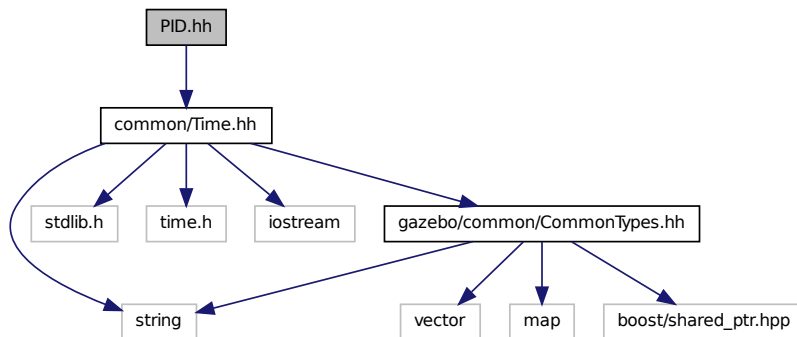
11.103.2.5 #define GZ_SENSOR_COLLIDE 0x00000002

Collision object will collide only with sensors.

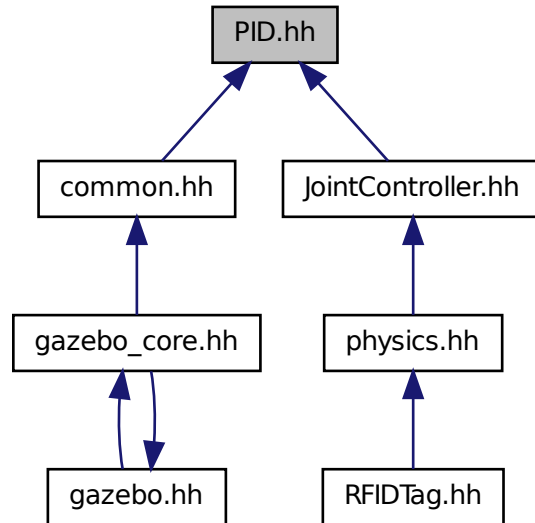
11.104 PID.hh File Reference

```
#include "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. 571) controller class.*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

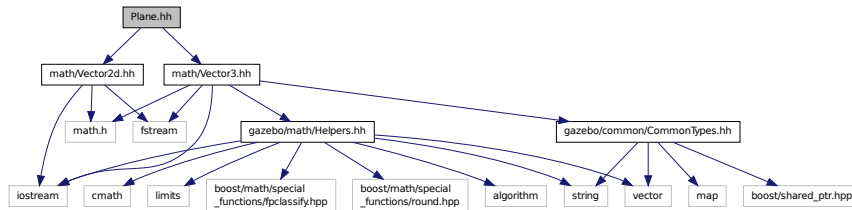
- namespace **gazebo::common**

Common namespace.

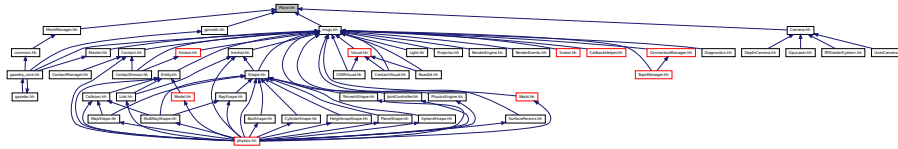
11.105 Plane.hh File Reference

```
#include "math/Vector3.hh"
#include "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 (p. 109) plane and related functions.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

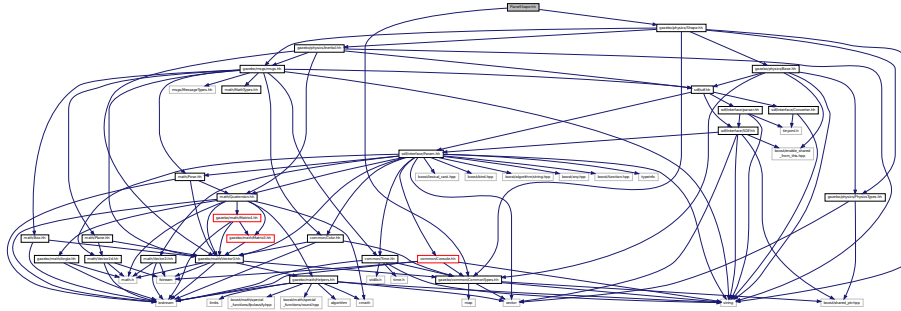
- namespace **gazebo::math**

Math namespace.

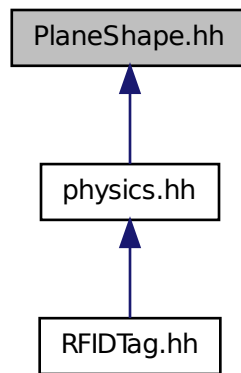
11.106 PlaneShape.hh File Reference

```
#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. 192) for an infinite plane.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

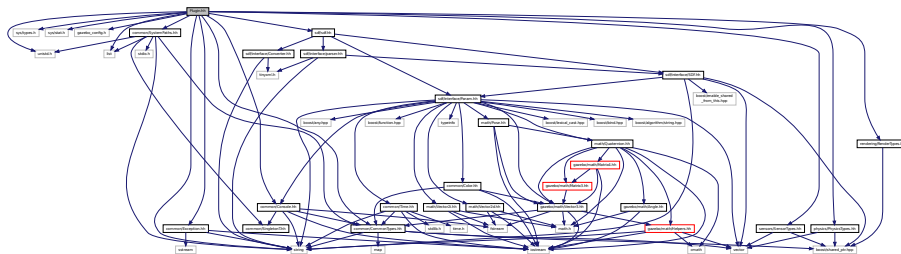
11.107 Plugin.hh File Reference

```

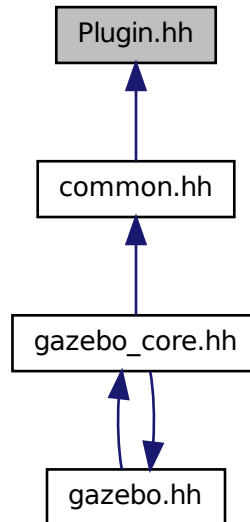
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <gazebo_config.h>
#include <list>
#include <string>
#include "common/CommonTypes.hh"
#include "common/SystemPaths.hh"
#include "common/Console.hh"
#include "common/Exception.hh"
#include "physics/PhysicsTypes.hh"
#include "sensors/SensorTypes.hh"
#include "sdf/sdf.hh"
#include "rendering/RenderTypes.hh"

```

Include dependency graph for common/Plugin.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::ModelPlugin**
*A (p. 109) plugin with access to **physics::Model** (p. 480).*
- class **gazebo::PluginT< T >**
A (p. 109) class which all plugins must inherit from.
- class **gazebo::SensorPlugin**
*A (p. 109) plugin with access to **physics::Sensor**.*
- class **gazebo::SystemPlugin**
A (p. 109) plugin loaded within the gzserver on startup.
- class **gazebo::VisualPlugin**
A (p. 109) plugin loaded within the gzserver on startup.
- class **gazebo::WorldPlugin**
*A (p. 109) plugin with access to **physics::World** (p. 893).*

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.

11.107.1 Macro Definition Documentation

11.107.1.1 #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

11.107.1.2 #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

11.107.1.3 #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

11.107.1.4 #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

11.107.1.5 #define GZ_REGISTER_WORLD_PLUGIN(*classname*)**Value:**

```
extern "C" gazebo::WorldPlugin *RegisterPlugin();    gazebo::WorldPlugin *RegisterPlugin() \
{\
    return new classname();\
}
```

Plugin registration function for world plugin.

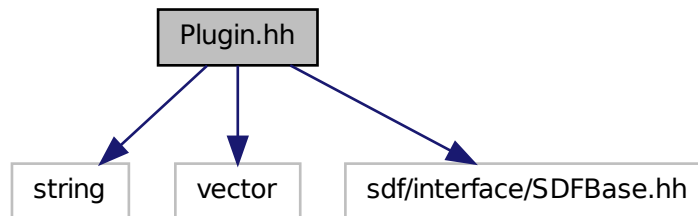
Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.108 Plugin.hh File Reference

```
#include <string>
#include <vector>
#include "sdf/interface/SDFBase.hh"
Include dependency graph for sdf/interface/Plugin.hh:
```



Classes

- class `sdf::Plugin`

Namespaces

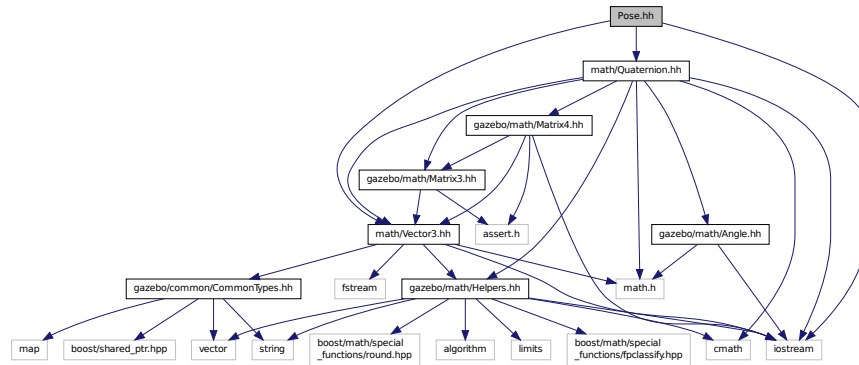
- namespace `sdf`

namespace for Simulation Description Format parser

11.109 Pose.hh File Reference

```
#include <iostream>
#include "math/Vector3.hh"
#include "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.

11.110 Projector.hh File Reference

```
#include <string>
#include <map>
#include <list>
#include "rendering/ogre_gazebo.h"
#include "msgs/msgs.hh"
#include "sdf/sdf.hh"
#include "transport/transport.hh"
#include "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.

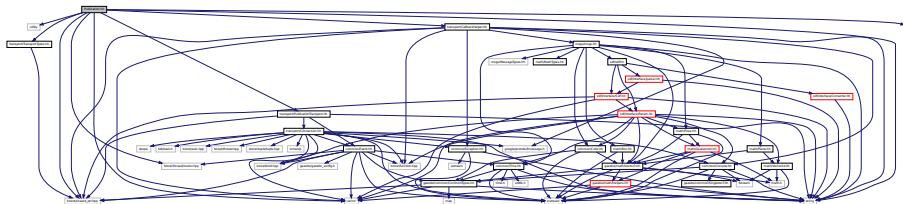
- namespace **gazebo::rendering**

Rendering namespace.

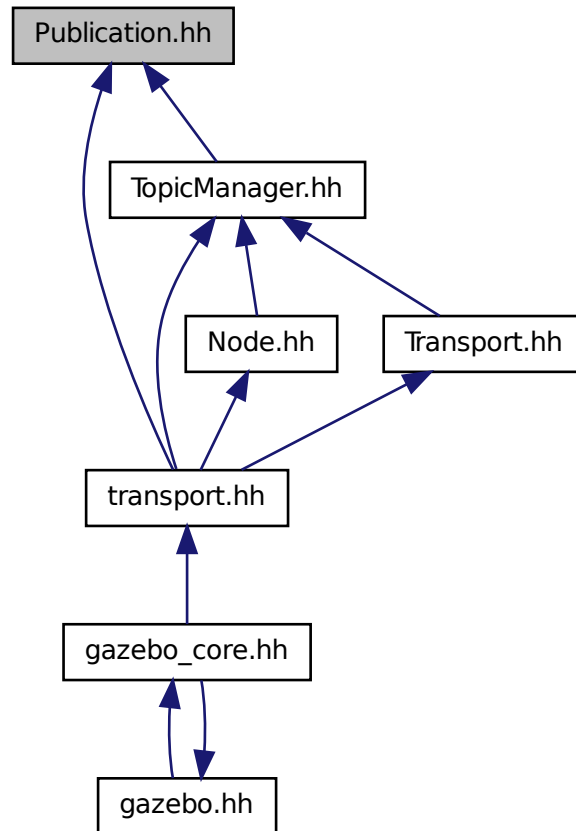
11.111 Publication.hh File Reference

```
#include <utility>
#include <boost/shared_ptr.hpp>
#include <boost/thread/mutex.hpp>
#include <list>
#include <string>
#include <vector>
#include "transport/CallbackHelper.hh"
#include "transport/TransportTypes.hh"
#include "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 (p. 109) publication for a topic.

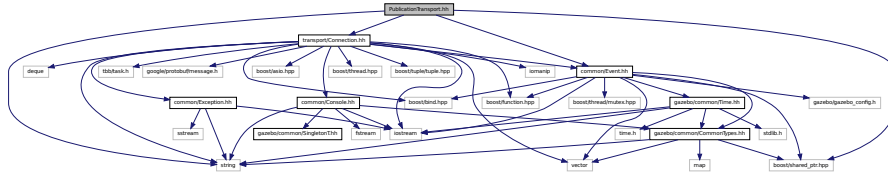
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

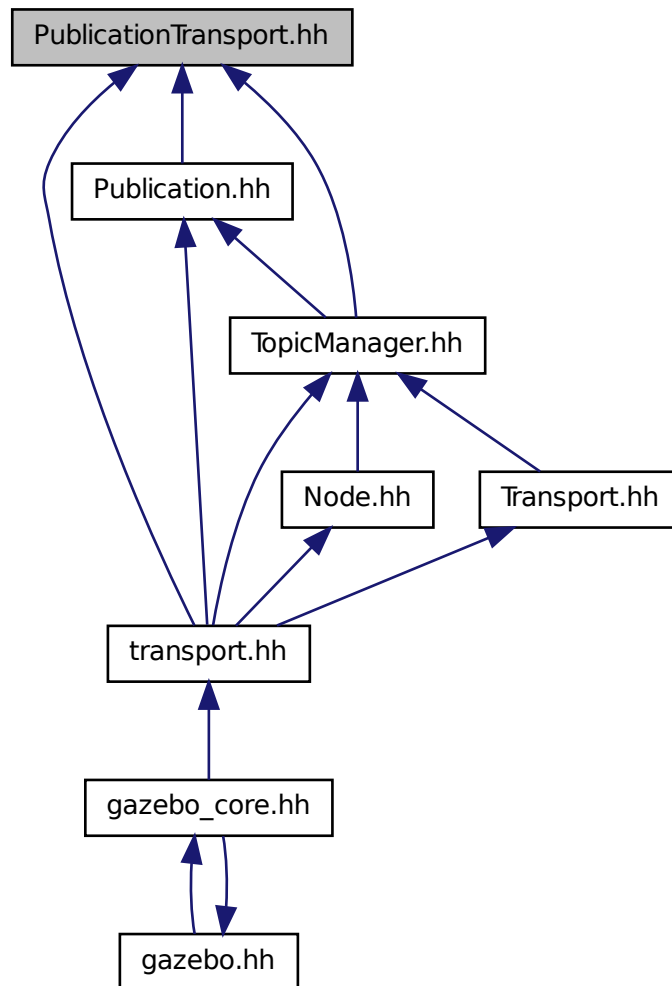
11.112 PublicationTransport.hh File Reference

```
#include <boost/shared_ptr.hpp>
```

```
#include <string>
#include "transport/Connection.hh"
#include "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

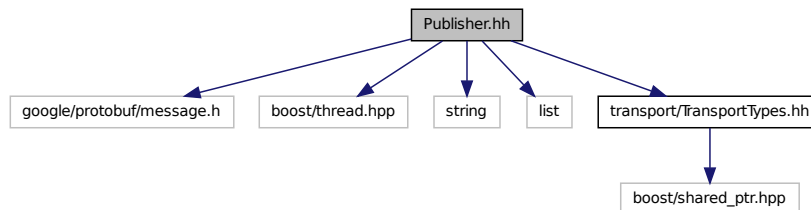
- namespace **gazebo**

Forward declarations for the common classes.

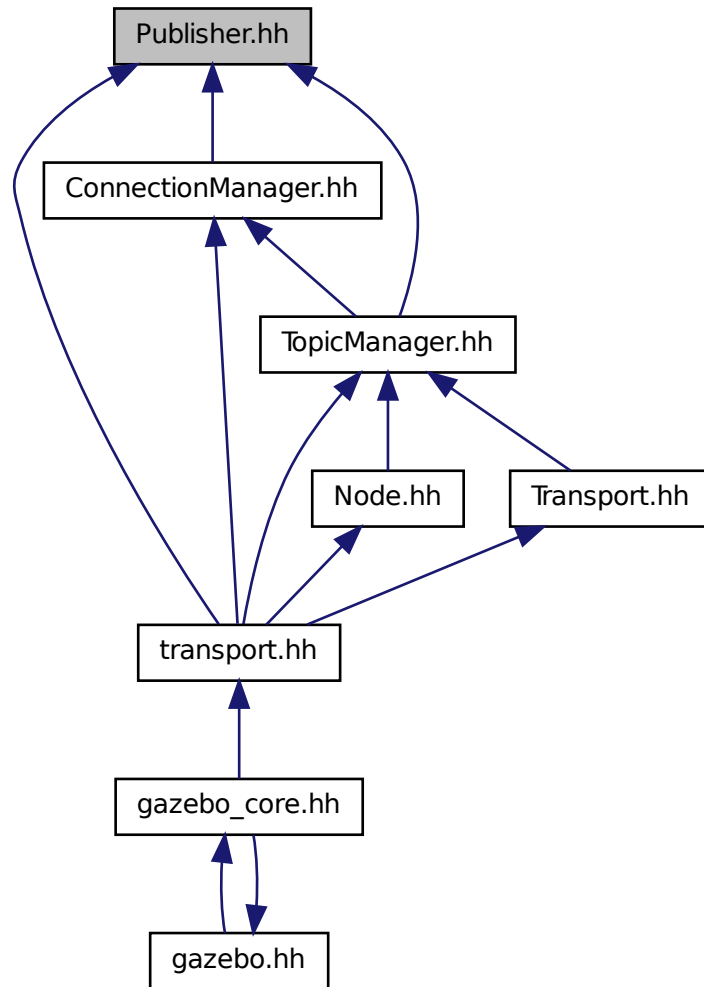
- namespace **gazebo::transport**

11.113 Publisher.hh File Reference

```
#include <google/protobuf/message.h>
#include <boost/thread.hpp>
#include <string>
#include <list>
#include "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 (p. 109) publisher of messages on a topic.

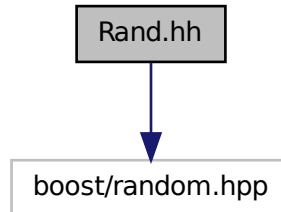
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

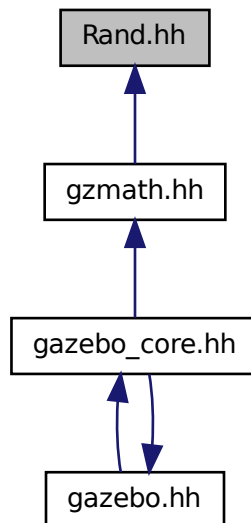
11.115 Rand.hh File Reference

```
#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.

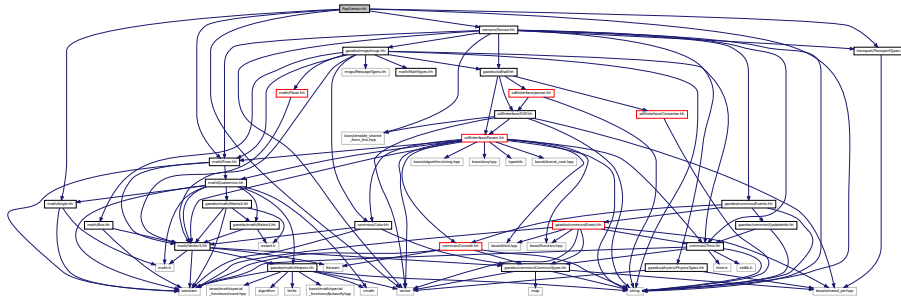
Typedefs

- 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**

11.116 RaySensor.hh File Reference

```
#include <vector>
#include <string>
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "transport/TransportTypes.hh"
#include "sensors/Sensor.hh"
```

Include dependency graph for RaySensor.hh:



Classes

- class **gazebo::sensors::RaySensor**

Sensor (p. 683) with one or more rays.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

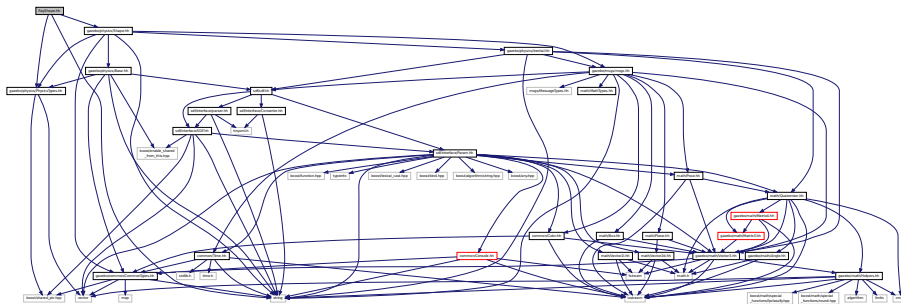
- namespace **gazebo::sensors**

Sensors namespace.

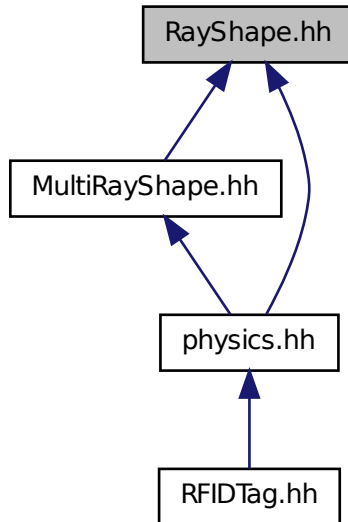
11.117 RayShape.hh File Reference

```
#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. 135) class for Ray collision geometry.

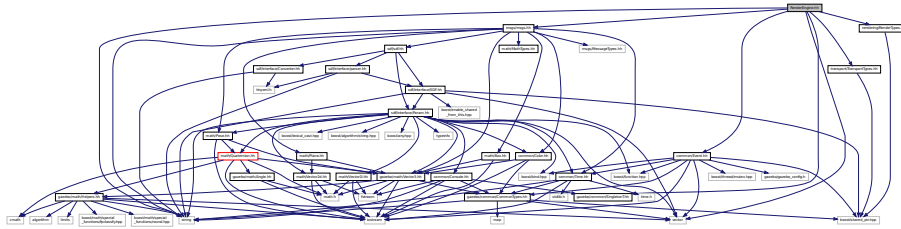
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.118 RenderEngine.hh File Reference

```
#include <vector>
#include <string>
#include "msgs/msgs.hh"
#include "common/SingletonT.hh"
#include "common/Event.hh"
#include "transport/TransportTypes.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for RenderEngine.hh:



Classes

- class **gazebo::rendering::RenderEngine**
Adaptor to Ogre3d.

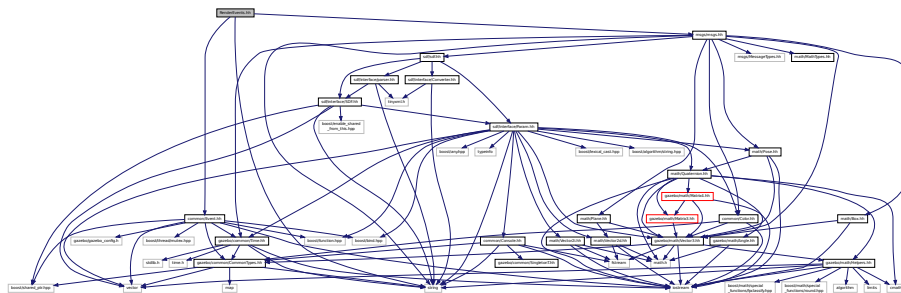
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.119 RenderEvents.hh File Reference

```
#include <string>
#include "common/Event.hh"
#include "msgs/msgs.hh"
```

Include dependency graph for RenderEvents.hh:



Classes

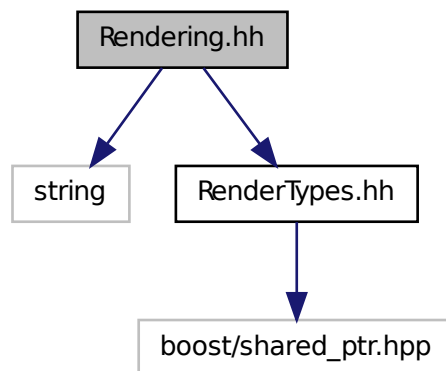
- class **gazebo::rendering::Events**
Base class for rendering events.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.120 Rendering.hh File Reference

```
#include <string>
#include "RenderTypes.hh"
Include dependency graph for Rendering.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)

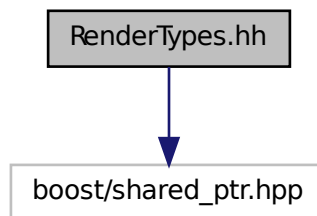
*create **rendering::Scene** (p. 662) by name.*
- bool **gazebo::rendering::fini** ()
teardown rendering engine.
- rendering::ScenePtr **gazebo::rendering::get_scene** (const std::string &_name)
*get pointer to **rendering::Scene** (p. 662) by name.*

- 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. 662) by name*

11.121 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_NOT_SELECTABLE 0x00000002`
Render visuals that are not selectable mask.
- `#define GZ_VISIBILITY_SELECTION 0x10000000`
Renders only objects that can be selected.

Typedefs

- `typedef ArrowVisual * gazebo::rendering::ArrowVisualPtr`
- `typedef AxisVisual * gazebo::rendering::AxisVisualPtr`
- `typedef Camera * gazebo::rendering::CameraPtr`
- `typedef CameraVisual * gazebo::rendering::CameraVisualPtr`
- `typedef COMVisual * gazebo::rendering::COMVisualPtr`
- `typedef ContactVisual * gazebo::rendering::ContactVisualPtr`
- `typedef DepthCamera * gazebo::rendering::DepthCameraPtr`
- `typedef DynamicLines * gazebo::rendering::DynamicLinesPtr`
- `typedef GpuLaser * gazebo::rendering::GpuLaserPtr`
- `typedef JointVisual * gazebo::rendering::JointVisualPtr`
- `typedef LaserVisual * gazebo::rendering::LaserVisualPtr`
- `typedef Light * gazebo::rendering::LightPtr`
- `typedef RFIDTagVisual * gazebo::rendering::RFIDTagVisualPtr`
- `typedef RFIDVisual * gazebo::rendering::RFIDVisualPtr`
- `typedef Scene * gazebo::rendering::ScenePtr`
- `typedef UserCamera * gazebo::rendering::UserCameraPtr`
- `typedef Visual * gazebo::rendering::VisualPtr`

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.

11.121.1 Macro Definition Documentation

11.121.1.1 `#define GZ_VISIBILITY_ALL 0xFFFFFFFF`

Render everything visibility mask.

11.121.1.2 `#define GZ_VISIBILITY_GUI 0x00000001`

Render GUI visuals mask.

11.121.1.3 `#define GZ_VISIBILITY_NOT_SELECTABLE 0x00000002`

Render visuals that are not selectable mask.

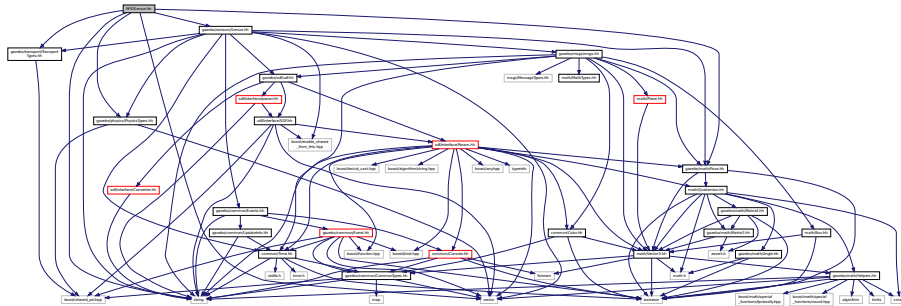
11.121.1.4 `#define GZ_VISIBILITY_SELECTION 0x10000000`

Renders only objects that can be selected.

11.122 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. 683) class for RFID type of sensor.

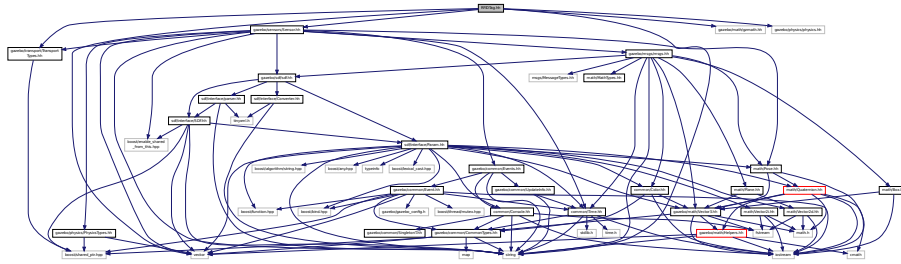
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.123 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. 646) to interact with *RFIDTagSensors*.

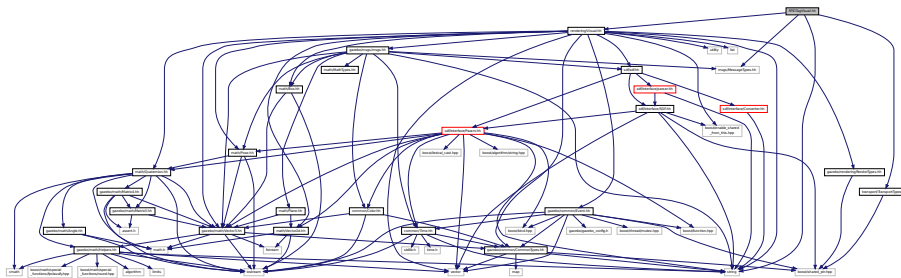
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.124 RFIDTagVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
```

Include dependency graph for RFIDTagVisual.hh:



Classes

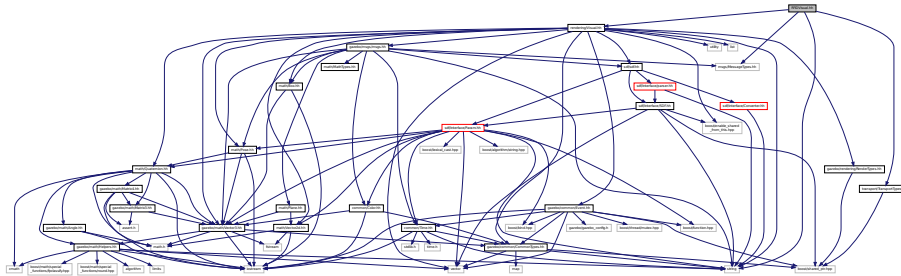
- class **gazebo::rendering::RFIDTagVisual**
Visualization for RFID tags sensor.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.125 RFIDVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
Include dependency graph for RFIDVisual.hh:
```



Classes

- class **gazebo::rendering::RFIDVisual**
Visualization for RFID sensor.

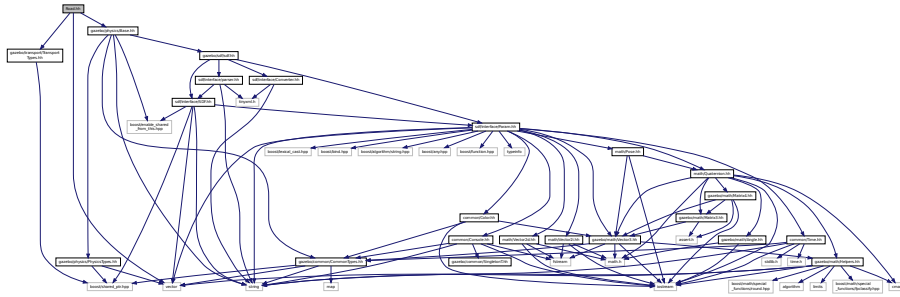
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

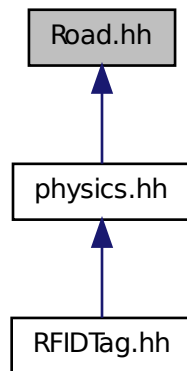
11.126 Road.hh File Reference

```
#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. 651) from SDF*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.127 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`

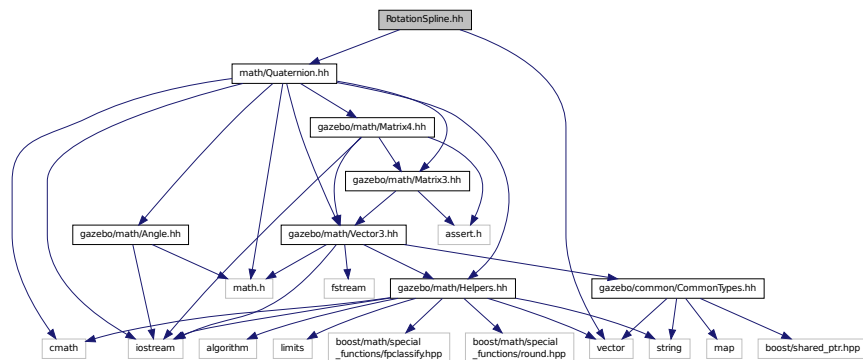
Namespaces

- namespace `gazebo`
Forward declarations for the common classes.
- namespace `gazebo::rendering`
Rendering namespace.

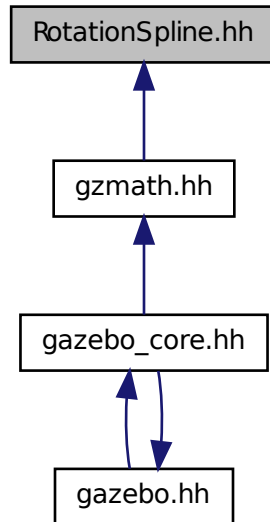
11.128 RotationSpline.hh File Reference

```
#include <vector>
#include "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. 740) for rotations.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

11.129 RTShaderSystem.hh File Reference

```
#include <list>
#include <string>
#include <vector>
#include "rendering/ogre_gazebo.h"
#include "gazebo_config.h"
#include "rendering/Camera.hh"
#include "common/SingletonT.hh"
```

Include dependency graph for RTShaderSystem.hh:



Classes

- class **gazebo::rendering::RTShaderSystem**

Implements *Ogre* (p. 106)'s Run-Time Shader system.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

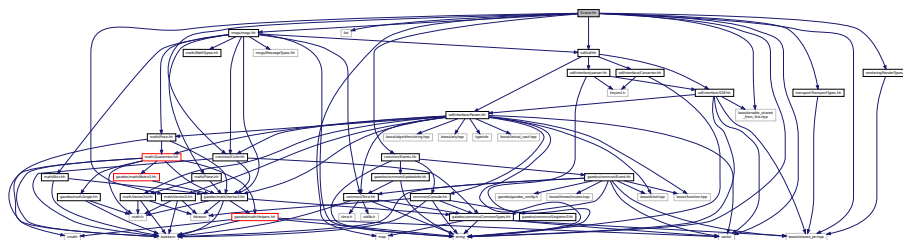
- namespace **gazebo::rendering**

Rendering namespace.

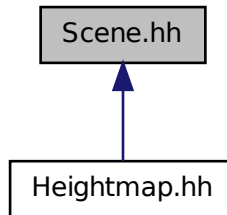
11.130 Scene.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include <list>
#include <boost/enable_shared_from_this.hpp>
#include <boost/shared_ptr.hpp>
#include "sdf/sdf.hh"
#include "msgs/msgs.hh"
#include "rendering/RenderTypes.hh"
#include "transport/TransportTypes.hh"
#include "common/Events.hh"
#include "common/Color.hh"
#include "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.

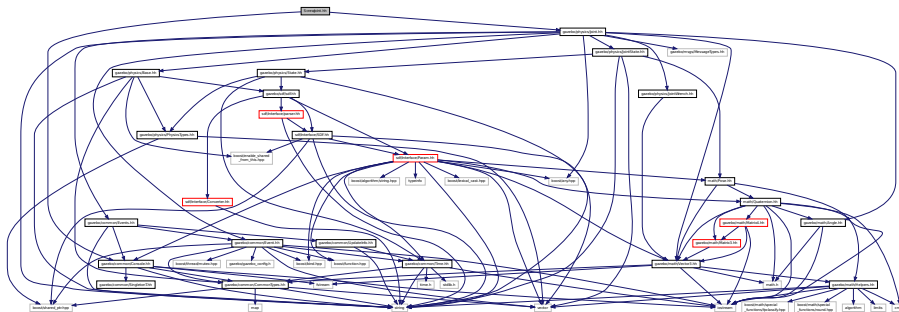
Namespaces

- namespace **boost**
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**
- namespace **SkyX**

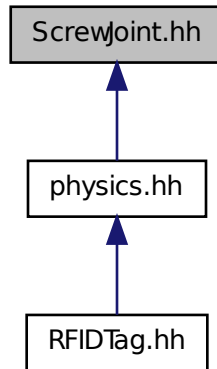
11.131 ScrewJoint.hh File Reference

```
#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 (p. 109) screw joint, which has both prismatic and rotational DOFs.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

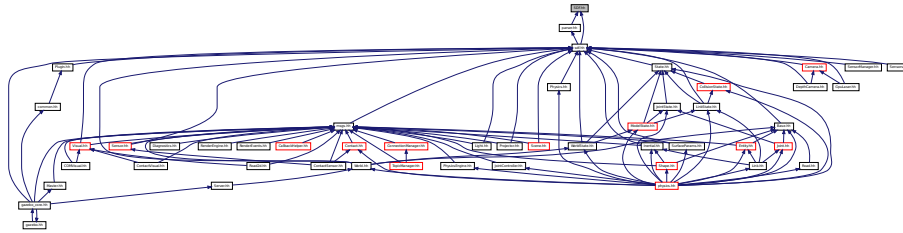
- namespace **gazebo::physics**

namespace for physics

11.132 sdf.hh File Reference

```
#include "sdf/interface/SDF.hh"  
#include "sdf/interface/Param.hh"  
#include "sdf/interface/parser.hh"  
#include "sdf/interface/Converter.hh"
```


This graph shows which files directly or indirectly include this file:



Classes

- class **sdf::Element**
SDF (p. 680) *Element* (p. 269) class.
- class **sdf::SDF**
Base SDF (p. 680) class.

Namespaces

- namespace **sdf**
namespace for Simulation Description Format parser

Macros

- `#define SDF_VERSION "1.3"`

Typedefs

- typedef Element * **sdf::ElementPtr**
- typedef std::vector< ElementPtr > **sdf::ElementPtr_V**
- typedef SDF * **sdf::SDFPtr**

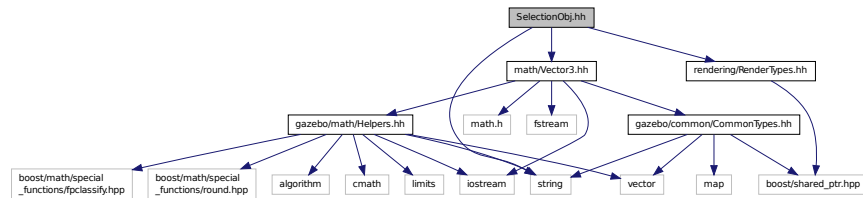
11.133.1 Macro Definition Documentation

11.133.1.1 `#define SDF_VERSION "1.3"`

11.134 SelectionObj.hh File Reference

```
#include <string>
#include "math/Vector3.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for SelectionObj.hh:



Classes

- class **gazebo::rendering::SelectionObj**
A (p. 109) graphical selection object.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

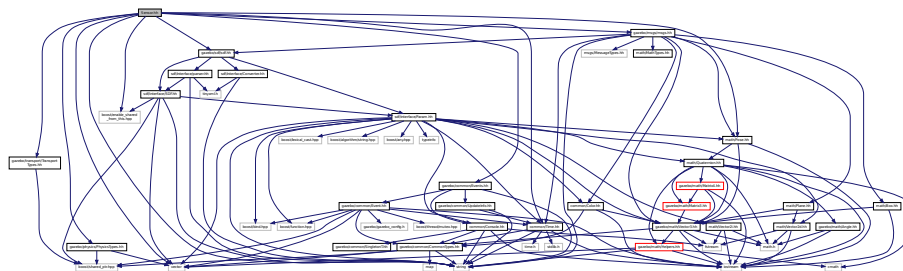
11.135 Sensor.hh File Reference

```

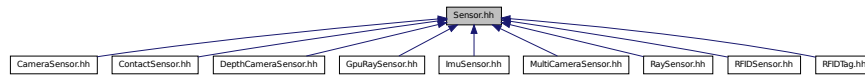
#include <boost/enable_shared_from_this.hpp>
#include <vector>
#include <string>
#include "gazebo/sdf/sdf.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/messages/messages.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/transport/TransportTypes.hh"

```

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.

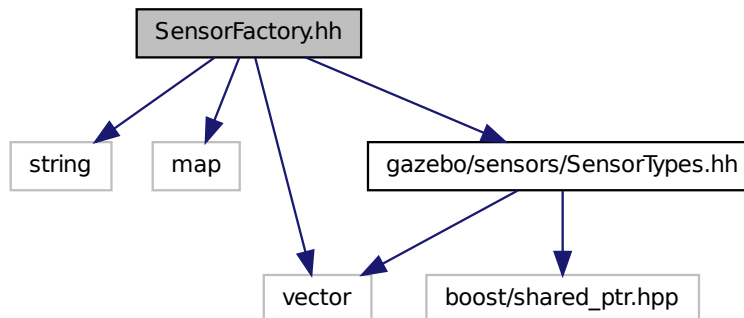
11.136 SensorFactory.hh File Reference

```

#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

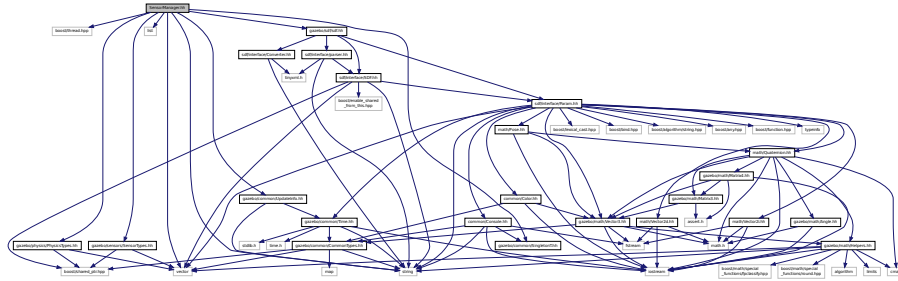
- typedef Sensor *(* **gazebo::sensors::SensorFactoryFn**)()

11.137 SensorManager.hh File Reference

```
#include <boost/thread.hpp>
```

```
#include <string>
#include <vector>
#include <list>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/sensors/SensorTypes.hh"
#include "gazebo/sdf/sdf.hh"
```

Include dependency graph for SensorManager.hh:



Classes

- class **gazebo::sensors::SensorManager**

Class to manage and update all sensors.

- class **gazebo::sensors::SimTimeEvent**

- class **gazebo::sensors::SimTimeEventHandler**

Monitors simulation time, and notifies conditions when a specified time has been reached.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

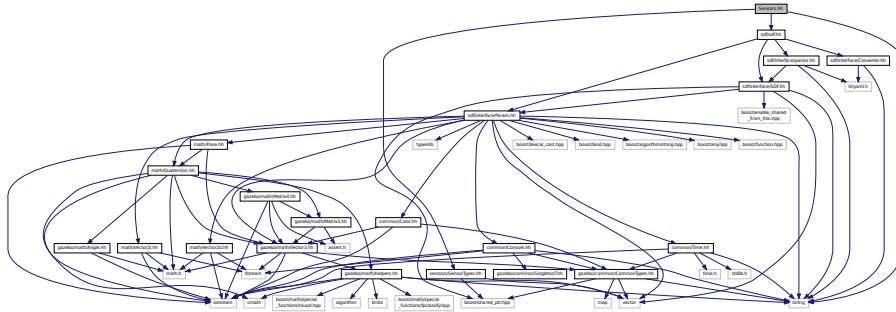
- namespace **gazebo::sensors**

Sensors namespace.

11.138 Sensors.hh File Reference

```
#include <string>
#include "sdf/sdf.hh"
#include "sensors/SensorTypes.hh"
```


Include dependency graph for Sensors.hh:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

Functions

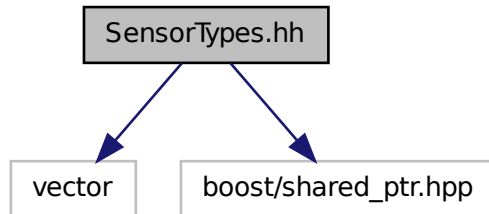
- std::string **gazebo::sensors::create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName)
Create a sensor using SDF.
- bool **gazebo::sensors::fini** ()
shutdown the sensor generation loop.
- SensorPtr **gazebo::sensors::get_sensor** (const std::string &_name)
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** () **GAZEBO_DEPRECATED**
Deprecated.
- 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.

11.139 SensorTypes.hh File Reference

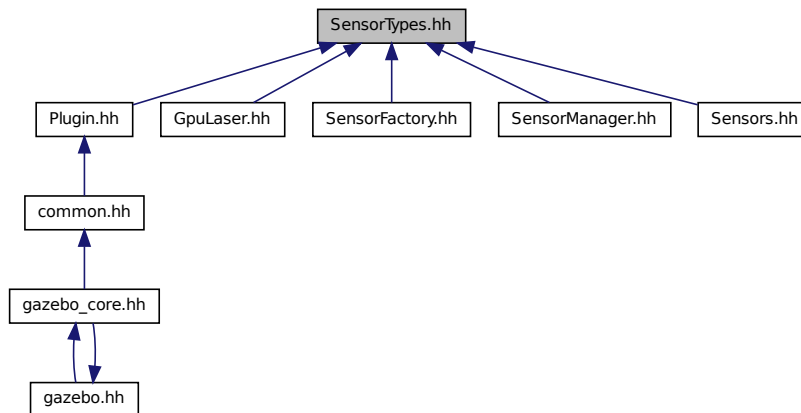
Forward declarations and typedefs for sensors.

```
#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 CameraSensor * **gazebo::sensors::CameraSensorPtr**
- typedef std::vector
< ContactSensorPtr > **gazebo::sensors::ContactSensor_V**
- typedef ContactSensor * **gazebo::sensors::ContactSensorPtr**
- typedef std::vector
< DepthCameraSensorPtr > **gazebo::sensors::DepthCameraSensor_V**
- typedef DepthCameraSensor * **gazebo::sensors::DepthCameraSensorPtr**
- typedef std::vector
< GpuRaySensorPtr > **gazebo::sensors::GpuRaySensor_V**
- typedef GpuRaySensor * **gazebo::sensors::GpuRaySensorPtr**
- typedef std::vector< ImuSensorPtr > **gazebo::sensors::ImuSensor_V**
- typedef ImuSensor * **gazebo::sensors::ImuSensorPtr**
- typedef std::vector< RaySensorPtr > **gazebo::sensors::RaySensor_V**
- typedef RaySensor * **gazebo::sensors::RaySensorPtr**
- typedef std::vector< RFIDSensor > **gazebo::sensors::RFIDSensor_V**
- typedef RFIDSensor * **gazebo::sensors::RFIDSensorPtr**
- typedef std::vector< RFIDTag > **gazebo::sensors::RFIDTag_V**
- typedef RFIDTag * **gazebo::sensors::RFIDTagPtr**
- typedef std::vector< SensorPtr > **gazebo::sensors::Sensor_V**
- typedef Sensor * **gazebo::sensors::SensorPtr**

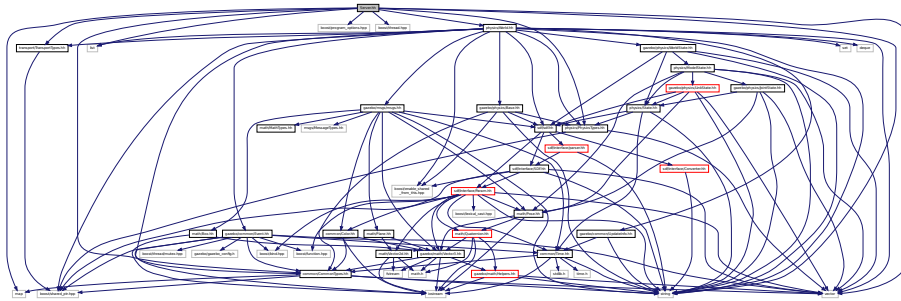
11.139.1 Detailed Description

Forward declarations and typedefs for sensors.

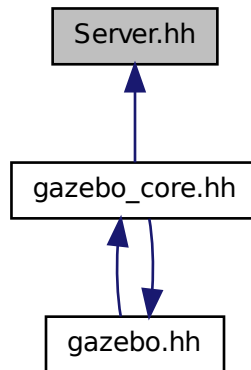
11.140 Server.hh File Reference

```
#include <string>
#include <vector>
#include <list>
#include <map>
#include <boost/program_options.hpp>
#include <boost/thread.hpp>
#include "transport/TransportTypes.hh"
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "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**

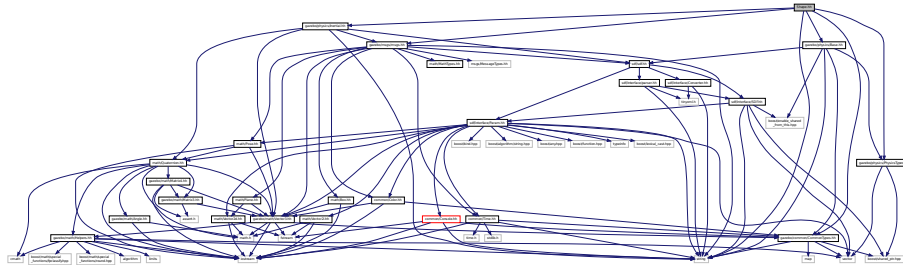
Forward declarations for the common classes.

11.141 Shape.hh File Reference

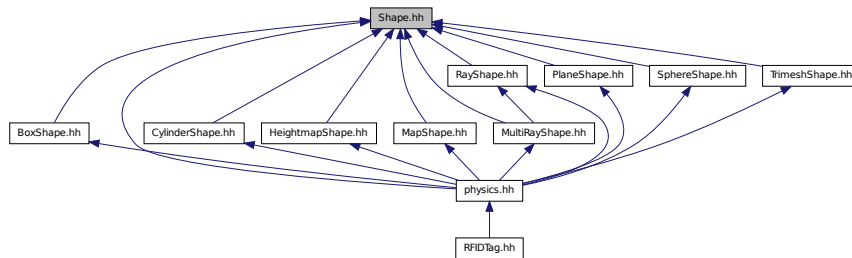
```
#include <string>
```

```
#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

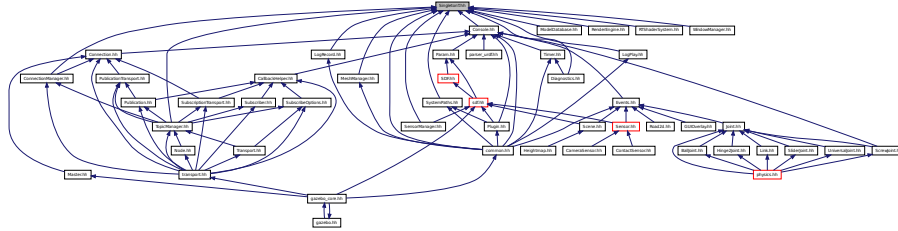
- class **gazebo::physics::Shape**
Base (p. 135) class for all shapes.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.142 SingletonT.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

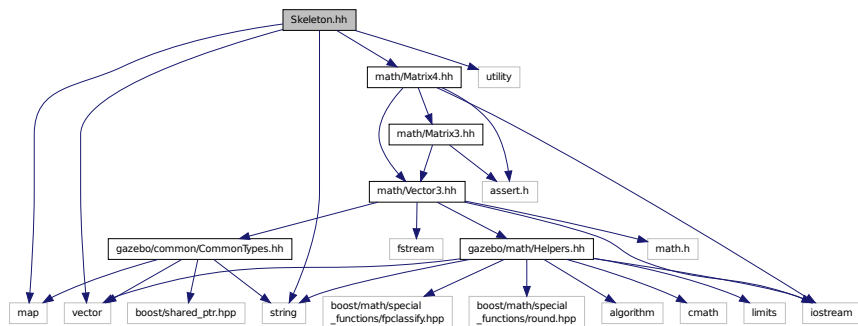
- class `SingletonT< T >`

Singleton template class.

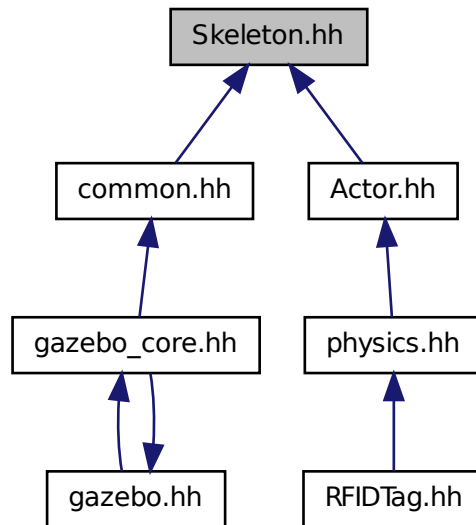
11.143 Skeleton.hh File Reference

```
#include <vector>
#include <string>
#include <map>
#include <utility>
#include "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. 537) *Skeleton.hh* (p. 1070) *common/common.hh*
- class **gazebo::common::Skeleton**
A (p. 109) *skeleton*.
- class **gazebo::common::SkeletonNode**
A (p. 109) *skeleton node*.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

Typedefs

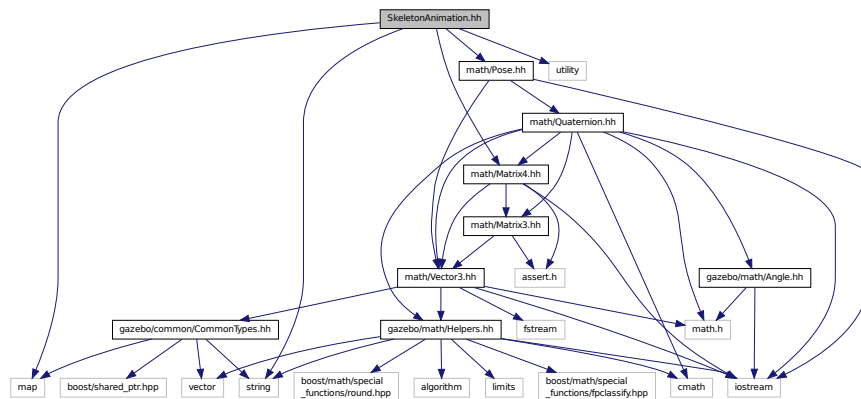
- typedef `std::map< unsigned int, SkeletonNode * >` **gazebo::common::NodeMap**
- typedef `std::map< unsigned int, SkeletonNode * >::iterator` **gazebo::common::NodeMapIter**

- 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**

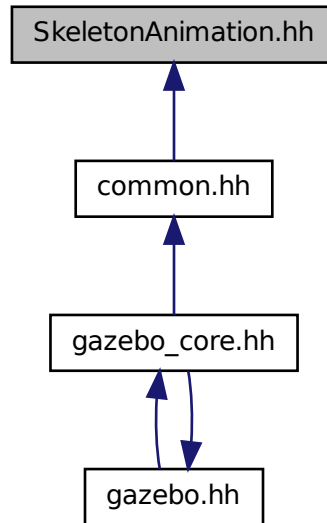
11.144 SkeletonAnimation.hh File Reference

```
#include <math/Matrix4.hh>
#include <math/Pose.hh>
#include <map>
#include <utility>
#include <string>
```

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. 713) animation.*

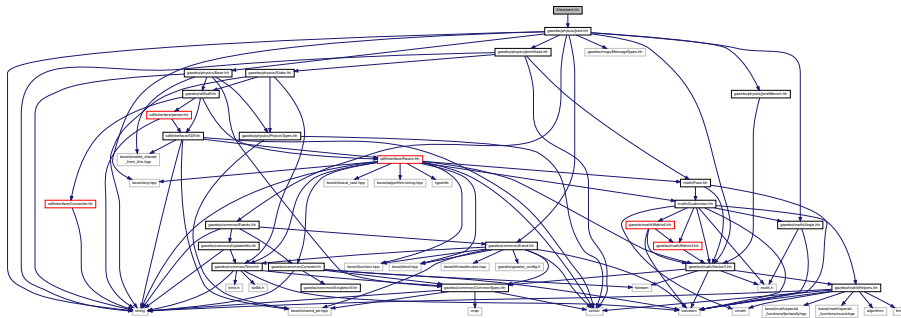
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

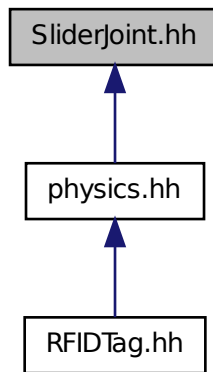
11.145 SliderJoint.hh File Reference

```
#include "gazebo/physics/Joint.hh"
```

Include dependency graph for SliderJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

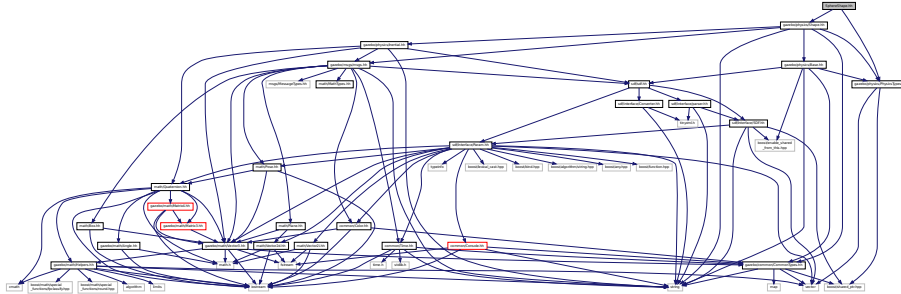
- class **gazebo::physics::SliderJoint**< T >
A (p. 109) slider joint.

Namespaces

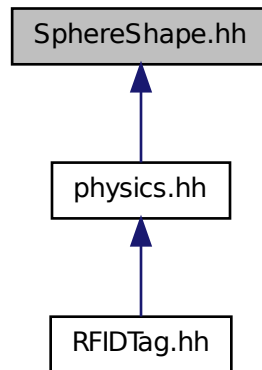
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.146 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

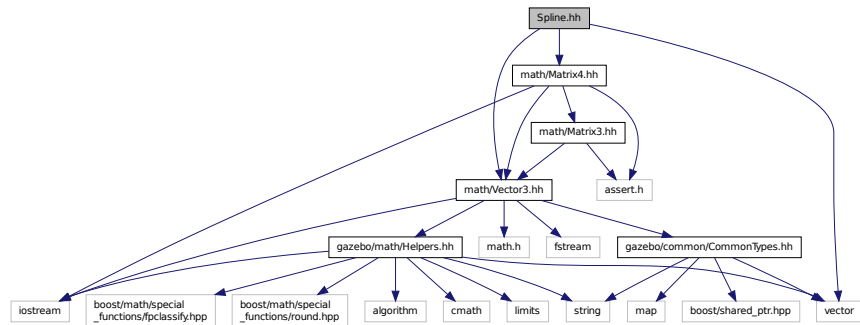
- class **gazebo::physics::SphereShape**
Sphere collision shape.

Namespaces

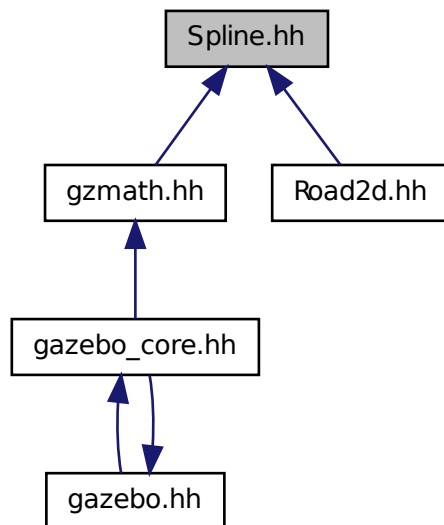
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.147 Spline.hh File Reference

```
#include <vector>
#include "math/Vector3.hh"
#include "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.

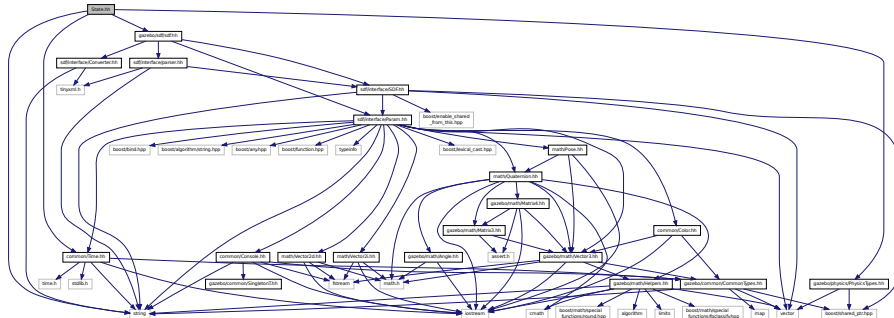
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

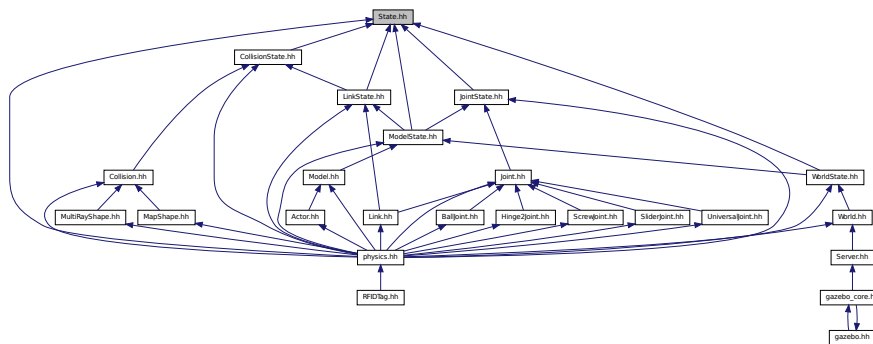
11.148 State.hh File Reference

```
#include <string>
#include "gazebo/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. 744) of an entity.

Namespaces

- namespace **gazebo**

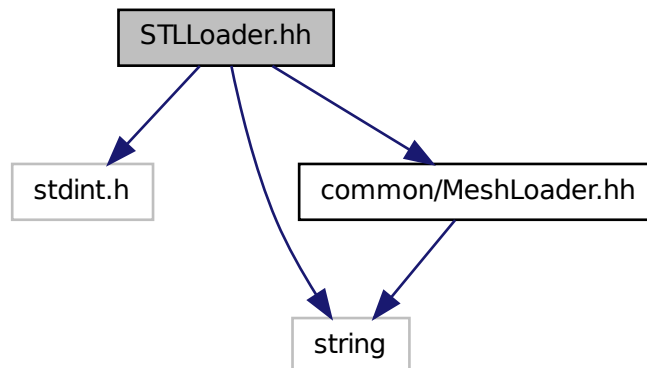
Forward declarations for the common classes.

- namespace **gazebo::physics**

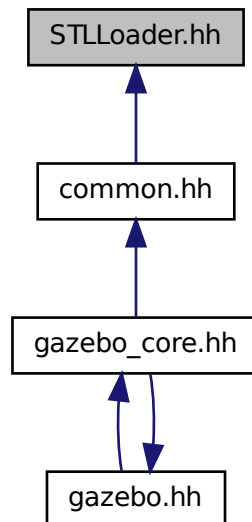
namespace for physics

11.149 STLLoader.hh File Reference

```
#include <stdint.h>
#include <string>
#include "common/MeshLoader.hh"
Include dependency graph for STLLoader.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::STLLoader**
Class used to load STL mesh files.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

Macros

- #define **COR3_MAX** 200000
- #define **FACE_MAX** 200000
- #define **LINE_MAX_LEN** 256
- #define **ORDER_MAX** 10

11.149.1 Macro Definition Documentation

11.149.1.1 #define COR3_MAX 200000

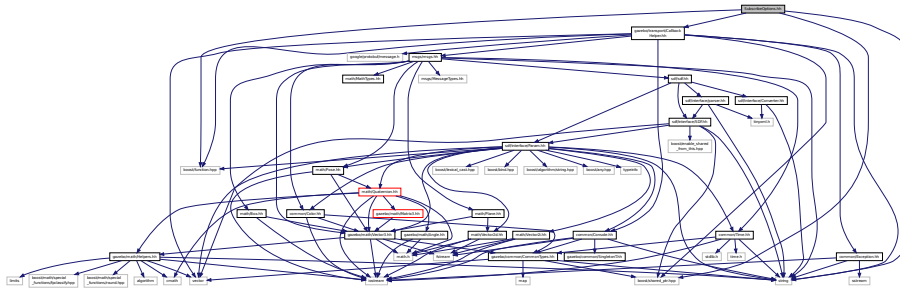
11.149.1.2 `#define FACE_MAX 200000`

11.149.1.3 `#define LINE_MAX_LEN 256`

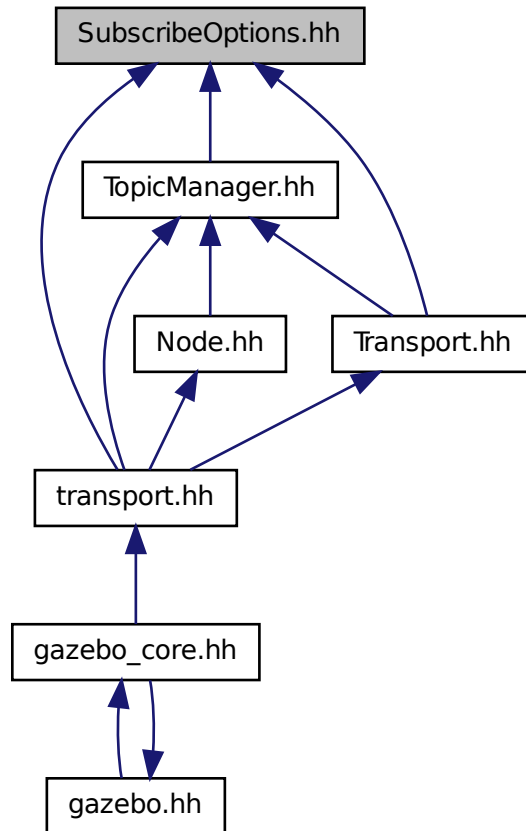
11.149.1.4 `#define ORDER_MAX 10`

11.150 SubscribeOptions.hh File Reference

```
#include <boost/function.hpp>
#include <boost/shared_ptr.hpp>
#include <string>
#include "gazebo/transport/CallbackHelper.hh"
Include dependency graph for SubscribeOptions.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::SubscribeOptions**
Options for a subscription.

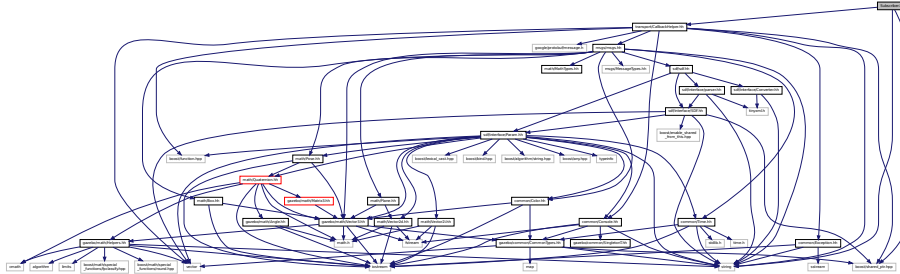
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

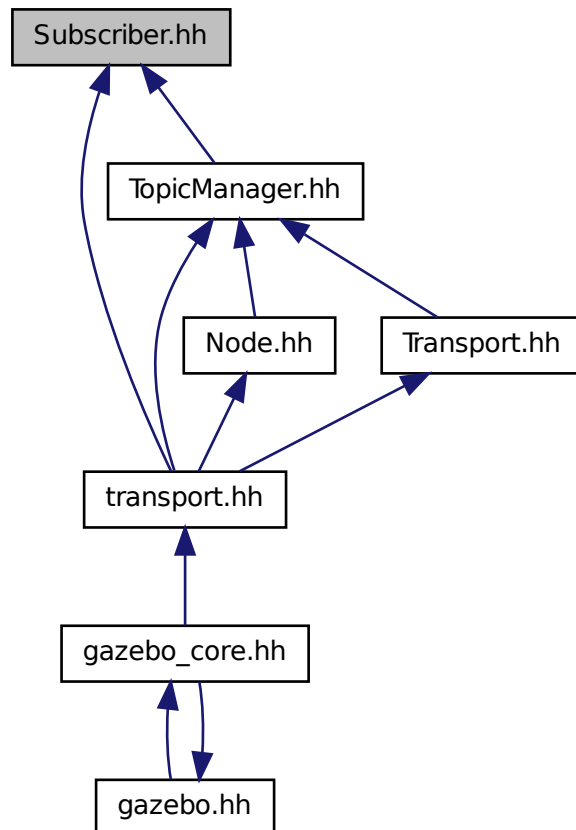
11.151 Subscriber.hh File Reference

```
#include <string>
```

```
#include <boost/shared_ptr.hpp>
#include "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 (p. 109) subscriber to a topic.

Namespaces

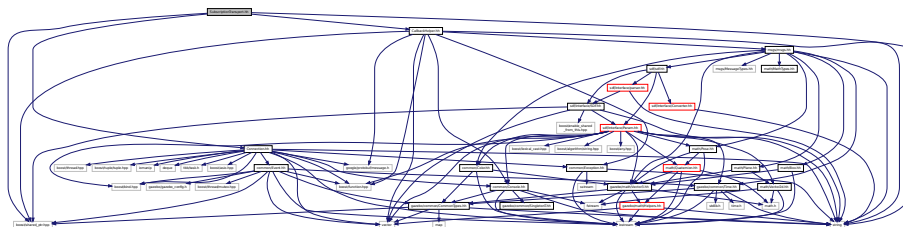
- namespace `gazebo`

Forward declarations for the common classes.

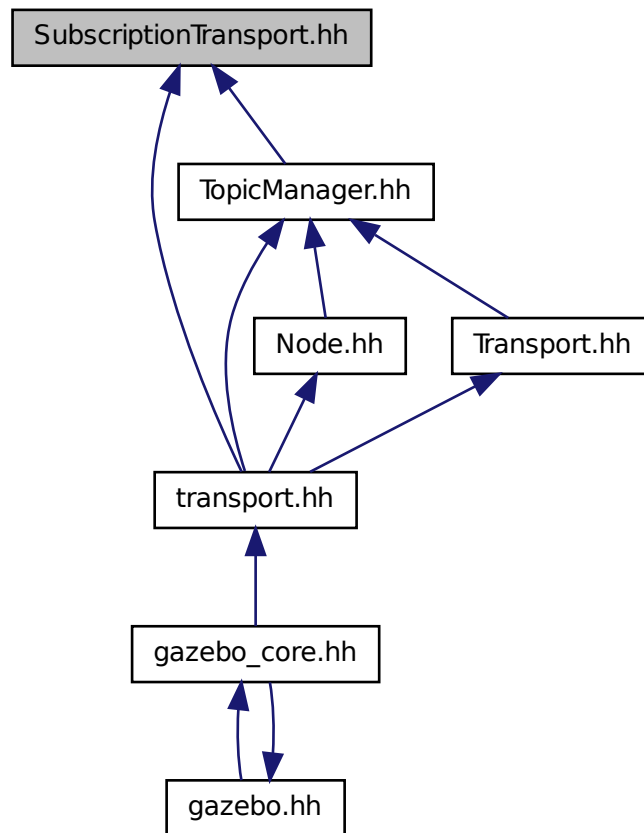
- namespace `gazebo::transport`

11.152 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

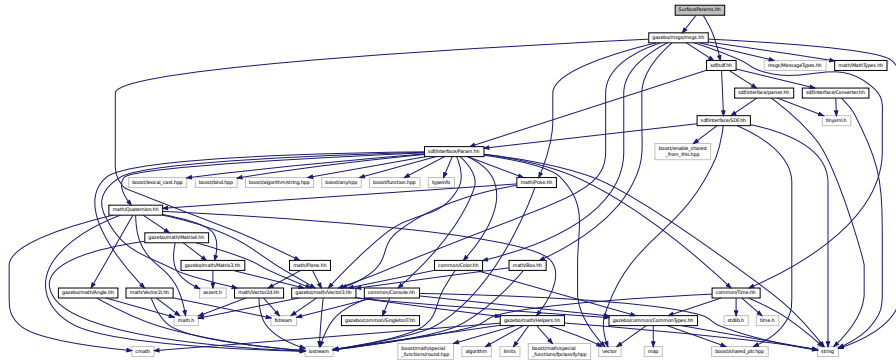
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

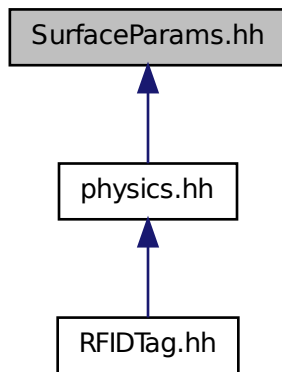
11.153 SurfaceParams.hh File Reference

```
#include "gazebo/msgs/msgs.hh"
```

```
#include "gazebo/sdf/sdf.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. 765) defines various Surface contact parameters.

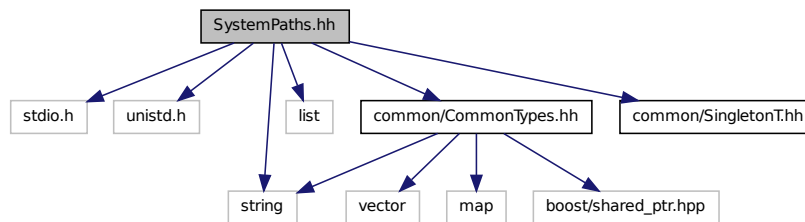
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

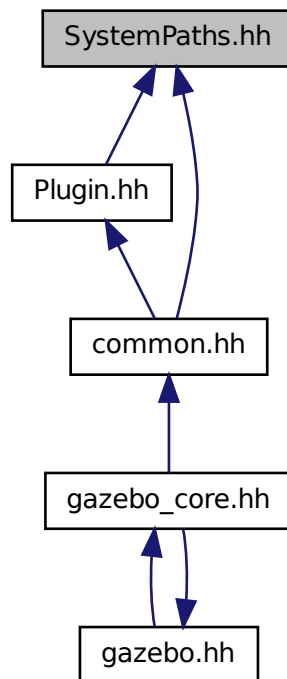
11.154 SystemPaths.hh File Reference

```
#include <stdio.h>
#include <unistd.h>
#include <string>
#include <list>
#include "common/CommonTypes.hh"
#include "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**

11.154.1 Macro Definition Documentation

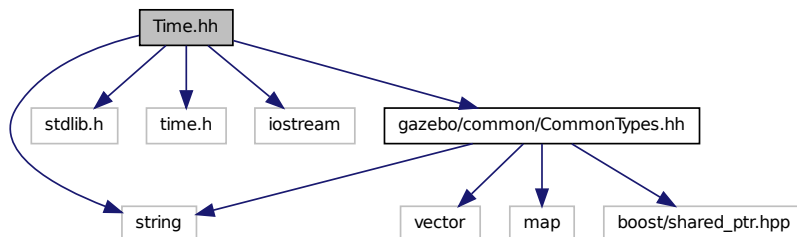
11.154.1.1 #define GetCurrentDir getcwd

11.154.1.2 #define LINUX

11.155 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 (p. 109) **Time** (p. 776) class, can be used to hold wall- or sim-time.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

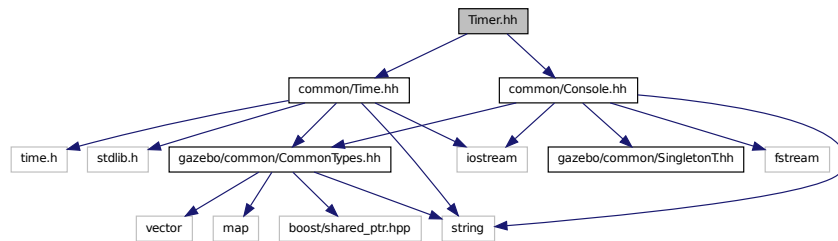
Common namespace.

11.156 Timer.hh File Reference

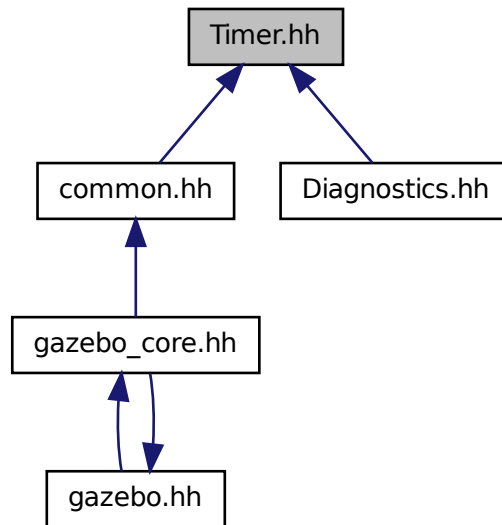
```
#include "common/Console.hh"
```

```
#include "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 (p. 109) timer class, used to time things in real world walltime.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

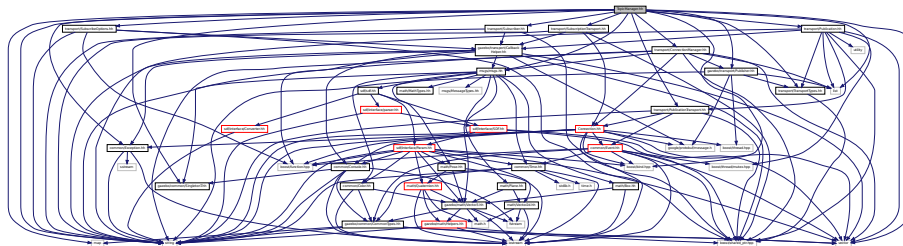
Common namespace.

11.157 TopicManager.hh File Reference

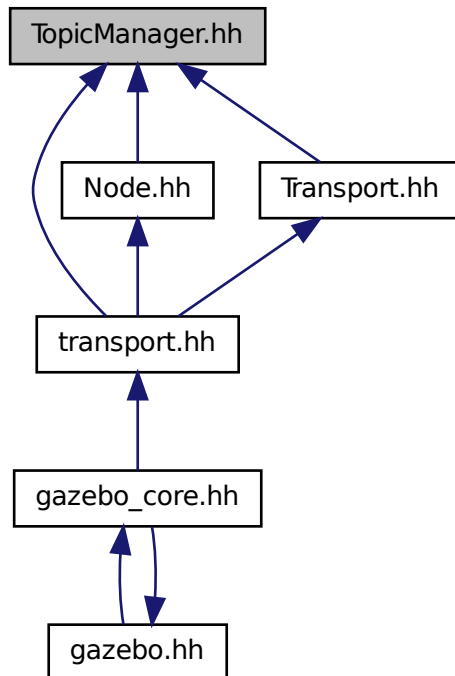
```
#include <boost/bind.hpp>
```

```
#include <map>
#include <list>
#include <string>
#include <vector>
#include "common/Exception.hh"
#include "msgs/msgs.hh"
#include "common/SingletonT.hh"
#include "transport/TransportTypes.hh"
#include "transport/SubscribeOptions.hh"
#include "transport/SubscriptionTransport.hh"
#include "transport/PublicationTransport.hh"
#include "transport/ConnectionManager.hh"
#include "transport/Publisher.hh"
#include "transport/Publication.hh"
#include "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.

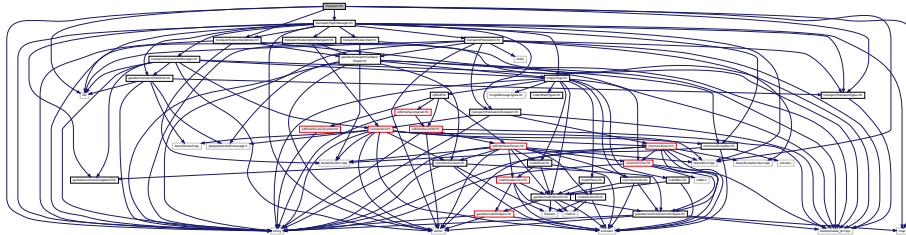
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

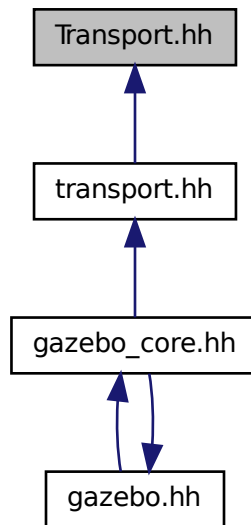
11.158 Transport.hh File Reference

```
#include <boost/bind.hpp>
```

```
#include <string>
#include <list>
#include <map>
#include "transport/TransportTypes.hh"
#include "transport/SubscribeOptions.hh"
#include "transport/TopicManager.hh"
Include dependency graph for Transport.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

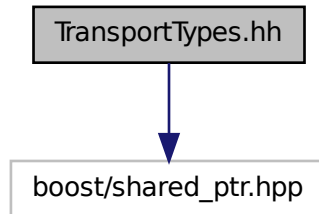
- void **gazebo::transport::clear_buffers** ()
Clear any remaining communication buffers.
- void **gazebo::transport::fini** ()
Cleanup the transport component.
- bool **gazebo::transport::get_master_uri** (std::string &_master_host, unsigned int &_master_port)
Get the hostname and port of the master 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.
- 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.
- 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::stop** ()
Stop the transport component from running.

11.159 TransportTypes.hh File Reference

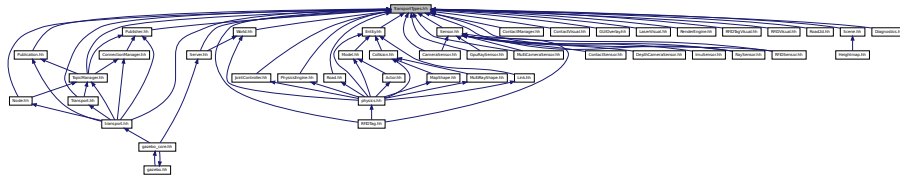
Forward declarations for transport.

```
#include <boost/shared_ptr.hpp>
```

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 Node * **gazebo::transport::NodePtr**
- typedef Publication * **gazebo::transport::PublicationPtr**
- typedef PublicationTransport * **gazebo::transport::PublicationTransportPtr**
- typedef Publisher * **gazebo::transport::PublisherPtr**
- typedef Subscriber * **gazebo::transport::SubscriberPtr**
- typedef SubscriptionTransport * **gazebo::transport::SubscriptionTransportPtr**

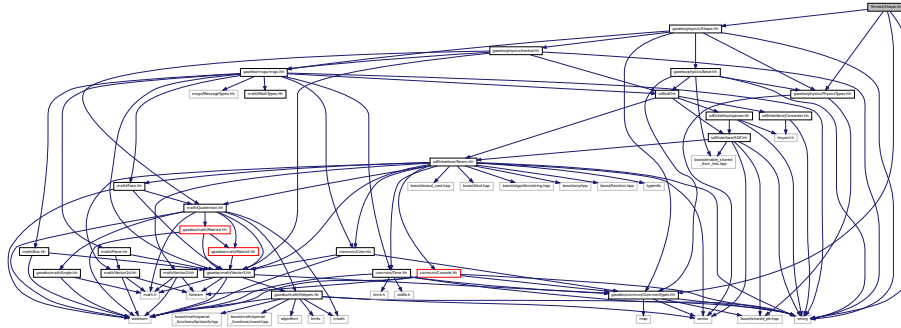
11.159.1 Detailed Description

Forward declarations for transport.

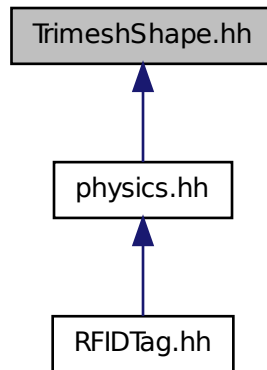
11.160 TrimeshShape.hh File Reference

```
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for TrimeshShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::TrimeshShape**
Triangle mesh collision shape.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

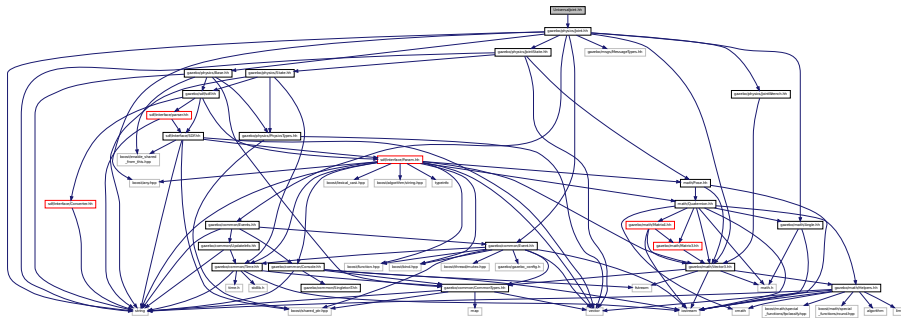
- namespace **gazebo::physics**

namespace for physics

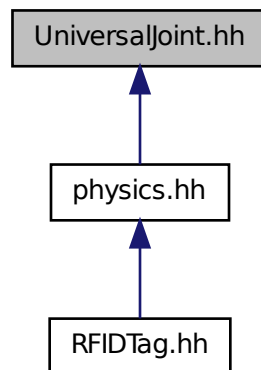
11.161 UniversalJoint.hh File Reference

```
#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 >

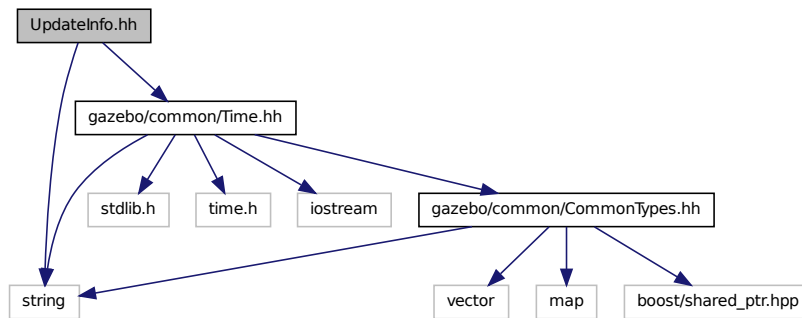
A (p. 109) *universal joint*.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.162 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

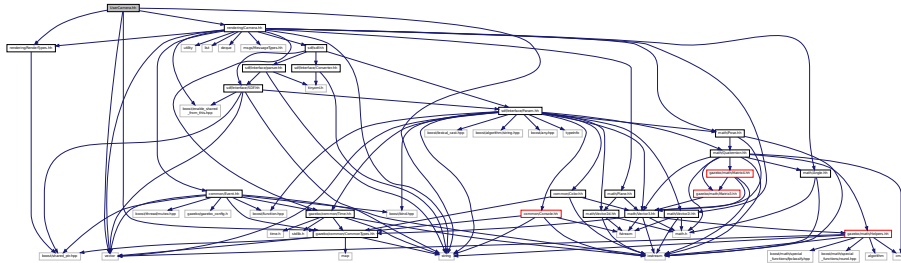
- class **gazebo::common::UpdateInfo**
Information for use in an update event.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.163 UserCamera.hh File Reference

```
#include <string>
#include <vector>
#include "rendering/Camera.hh"
#include "rendering/RenderTypes.hh"
#include "common/CommonTypes.hh"
Include dependency graph for UserCamera.hh:
```



Classes

- class **gazebo::rendering::UserCamera**

A (p. 109) camera used for user visualization of a scene.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

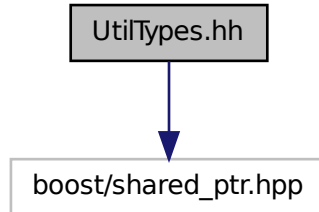
- namespace **gazebo::rendering**

Rendering namespace.

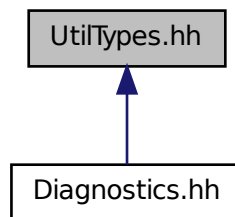
11.164 UtilTypes.hh File Reference

```
#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**

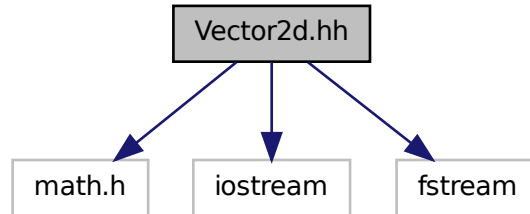
Typedefs

- typedef DiagnosticTimer * **gazebo::util::DiagnosticTimerPtr**

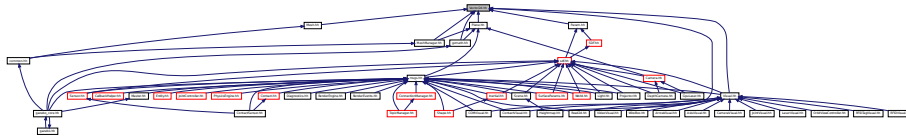
11.165 Vector2d.hh File Reference

```
#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**

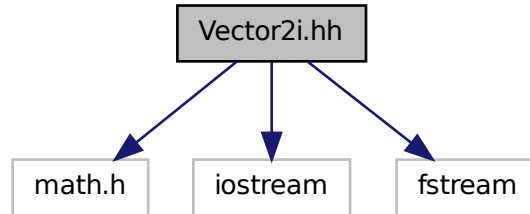
Math namespace.

11.166 Vector2i.hh File Reference

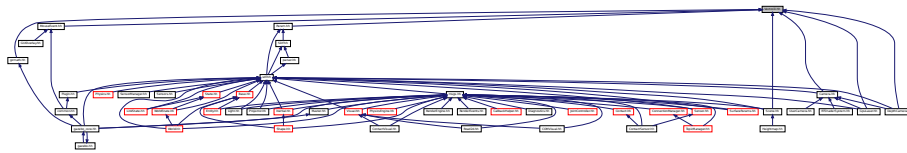
```

#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**

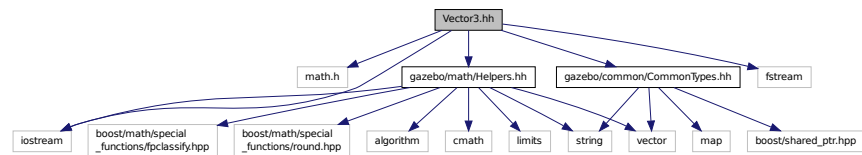
Math namespace.

11.167 Vector3.hh File Reference

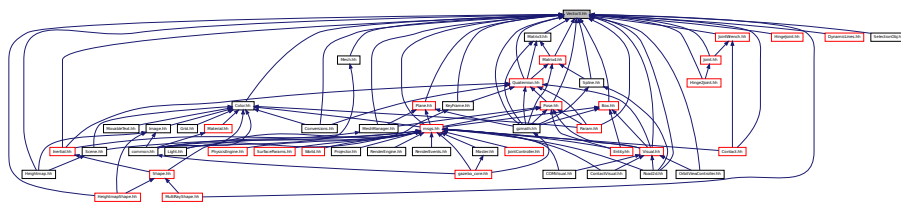
```

#include <math.h>
#include <iostream>
#include <fstream>
#include "gazebo/math/Helpers.hh"
#include "gazebo/common/CommonTypes.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. 839) class represents the generic vector containing 3 elements.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::math**

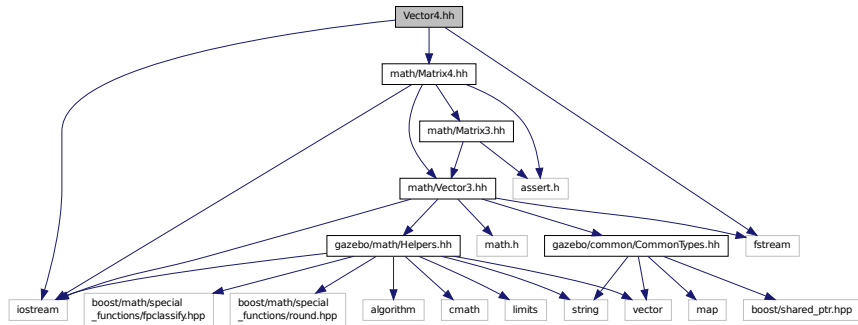
Math namespace.

11.168 Vector4.hh File Reference

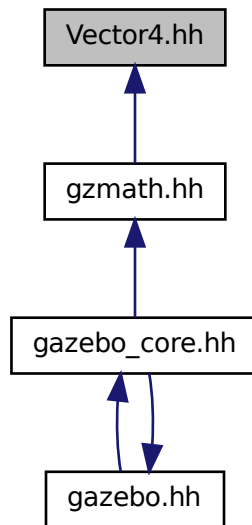
```

#include <iostream>
#include <fstream>
#include "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.

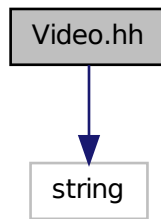
- namespace **gazebo::math**

Math namespace.

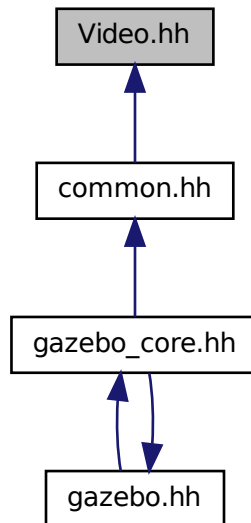
11.169 Video.hh File Reference

```
#include <string>
```

Include dependency graph for Video.hh:



This graph shows which files directly or indirectly include this file:



Classes

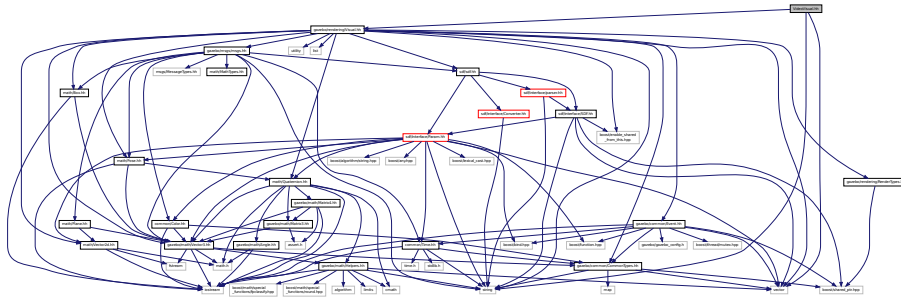
- class **gazebo::common::Video**
Handle video encoding and decoding using libavcodec.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.170 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 (p. 109) visual element that displays a video as a texture.

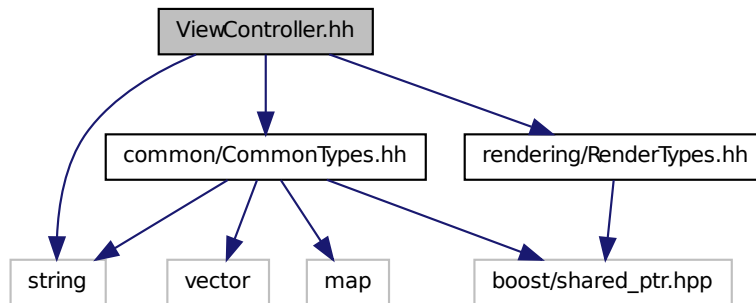
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::rendering**
Rendering namespace.

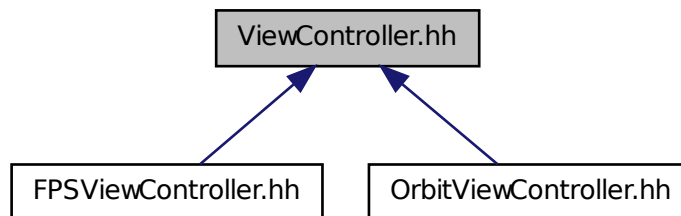
11.171 ViewController.hh File Reference

```
#include <string>
#include "common/CommonTypes.hh"
#include "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

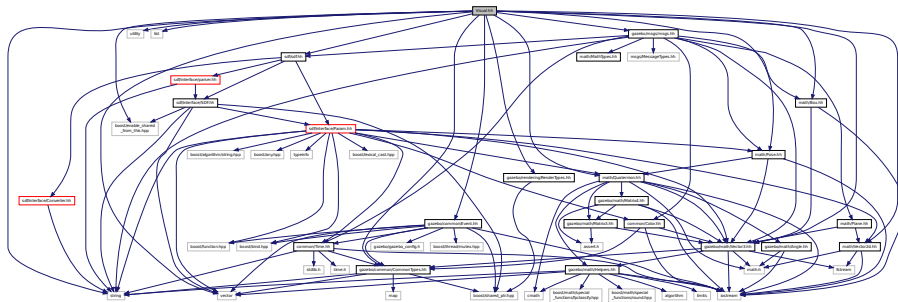
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**

Rendering namespace.

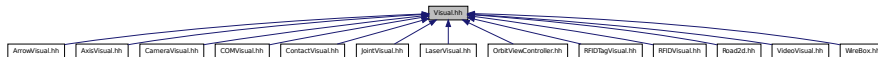
11.172 Visual.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include <utility>
#include <list>
#include <vector>
#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/sdf/sdf.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 (p. 109) *renderable object.*

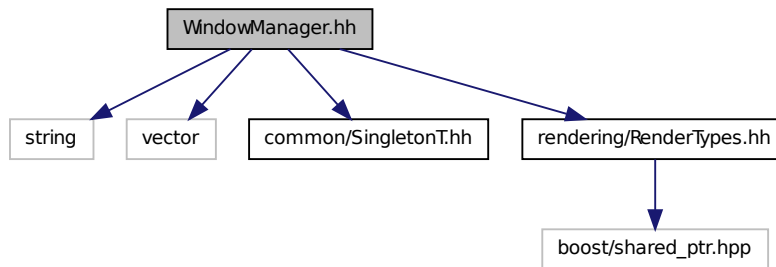
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.

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

11.173 WindowManager.hh File Reference

```
#include <string>
#include <vector>
#include "common/SingletonT.hh"
#include "rendering/RenderTypes.hh"
Include dependency graph for WindowManager.hh:
```



Classes

- class **gazebo::rendering::WindowManager**
Class to mangage render windows.

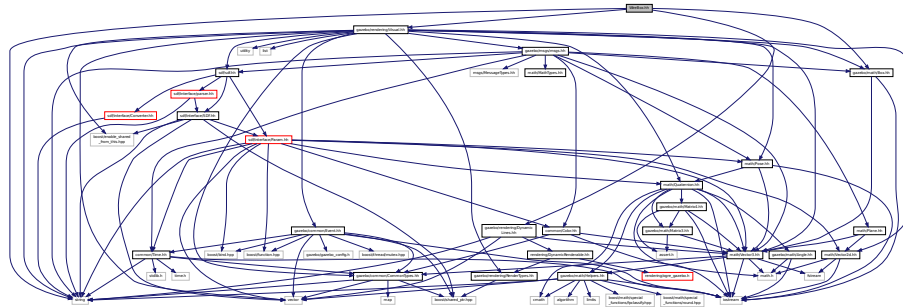
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.174 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.

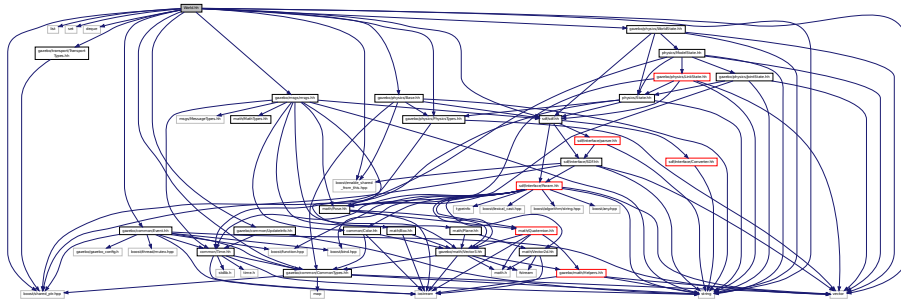
- namespace **gazebo::rendering**

Rendering namespace.

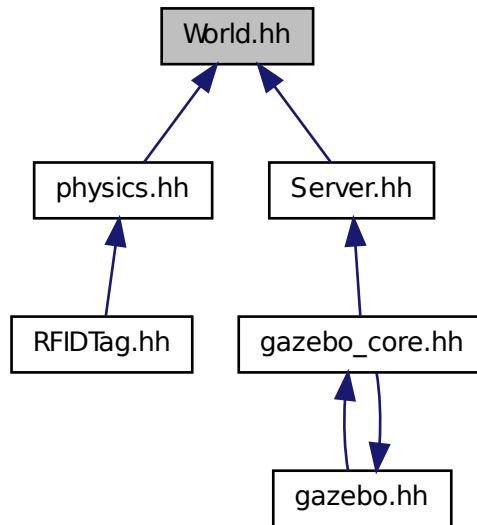
11.175 World.hh File Reference

```
#include <vector>
#include <list>
#include <set>
#include <deque>
#include <string>
#include <boost/enable_shared_from_this.hpp>
#include <boost/shared_ptr.hpp>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/physics/Base.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/WorldState.hh"
#include "gazebo/sdf/sdf.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 **boost**
- namespace **gazebo**

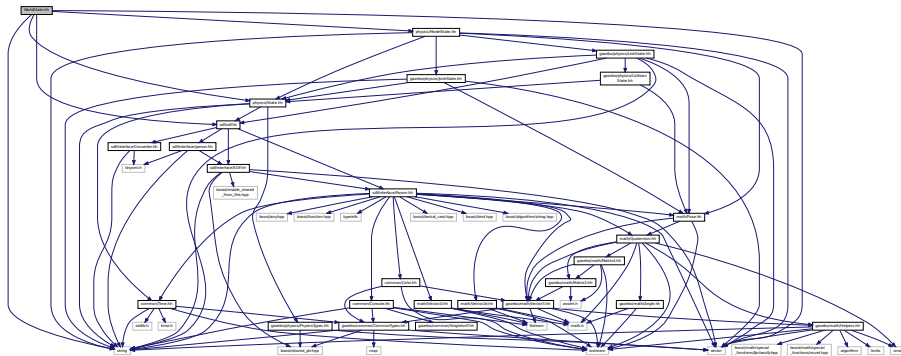
Forward declarations for the common classes.

- namespace **gazebo::physics**

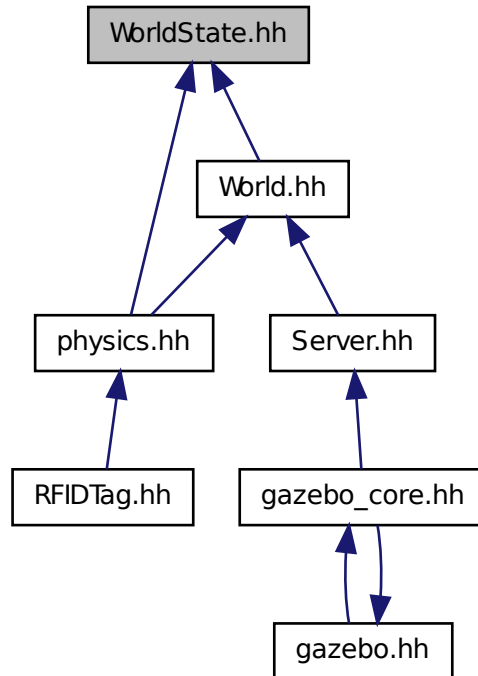
namespace for physics

11.176 WorldState.hh File Reference

```
#include <string>
#include <vector>
#include "sdf/sdf.hh"
#include "physics/State.hh"
#include "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. 893) object.*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Index

- ~Actor
 - gazebo::physics::Actor, 112
- ~Angle
 - gazebo::math::Angle, 118
- ~Animation
 - gazebo::common::Animation, 125
- ~ArrowVisual
 - gazebo::rendering::ArrowVisual, 129
- ~AssertionInternalError
 - gazebo::common::AssertionInternalError, 131
- ~AxisVisual
 - gazebo::rendering::AxisVisual, 132
- ~BVHLoader
 - gazebo::common::BVHLoader, 154
- ~BallJoint
 - gazebo::physics::BallJoint, 135
- ~Base
 - gazebo::physics::Base, 139
- ~Box
 - gazebo::math::Box, 148
- ~BoxShape
 - gazebo::physics::BoxShape, 153
- ~COMVisual
 - gazebo::rendering::COMVisual, 217
- ~CallbackHelper
 - gazebo::transport::CallbackHelper, 156
- ~Camera
 - gazebo::rendering::Camera, 166
- ~CameraSensor
 - gazebo::sensors::CameraSensor, 186
- ~CameraVisual
 - gazebo::rendering::CameraVisual, 190
- ~ColladaLoader
 - gazebo::common::ColladaLoader, 191
- ~Collision
 - gazebo::physics::Collision, 194
- ~CollisionState
 - gazebo::physics::CollisionState, 203
- ~Color
 - gazebo::common::Color, 208
- ~Connection
 - gazebo::event::Connection, 218
 - gazebo::transport::Connection, 221
- ~Contact
 - gazebo::physics::Contact, 233
- ~ContactManager
 - gazebo::physics::ContactManager, 236
- ~ContactSensor
 - gazebo::sensors::ContactSensor, 239
- ~ContactVisual
 - gazebo::rendering::ContactVisual, 243
- ~CylinderShape
 - gazebo::physics::CylinderShape, 248
- ~DepthCamera
 - gazebo::rendering::DepthCamera, 251
- ~DepthCameraSensor
 - gazebo::sensors::DepthCameraSensor, 255
- ~DiagnosticTimer
 - gazebo::util::DiagnosticTimer, 261
- ~DynamicLines
 - gazebo::rendering::DynamicLines, 263
- ~DynamicRenderable
 - gazebo::rendering::DynamicRenderable, 267
- ~Element
 - sdf::Element, 272
- ~Entity
 - gazebo::physics::Entity, 280
- ~Event
 - gazebo::event::Event, 290
- ~EventT
 - Events, 38
- ~Exception
 - gazebo::common::Exception, 315
- ~FPSViewController
 - gazebo::rendering::FPSViewController, 318
- ~GUIOverlay
 - gazebo::rendering::GUIOverlay, 342
- ~GazeboGenerator
 - google::protobuf::compiler::cpp::GazeboGenerator, 320
- ~GpuLaser
 - gazebo::rendering::GpuLaser, 322
- ~GpuRaySensor
 - gazebo::sensors::GpuRaySensor, 327
- ~Grid
 - gazebo::rendering::Grid, 337
- ~Gripper
 - gazebo::physics::Gripper, 340
- ~GzTerrainMatGen
 - gazebo::rendering::GzTerrainMatGen, 346

- ~Heightmap
 - gazebo::rendering::Heightmap, 347
- ~HeightmapShape
 - gazebo::physics::HeightmapShape, 350
- ~Hinge2Joint
 - gazebo::physics::Hinge2Joint, 353
- ~HingeJoint
 - gazebo::physics::HingeJoint, 355
- ~IOManager
 - gazebo::transport::IOManager, 374
- ~Image
 - gazebo::common::Image, 358
- ~ImuSensor
 - gazebo::sensors::ImuSensor, 363
- ~Inertial
 - gazebo::physics::Inertial, 367
- ~InternalError
 - gazebo::common::InternalError, 374
- ~Joint
 - gazebo::physics::Joint, 380
- ~JointState
 - gazebo::physics::JointState, 397
- ~JointVisual
 - gazebo::rendering::JointVisual, 400
- ~KeyFrame
 - gazebo::common::KeyFrame, 403
- ~LaserVisual
 - gazebo::rendering::LaserVisual, 405
- ~Light
 - gazebo::rendering::Light, 407
- ~Link
 - gazebo::physics::Link, 416
- ~LinkState
 - gazebo::physics::LinkState, 433
- ~Master
 - gazebo::Master, 445
- ~Material
 - gazebo::common::Material, 449
- ~Matrix3
 - gazebo::math::Matrix3, 457
- ~Matrix4
 - gazebo::math::Matrix4, 461
- ~Mesh
 - gazebo::common::Mesh, 468
- ~MeshCSG
 - gazebo::common::MeshCSG, 473
- ~MeshLoader
 - gazebo::common::MeshLoader, 474
- ~Model
 - gazebo::physics::Model, 483
- ~ModelPlugin
 - gazebo::ModelPlugin, 496
- ~ModelState
 - gazebo::physics::ModelState, 499
- ~MovableText
 - gazebo::rendering::MovableText, 508
- ~MultiCameraSensor
 - gazebo::sensors::MultiCameraSensor, 515
- ~MultiRayShape
 - gazebo::physics::MultiRayShape, 521
- ~Node
 - gazebo::transport::Node, 528
- ~NodeAnimation
 - gazebo::common::NodeAnimation, 533
- ~NodeTransform
 - gazebo::common::NodeTransform, 539
- ~NumericAnimation
 - gazebo::common::NumericAnimation, 543
- ~NumericKeyFrame
 - gazebo::common::NumericKeyFrame, 545
- ~OrbitViewController
 - gazebo::rendering::OrbitViewController, 547
- ~PID
 - gazebo::common::PID, 572
- ~Param
 - sdf::Param, 551
- ~ParamT
 - sdf::ParamT, 556
- ~PhysicsEngine
 - gazebo::physics::PhysicsEngine, 561
- ~Plane
 - gazebo::math::Plane, 576
- ~PlaneShape
 - gazebo::physics::PlaneShape, 579
- ~Pose
 - gazebo::math::Pose, 587
- ~PoseAnimation
 - gazebo::common::PoseAnimation, 594
- ~PoseKeyFrame
 - gazebo::common::PoseKeyFrame, 596
- ~Projector
 - gazebo::rendering::Projector, 598
- ~Publication
 - gazebo::transport::Publication, 601
- ~PublicationTransport
 - gazebo::transport::PublicationTransport, 605
- ~Publisher
 - gazebo::transport::Publisher, 607
- ~Quaternion
 - gazebo::math::Quaternion, 613
- ~RFIDSensor
 - gazebo::sensors::RFIDSensor, 645
- ~RFIDTag
 - gazebo::sensors::RFIDTag, 647
- ~RFIDTagVisual
 - gazebo::rendering::RFIDTagVisual, 650
- ~RFIDVisual
 - gazebo::rendering::RFIDVisual, 651

- ~RaySensor
 - gazebo::sensors::RaySensor, 629
- ~RayShape
 - gazebo::physics::RayShape, 636
- ~Road
 - gazebo::physics::Road, 652
- ~Road2d
 - gazebo::rendering::Road2d, 653
- ~RotationSpline
 - gazebo::math::RotationSpline, 655
- ~SM2Profile
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 736
- ~STLLoader
 - gazebo::common::STLLoader, 748
- ~Scene
 - gazebo::rendering::Scene, 665
- ~ScrewJoint
 - gazebo::physics::ScrewJoint, 678
- ~SelectionObj
 - gazebo::rendering::SelectionObj, 682
- ~Sensor
 - gazebo::sensors::Sensor, 686
- ~SensorPlugin
 - gazebo::SensorPlugin, 699
- ~Server
 - gazebo::Server, 700
- ~Shape
 - gazebo::physics::Shape, 707
- ~SimTimeEventHandler
 - gazebo::sensors::SimTimeEventHandler, 709
- ~SingletonT
 - SingletonT, 712
- ~Skeleton
 - gazebo::common::Skeleton, 714
- ~SkeletonAnimation
 - gazebo::common::SkeletonAnimation, 720
- ~SkeletonNode
 - gazebo::common::SkeletonNode, 726
- ~SliderJoint
 - gazebo::physics::SliderJoint, 733
- ~SphereShape
 - gazebo::physics::SphereShape, 738
- ~Spline
 - gazebo::math::Spline, 740
- ~State
 - gazebo::physics::State, 745
- ~SubMesh
 - gazebo::common::SubMesh, 751
- ~Subscriber
 - gazebo::transport::Subscriber, 761
- ~SubscriptionTransport
 - gazebo::transport::SubscriptionTransport, 763
- ~SurfaceParams
 - gazebo::physics::SurfaceParams, 765
- ~SystemPlugin
 - gazebo::SystemPlugin, 774
- ~Time
 - gazebo::common::Time, 779
- ~Timer
 - gazebo::common::Timer, 797
- ~TrimeshShape
 - gazebo::physics::TrimeshShape, 807
- ~URDF2Gazebo
 - urdf2gazebo::URDF2Gazebo, 812
- ~UniversalJoint
 - gazebo::physics::UniversalJoint, 810
- ~UserCamera
 - gazebo::rendering::UserCamera, 815
- ~Vector2d
 - gazebo::math::Vector2d, 823
- ~Vector2i
 - gazebo::math::Vector2i, 832
- ~Vector3
 - gazebo::math::Vector3, 841
- ~Vector4
 - gazebo::math::Vector4, 854
- ~Video
 - gazebo::common::Video, 862
- ~VideoVisual
 - gazebo::rendering::VideoVisual, 864
- ~ViewController
 - gazebo::rendering::ViewController, 865
- ~Visual
 - gazebo::rendering::Visual, 873
- ~WireBox
 - gazebo::rendering::WireBox, 892
- ~World
 - gazebo::physics::World, 895
- ~WorldPlugin
 - gazebo::WorldPlugin, 904
- ~WorldState
 - gazebo::physics::WorldState, 907
- _setupGeometry
 - gazebo::rendering::MovableText, 509
- _updateColors
 - gazebo::rendering::MovableText, 509
- A, 109
- a
 - gazebo::common::Color, 215
- ABGR
 - gazebo::common::Color, 207
- ACTOR
 - gazebo::physics::Base, 138
- ADD
 - gazebo::common::Material, 449
- ARGB

- gazebo::common::Color, 207
- AcceptCallback
 - gazebo::transport::Connection, 221
- active
 - gazebo::physics::Actor, 114
 - gazebo::sensors::Sensor, 692
- Actor
 - gazebo::physics::Actor, 112
- Actor.hh, 911
- Actor_V
 - gazebo::physics, 95
- ActorPtr
 - gazebo::physics, 95
- Add
 - gazebo::common::LogRecord, 441
- add_plugin
 - gazebo, 83
- add_search_path_suffix
 - Common, 32
- AddAnimation
 - gazebo::common::Skeleton, 714
- AddAttribute
 - sdf::Element, 272
- AddCallback
 - gazebo::transport::PublicationTransport, 605
- AddChild
 - gazebo::common::SkeletonNode, 726
 - gazebo::physics::Base, 139
- AddChildJoint
 - gazebo::physics::Link, 416
- AddContact
 - gazebo::physics::Collision, 195
- AddElement
 - sdf::Element, 272
- AddElementDescription
 - sdf::Element, 272
- addEntity
 - gazebo::event::Events, 303
- AddForce
 - gazebo::physics::Link, 416
- AddForceAtRelativePosition
 - gazebo::physics::Link, 417
- AddForceAtWorldPosition
 - gazebo::physics::Link, 417
- AddGazeboPaths
 - gazebo::common::SystemPaths, 770
- AddIndex
 - gazebo::common::SubMesh, 751
- AddJoint
 - gazebo::physics::JointController, 394
- AddKeyFrame
 - gazebo::common::NodeAnimation, 533, 534
 - gazebo::common::SkeletonAnimation, 720
- AddMaterial
 - gazebo::common::Mesh, 468
- AddMesh
 - gazebo::common::MeshManager, 476
- AddModelPaths
 - gazebo::common::SystemPaths, 770
- addNestedModel
 - sdf, 107
- AddNode
 - gazebo::transport::TopicManager, 800
- AddNodeAssignment
 - gazebo::common::SubMesh, 752
- AddNormal
 - gazebo::common::SubMesh, 752
- AddOgrePaths
 - gazebo::common::SystemPaths, 770
- AddParentJoint
 - gazebo::physics::Link, 417
- AddPluginPaths
 - gazebo::common::SystemPaths, 770
- AddPoint
 - gazebo::math::RotationSpline, 655
 - gazebo::math::Spline, 740
 - gazebo::rendering::DynamicLines, 264
- AddPublisher
 - gazebo::transport::Publication, 601
- AddRawTransform
 - gazebo::common::SkeletonNode, 726
- AddRay
 - gazebo::physics::MultiRayShape, 521
- AddRelativeEvent
 - gazebo::sensors::SimTimeEventHandler, 710
- AddRelativeForce
 - gazebo::physics::Link, 417
- AddRelativeTorque
 - gazebo::physics::Link, 417
- AddResourcePath
 - gazebo::rendering::RenderEngine, 641
- AddScene
 - gazebo::rendering::RTShaderSystem, 659
- AddSearchPathSuffix
 - gazebo::common::SystemPaths, 770
- AddSubMesh
 - gazebo::common::Mesh, 468
- AddSubscription
 - gazebo::transport::Publication, 601
- AddTag
 - gazebo::sensors::RFIDSensor, 645
- addTechnique
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 736
- AddTexCoord
 - gazebo::common::SubMesh, 752
- AddTime
 - gazebo::common::Animation, 125

- AddTorque
 - gazebo::physics::Link, 418
- AddTransport
 - gazebo::transport::Publication, 601
- AddType
 - gazebo::physics::Base, 139
- AddValue
 - sdf::Element, 272
- AddVertNodeWeight
 - gazebo::common::Skeleton, 714
- AddVertex
 - gazebo::common::SubMesh, 752
- AddVisual
 - gazebo::rendering::Scene, 665
- Advertise
 - gazebo::transport::ConnectionManager, 227
 - gazebo::transport::Node, 528
 - gazebo::transport::TopicManager, 801
- alt
 - gazebo::common::MouseEvent, 505
- ambient
 - gazebo::common::Material, 454
- anchorLink
 - gazebo::physics::Joint, 391
- anchorPos
 - gazebo::physics::Joint, 391
- anchorPose
 - gazebo::physics::Joint, 392
- Angle
 - gazebo::math::Angle, 117
- Angle.hh, 912
 - GZ_DTOR, 914
 - GZ_NORMALIZE, 914
 - GZ_RTOD, 914
- angularAccel
 - gazebo::physics::Link, 430
- animState
 - gazebo::rendering::Camera, 182
- Animation
 - gazebo::common::Animation, 125
- animation
 - gazebo::physics::Entity, 287
- Animation.hh, 915
- AnimationComplete
 - gazebo::rendering::Camera, 166
 - gazebo::rendering::UserCamera, 815
- animationConnection
 - gazebo::physics::Entity, 287
- AnimationPtr
 - gazebo::common, 86
- animationStartPose
 - gazebo::physics::Entity, 287
- animations
 - gazebo::common::SkeletonAnimation, 722
- anims
 - gazebo::common::Skeleton, 718
- ApplyDamping
 - gazebo::physics::Joint, 380
- applyDamping
 - gazebo::physics::Joint, 392
- ApplyShadows
 - gazebo::rendering::RTShaderSystem, 660
- AreConnected
 - gazebo::physics::Joint, 380
- ArrowVisual
 - gazebo::rendering::ArrowVisual, 128
- ArrowVisual.hh, 916
- ArrowVisualPtr
 - gazebo::rendering, 99
- Assert.hh, 917
 - GZ_ASSERT, 918
- AssertionInternalError
 - gazebo::common::AssertionInternalError, 130
- AsyncRead
 - gazebo::transport::Connection, 221
- Attach
 - gazebo::physics::Joint, 380
 - gazebo::rendering::SelectionObj, 682
- AttachAxes
 - gazebo::rendering::Visual, 873
- AttachCameraToImage
 - gazebo::rendering::GUIOverlay, 342
- AttachEntity
 - gazebo::rendering::RTShaderSystem, 660
- AttachLineVertex
 - gazebo::rendering::Visual, 873
- AttachMesh
 - gazebo::rendering::Visual, 873
- AttachObject
 - gazebo::rendering::Visual, 873
- AttachStaticModel
 - gazebo::physics::Link, 418
 - gazebo::physics::Model, 483
- AttachToVisual
 - gazebo::rendering::Camera, 166
- AttachToVisualImpl
 - gazebo::rendering::Camera, 166, 167
 - gazebo::rendering::UserCamera, 815
- AttachViewport
 - gazebo::rendering::RTShaderSystem, 660
- AttachVisual
 - gazebo::rendering::Visual, 873
- attachedModels
 - gazebo::physics::Model, 493
- attachedModelsOffset
 - gazebo::physics::Link, 430
 - gazebo::physics::Model, 493
- Attribute

- gazebo::physics::Joint, 379
- autoCalc
 - gazebo::math::RotationSpline, 657
 - gazebo::math::Spline, 743
- autoStart
 - gazebo::physics::Actor, 114
- AxisVisual
 - gazebo::rendering::AxisVisual, 132
- AxisVisual.hh, 918
- AxisVisualPtr
 - gazebo::rendering, 99
- b
 - gazebo::common::Color, 215
- BALL_JOINT
 - gazebo::physics::Base, 139
- BASE
 - gazebo::physics::Base, 138
- BAYER_GBRG8
 - gazebo::common::Image, 358
- BAYER_GRBG8
 - gazebo::common::Image, 358
- BAYER_RGGB8
 - gazebo::common::Image, 358
- BAYER_RGGR8
 - gazebo::common::Image, 358
- BGR_INT16
 - gazebo::common::Image, 357
- BGR_INT32
 - gazebo::common::Image, 357
- BGR_INT8
 - gazebo::common::Image, 357
- BGRA
 - gazebo::common::Color, 207
- BGRA_INT8
 - gazebo::common::Image, 357
- BLEND_COUNT
 - gazebo::common::Material, 449
- BLINN
 - gazebo::common::Material, 449
- BOX_SHAPE
 - gazebo::physics::Base, 139
- BVHLoader
 - gazebo::common::BVHLoader, 154
- BVHLoader.hh, 923
 - X_POSITION, 924
 - X_ROTATION, 924
 - Y_POSITION, 924
 - Y_ROTATION, 924
 - Z_POSITION, 924
 - Z_ROTATION, 924
- BallJoint
 - gazebo::physics::BallJoint, 134
- BallJoint.hh, 919
- Base
 - gazebo::physics::Base, 139
- Base.hh, 920
- Base_V
 - gazebo::physics, 95
- BasePtr
 - gazebo::physics, 95
- bayerFrameBuffer
 - gazebo::rendering::Camera, 182
- bindShapeTransform
 - gazebo::common::Skeleton, 718
- Black
 - gazebo::common::Color, 215
- BlendMode
 - gazebo::common::Material, 449
- blendMode
 - gazebo::common::Material, 454
- BlendModeStr
 - gazebo::common::Material, 454
- Blue
 - gazebo::common::Color, 215
- body1Force
 - gazebo::physics::JointWrench, 401
- body1Torque
 - gazebo::physics::JointWrench, 401
- body2Force
 - gazebo::physics::JointWrench, 402
- body2Torque
 - gazebo::physics::JointWrench, 402
- bonePosePub
 - gazebo::physics::Actor, 114
- BooleanOperation
 - gazebo::common::MeshCSG, 473
- boost, 81
- bounce
 - gazebo::physics::SurfaceParams, 765
- bounceThreshold
 - gazebo::physics::SurfaceParams, 766
- Box
 - gazebo::math::Box, 148
- Box.hh, 921
- BoxShape
 - gazebo::physics::BoxShape, 153
- BoxShape.hh, 921
- BoxShapePtr
 - gazebo::physics, 95
- build
 - gazebo::common::Animation, 126
- BuildInterpolationSplines
 - gazebo::common::PoseAnimation, 594
- BuildNodeMap
 - gazebo::common::Skeleton, 715
- button
 - gazebo::common::MouseEvent, 505

- ButtonCallback
 - gazebo::rendering::GUIOverlay, 343
- Buttons
 - gazebo::common::MouseEvent, 504
- buttons
 - gazebo::common::MouseEvent, 505
- CATEGORY_COUNT
 - gazebo::sensors, 102
- CFM
 - gazebo::physics::Joint, 380
- COLLISION
 - gazebo::physics::Base, 138
- COMVisual
 - gazebo::rendering::COMVisual, 217
- COMVisual.hh, 937
- COMVisualPtr
 - gazebo::rendering, 99
- COR3_MAX
 - STLLoader.hh, 1077
- CYLINDER_SHAPE
 - gazebo::physics::Base, 139
- CallbackHelper
 - gazebo::transport::CallbackHelper, 156
- CallbackHelper.hh, 924
- CallbackHelperPtr
 - Transport, 75
- CallbackHelperT
 - gazebo::transport::CallbackHelperT, 158
- Camera
 - gazebo::rendering::Camera, 165
- camera
 - gazebo::rendering::Camera, 182
 - gazebo::rendering::ViewController, 867
- Camera.hh, 926
- cameraCount
 - gazebo::sensors::GpuRaySensor, 334
- cameraElem
 - gazebo::sensors::GpuRaySensor, 334
- CameraPtr
 - gazebo::rendering, 99
- CameraSensor
 - gazebo::sensors::CameraSensor, 186
- CameraSensor.hh, 927
- CameraSensor_V
 - gazebo::sensors, 102
- CameraSensorPtr
 - gazebo::sensors, 102
- CameraVisual
 - gazebo::rendering::CameraVisual, 190
- CameraVisual.hh, 928
- CameraVisualPtr
 - gazebo::rendering, 99
- Cancel
 - gazebo::transport::Connection, 221
- captureData
 - gazebo::rendering::Camera, 182
- cegui.h, 928
- cfm
 - gazebo::physics::SurfaceParams, 766
- cgVisuals
 - gazebo::physics::Link, 430
- chfov
 - gazebo::sensors::GpuRaySensor, 334
- childLink
 - gazebo::physics::Joint, 392
- children
 - gazebo::common::SkeletonNode, 731
 - gazebo::physics::Base, 146
- childrenEnd
 - gazebo::physics::Base, 146
- clamp
 - Math, 50
- Classes for physics and dynamics, 41
 - create_world, 44
 - EntityTypename, 46
 - fini, 44
 - GZ_REGISTER_PHYSICS_ENGINE, 44
 - get_world, 44
 - init_world, 45
 - init_worlds, 45
 - load, 45
 - load_world, 45
 - load_worlds, 45
 - pause_world, 45
 - pause_worlds, 46
 - PhysicsFactoryFn, 44
 - remove_worlds, 46
 - run_world, 46
 - run_worlds, 46
 - stop_world, 46
 - stop_worlds, 46
- Clear
 - gazebo::math::RotationSpline, 655
 - gazebo::math::Spline, 740
 - gazebo::physics::ContactManager, 236
 - gazebo::physics::World, 896
 - gazebo::rendering::DynamicLines, 264
 - gazebo::rendering::RTShaderSystem, 660
 - gazebo::rendering::Scene, 666
 - gazebo::rendering::SelectionObj, 682
 - sdf::Plugin, 582
- clear_buffers
 - Transport, 76
- ClearBuffers
 - gazebo::transport::TopicManager, 801
- ClearElements
 - sdf::Element, 272

- ClearGazeboPaths
 - gazebo::common::SystemPaths, 771
- ClearModelPaths
 - gazebo::common::SystemPaths, 771
- ClearOgrePaths
 - gazebo::common::SystemPaths, 771
- ClearParent
 - gazebo::rendering::Visual, 874
- ClearPluginPaths
 - gazebo::common::SystemPaths, 771
- Clone
 - gazebo::rendering::Visual, 874
 - sdf::Element, 272
 - sdf::Param, 551
 - sdf::ParamT, 557
- CloneVisual
 - gazebo::rendering::Scene, 666
- coeffs
 - gazebo::math::Spline, 743
- ColladaLoader
 - gazebo::common::ColladaLoader, 191
- ColladaLoader.hh, 929
- Collision
 - gazebo::physics::Collision, 194
- Collision.hh, 931
- collision1
 - gazebo::physics::Contact, 234
- collision2
 - gazebo::physics::Contact, 234
- Collision_V
 - gazebo::physics, 95
- collisionParent
 - gazebo::physics::Shape, 708
- CollisionPtr
 - gazebo::physics, 95
- CollisionState
 - gazebo::physics::CollisionState, 202
- CollisionState.hh, 932
- Color
 - gazebo::common::Color, 207, 208
- Color.hh, 933
- ColorErr
 - Common, 32
- ColorMsg
 - Common, 33
- Common, 27
 - add_search_path_suffix, 32
 - ColorErr, 32
 - ColorMsg, 33
 - DownloadDependencies, 33
 - find_file, 33
 - find_file_path, 33
 - GetDBConfig, 33
 - GetManifest, 34
 - GetModelConfig, 34
 - GetModelFile, 34
 - GetModelName, 34
 - GetModelPath, 35
 - GetModels, 35
 - GetURI, 35
 - gzclr_end, 31
 - gzclr_start, 31
 - gzdbg, 31
 - gzerr, 31
 - gzlog, 31
 - gzmsg, 31
 - gzthrow, 31
 - gzwarn, 32
 - HasModel, 35
 - Init, 36
 - Log, 36
 - MODEL_PLUGIN, 32
 - NullStream, 32
 - PixelFormatNames, 36
 - PluginType, 32
 - SENSOR_PLUGIN, 32
 - SYSTEM_PLUGIN, 32
 - SetQuiet, 36
 - VISUAL_PLUGIN, 32
 - WORLD_PLUGIN, 32
- Common.hh, 934
- common/Plugin.hh
 - GZ_REGISTER_MODEL_PLUGIN, 1030
 - GZ_REGISTER_SENSOR_PLUGIN, 1030
 - GZ_REGISTER_SYSTEM_PLUGIN, 1031
 - GZ_REGISTER_VISUAL_PLUGIN, 1031
 - GZ_REGISTER_WORLD_PLUGIN, 1031
- CommonTypes.hh, 936
 - GAZEBO_DEPRECATED, 937
 - GAZEBO_FORCEINLINE, 937
 - NULL, 937
- condition
 - gazebo::sensors::SimTimeEvent, 708
- Connect
 - Events, 38
 - gazebo::transport::Connection, 221
- ConnectAddEntity
 - gazebo::event::Events, 295
- ConnectContact
 - gazebo::physics::Collision, 195
- ConnectCreateEntity
 - gazebo::event::Events, 296
- ConnectCreateScene
 - gazebo::rendering::Events, 291
- ConnectDeleteEntity
 - gazebo::event::Events, 296
- ConnectDiagTimerStart
 - gazebo::event::Events, 296

ConnectDiagTimerStop
 gazebo::event::Events, 297
 ConnectEnabled
 gazebo::physics::Link, 418
 ConnectJointUpdate
 gazebo::physics::Joint, 381
 ConnectNewDepthFrame
 gazebo::rendering::DepthCamera, 251
 ConnectNewImageFrame
 gazebo::rendering::Camera, 167
 ConnectNewLaserFrame
 gazebo::rendering::GpuLaser, 322
 gazebo::sensors::GpuRaySensor, 328
 ConnectNewLaserScans
 gazebo::physics::MultiRayShape, 521
 ConnectNewRGBPointCloud
 gazebo::rendering::DepthCamera, 252
 ConnectPause
 gazebo::event::Events, 297
 ConnectPostRender
 gazebo::event::Events, 297
 ConnectPreRender
 gazebo::event::Events, 297
 ConnectPubToSub
 gazebo::transport::TopicManager, 801
 ConnectRemoveScene
 gazebo::rendering::Events, 292
 ConnectRender
 gazebo::event::Events, 298
 ConnectSetSelectedEntity
 gazebo::event::Events, 298
 ConnectStep
 gazebo::event::Events, 298
 ConnectStop
 gazebo::event::Events, 299
 ConnectSubToPub
 gazebo::transport::TopicManager, 801
 ConnectSubscribers
 gazebo::transport::TopicManager, 801
 ConnectToRemoteHost
 gazebo::transport::ConnectionManager, 227
 ConnectToShutdown
 gazebo::transport::Connection, 222
 ConnectUpdated
 gazebo::sensors::Sensor, 686
 ConnectWorldCreated
 gazebo::event::Events, 299
 ConnectWorldUpdateBegin
 gazebo::event::Events, 299
 ConnectWorldUpdateEnd
 gazebo::event::Events, 299
 ConnectWorldUpdateStart
 gazebo::event::Events, 300
 Connection
 gazebo::event::Connection, 218
 gazebo::transport::Connection, 221
 Connection.hh, 938
 HEADER_LENGTH, 940
 Connection_V
 gazebo::event, 87
 ConnectionCount
 Events, 39
 ConnectionManager.hh, 940
 ConnectionPtr
 gazebo::event, 87
 gazebo::transport, 104
 ConnectionReadTask
 gazebo::transport::ConnectionReadTask, 230
 connections
 gazebo::physics::Entity, 287
 gazebo::rendering::Camera, 182
 gazebo::sensors::Sensor, 692
 Console.hh, 941
 ConstUrdfLinkPtr
 Gazebo_parser, 68
 Contact
 gazebo::physics::Contact, 233
 Contact.hh, 943
 MAX_COLLIDE_RETURNS, 944
 MAX_CONTACT_JOINTS, 944
 contactFiducial
 gazebo::physics::RayShape, 639
 contactLen
 gazebo::physics::RayShape, 639
 ContactManager
 gazebo::physics::ContactManager, 236
 contactManager
 gazebo::physics::PhysicsEngine, 569
 ContactManager.hh, 944
 ContactPtr
 gazebo::physics, 95
 contactRetro
 gazebo::physics::RayShape, 639
 ContactSensor
 gazebo::sensors::ContactSensor, 239
 ContactSensor.hh, 945
 ContactSensor_V
 gazebo::sensors, 102
 ContactSensorPtr
 gazebo::sensors, 102
 ContactVisual
 gazebo::rendering::ContactVisual, 243
 ContactVisual.hh, 945
 ContactVisualPtr
 gazebo::rendering, 99
 control
 gazebo::common::MouseEvent, 505
 Conversions.hh, 946

- Convert
 - gazebo::rendering::Conversions, 244, 245
 - Messages, 56–59
 - sdf::Converter, 246
- ConvertPixelFormat
 - gazebo::common::Image, 358
- Converter.hh, 947
- CoordPoseSolve
 - gazebo::math::Pose, 587
- CoordPositionAdd
 - gazebo::math::Pose, 587
- CoordPositionSub
 - gazebo::math::Pose, 588
- CoordRotationAdd
 - gazebo::math::Pose, 588
- CoordRotationSub
 - gazebo::math::Pose, 588
- Copy
 - sdf::Element, 273
- copyChildren
 - sdf, 107
- CopyNormals
 - gazebo::common::SubMesh, 753
- CopyVertices
 - gazebo::common::SubMesh, 753
- Correct
 - gazebo::math::Pose, 589
 - gazebo::math::Quaternion, 614
 - gazebo::math::Vector3, 842
- count
 - gazebo::physics::Contact, 234
- Create
 - gazebo::PluginT, 583
- create_scene
 - Rendering, 66
- create_sensor
 - Sensors, 71
- create_world
 - Classes for physics and dynamics, 44
- CreateBoolean
 - gazebo::common::MeshCSG, 473
- CreateBox
 - gazebo::common::MeshManager, 476
- CreateCamera
 - gazebo::common::MeshManager, 477
 - gazebo::rendering::Scene, 666
- CreateCollision
 - gazebo::physics::PhysicsEngine, 561
- CreateCone
 - gazebo::common::MeshManager, 477
- CreateCylinder
 - gazebo::common::MeshManager, 477
- CreateDepthCamera
 - gazebo::rendering::Scene, 666
- CreateDepthTexture
 - gazebo::rendering::DepthCamera, 252
- CreateDynamicLine
 - gazebo::rendering::Visual, 874
- CreateGrid
 - gazebo::rendering::Scene, 667
- CreateJoint
 - gazebo::physics::PhysicsEngine, 562
- CreateKeyFrame
 - gazebo::common::NumericAnimation, 543
 - gazebo::common::PoseAnimation, 594
- CreateLaserTexture
 - gazebo::rendering::GpuLaser, 322
- CreateLink
 - gazebo::physics::PhysicsEngine, 562
- CreatePlane
 - gazebo::common::MeshManager, 477, 478
 - gazebo::physics::PlaneShape, 579
- CreateRenderTexture
 - gazebo::rendering::Camera, 167
- CreateRequest
 - Messages, 59
- CreateScene
 - gazebo::rendering::RenderEngine, 641
- createScene
 - gazebo::rendering::Events, 292
- CreateSensor
 - gazebo::sensors::SensorManager, 696
- CreateShape
 - gazebo::physics::PhysicsEngine, 562
- CreateSphere
 - gazebo::common::MeshManager, 478
- CreateTube
 - gazebo::common::MeshManager, 478
- CreateUserCamera
 - gazebo::rendering::Scene, 667
- CreateVertexDeclaration
 - gazebo::rendering::DynamicLines, 264
 - gazebo::rendering::DynamicRenderable, 267
- CreateWindow
 - gazebo::rendering::GUIOverlay, 343
 - gazebo::rendering::WindowManager, 889
- Cross
 - gazebo::math::Vector2d, 823
 - gazebo::math::Vector2i, 832
 - gazebo::math::Vector3, 842
- cvfov
 - gazebo::sensors::GpuRaySensor, 334
- CylinderShape
 - gazebo::physics::CylinderShape, 248
- CylinderShape.hh, 948
- CylinderShapePtr
 - gazebo::physics, 95

- d
 - gazebo::math::Plane, 577
- DEFERRED
 - gazebo::rendering::RenderEngine, 641
- DIAG_TIMER_LAP
 - Utility, 79
- DIAG_TIMER_START
 - Utility, 79
- DIAG_TIMER_STOP
 - Utility, 79
- DIFFERENCE
 - gazebo::common::MeshCSG, 473
- dampingCoefficient
 - gazebo::physics::Joint, 392
- data
 - sdf::Plugin, 582
- DebugPrint
 - gazebo::physics::PhysicsEngine, 562
- DebugString
 - gazebo::physics::Contact, 233
- DecCount
 - gazebo::transport::IOManager, 375
- DecodeTopicName
 - gazebo::transport::Node, 528
- defaultValue
 - sdf::ParamT, 558
- defaultVpParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperCg, 702
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperGLSL, 704
- Degree
 - gazebo::math::Angle, 118
- DeleteDynamicLine
 - gazebo::rendering::Visual, 874
- deleteEntity
 - gazebo::event::Events, 303
- DepthCamera
 - gazebo::rendering::DepthCamera, 251
- DepthCamera.hh, 949
- DepthCameraPtr
 - gazebo::rendering, 99
- DepthCameraSensor
 - gazebo::sensors::DepthCameraSensor, 255
- DepthCameraSensor.hh, 949
- DepthCameraSensor_V
 - gazebo::sensors, 102
- DepthCameraSensorPtr
 - gazebo::sensors, 102
- depthTarget
 - gazebo::rendering::DepthCamera, 254
- depthTexture
 - gazebo::rendering::DepthCamera, 254
- depthViewport
 - gazebo::rendering::DepthCamera, 254
- depths
 - gazebo::physics::Contact, 234
- description
 - sdf::Param, 554
- Detach
 - gazebo::physics::Joint, 381
- DetachAllStaticModels
 - gazebo::physics::Link, 418
- DetachEntity
 - gazebo::rendering::RTShaderSystem, 660
- DetachObjects
 - gazebo::rendering::Visual, 874
- DetachStaticModel
 - gazebo::physics::Link, 418
 - gazebo::physics::Model, 484
- DetachViewport
 - gazebo::rendering::RTShaderSystem, 660
- DetachVisual
 - gazebo::rendering::Visual, 874, 875
- diagTimerStart
 - gazebo::event::Events, 303
- diagTimerStop
 - gazebo::event::Events, 304
- DiagnosticTimer
 - gazebo::util::DiagnosticTimer, 261
- DiagnosticTimerPtr
 - gazebo::common, 86
 - gazebo::util, 105
- Diagnostics.hh, 950
- diffuse
 - gazebo::common::Material, 455
- dirtyPose
 - gazebo::physics::Entity, 288
- dirtyPoses
 - gazebo::physics::World, 903
- DisableAllModels
 - gazebo::physics::World, 896
- DisableTrackVisual
 - gazebo::rendering::Visual, 875
- Disconnect
 - Events, 39, 40
 - gazebo::event::Event, 290
- DisconnectAddEntity
 - gazebo::event::Events, 300
- DisconnectContact
 - gazebo::physics::Collision, 195
- DisconnectCreateEntity
 - gazebo::event::Events, 300
- DisconnectCreateScene
 - gazebo::rendering::Events, 292
- DisconnectDeleteEntity
 - gazebo::event::Events, 300
- DisconnectDiagTimerStart

- gazebo::event::Events, 301
- DisconnectDiagTimerStop
 - gazebo::event::Events, 301
- DisconnectEnabled
 - gazebo::physics::Link, 418
- DisconnectJointUpdate
 - gazebo::physics::Joint, 381
- DisconnectNewDepthFrame
 - gazebo::rendering::DepthCamera, 252
- DisconnectNewImageFrame
 - gazebo::rendering::Camera, 167
- DisconnectNewLaserFrame
 - gazebo::rendering::GpuLaser, 323
 - gazebo::sensors::GpuRaySensor, 328
- DisconnectNewLaserScans
 - gazebo::physics::MultiRayShape, 522
- DisconnectNewRGBPointCloud
 - gazebo::rendering::DepthCamera, 252
- DisconnectPause
 - gazebo::event::Events, 301
- DisconnectPostRender
 - gazebo::event::Events, 301
- DisconnectPreRender
 - gazebo::event::Events, 301
- DisconnectPubFromSub
 - gazebo::transport::TopicManager, 801
- DisconnectRemoveScene
 - gazebo::rendering::Events, 292
- DisconnectRender
 - gazebo::event::Events, 302
- DisconnectSetSelectedEntity
 - gazebo::event::Events, 302
- DisconnectShutdown
 - gazebo::transport::Connection, 222
- DisconnectStep
 - gazebo::event::Events, 302
- DisconnectStop
 - gazebo::event::Events, 302
- DisconnectSubFromPub
 - gazebo::transport::TopicManager, 802
- DisconnectUpdated
 - gazebo::sensors::Sensor, 687
- DisconnectWorldCreated
 - gazebo::event::Events, 302
- DisconnectWorldUpdateBegin
 - gazebo::event::Events, 303
- DisconnectWorldUpdateEnd
 - gazebo::event::Events, 303
- DisconnectWorldUpdateStart
 - gazebo::event::Events, 303
- Distance
 - gazebo::math::Plane, 577
 - gazebo::math::Vector2d, 824
 - gazebo::math::Vector2i, 832
 - gazebo::math::Vector3, 842
 - gazebo::math::Vector4, 854
- Dot
 - gazebo::math::Quaternion, 614
 - gazebo::math::Vector3, 843
- Double
 - gazebo::common::Time, 780
- DownloadDependencies
 - Common, 33
- dragging
 - gazebo::common::MouseEvent, 505
- DrawLine
 - gazebo::rendering::Scene, 667
- dummyContext
 - gazebo::rendering::RenderEngine, 643
- dummyDisplay
 - gazebo::rendering::RenderEngine, 643
- dummyWindowId
 - gazebo::rendering::RenderEngine, 643
- duration
 - gazebo::physics::TrajectoryInfo, 805
- DynamicLines
 - gazebo::rendering::DynamicLines, 263
- DynamicLines.hh, 951
- DynamicLinesPtr
 - gazebo::rendering, 99
- DynamicRenderable
 - gazebo::rendering::DynamicRenderable, 267
- DynamicRenderable.hh, 952
- ENTITY
 - gazebo::physics::Base, 138
- ERP
 - gazebo::physics::Joint, 380
- effortLimit
 - gazebo::physics::Joint, 392
- Element
 - sdf::Element, 272
- ElementPtr
 - sdf, 107
- ElementPtr_V
 - sdf, 107
- emissive
 - gazebo::common::Material, 455
- Enable
 - gazebo::rendering::Grid, 338
- EnableAllModels
 - gazebo::physics::World, 896
- EnablePhysicsEngine
 - gazebo::physics::World, 896
- EnableSaveFrame
 - gazebo::rendering::Camera, 168
- EnableTrackVisual
 - gazebo::rendering::Visual, 875

- EnableViewController
 - gazebo::rendering::UserCamera, 816
- enabled
 - gazebo::rendering::ViewController, 867
- EncodeTopicName
 - gazebo::transport::Node, 528
- endTime
 - gazebo::physics::TrajectoryInfo, 805
- EnqueueMsg
 - gazebo::transport::Connection, 222
- Entity
 - gazebo::physics::Entity, 280
- Entity.hh, 953
- entityCreated
 - gazebo::event::Events, 304
- EntityPtr
 - gazebo::physics, 95
- EntityType
 - gazebo::physics::Base, 138
- EntityTypename
 - Classes for physics and dynamics, 46
- Equal
 - gazebo::math::Vector3, 843
- equal
 - Math, 50
- erp
 - gazebo::physics::SurfaceParams, 766
- EulerToQuaternion
 - gazebo::math::Quaternion, 614
- Event.hh, 954
- eventConnections
 - gazebo::transport::ConnectionManager, 230
- EventType
 - gazebo::common::MouseEvent, 504
- Events, 38
 - ~EventT, 38
 - Connect, 38
 - ConnectionCount, 39
 - Disconnect, 39, 40
- Events.hh, 955
- Exception
 - gazebo::common::Exception, 315
- Exception.hh, 956
- execute
 - gazebo::transport::ConnectionReadTask, 231
- FACE_MAX
 - STLLoader.hh, 1077
- FLAT
 - gazebo::common::Material, 449
- FMAX
 - gazebo::physics::Joint, 380
- FORWARD
 - gazebo::rendering::RenderEngine, 641
- FPSViewController
 - gazebo::rendering::FPSViewController, 318
- FPSViewController.hh, 957
- FUDGE_FACTOR
 - gazebo::physics::Joint, 379
- fakeAnchor
 - gazebo::physics::ScrewJoint, 680
 - gazebo::physics::SliderJoint, 734
- far
 - gazebo::sensors::GpuRaySensor, 335
- fdir1
 - gazebo::physics::SurfaceParams, 766
- filename
 - gazebo::PluginT, 584
 - sdf::Plugin, 582
- FillArrays
 - gazebo::common::Mesh, 468
 - gazebo::common::SubMesh, 753
- FillHardwareBuffers
 - gazebo::rendering::DynamicLines, 264
 - gazebo::rendering::DynamicRenderable, 267
- FillMsg
 - gazebo::physics::BoxShape, 153
 - gazebo::physics::Collision, 195
 - gazebo::physics::Contact, 233
 - gazebo::physics::CylinderShape, 248
 - gazebo::physics::HeightmapShape, 350
 - gazebo::physics::Joint, 381
 - gazebo::physics::Link, 419
 - gazebo::physics::Model, 484
 - gazebo::physics::MultiRayShape, 522
 - gazebo::physics::PlaneShape, 579
 - gazebo::physics::RayShape, 636
 - gazebo::physics::Shape, 707
 - gazebo::physics::SphereShape, 738
 - gazebo::physics::SurfaceParams, 765
 - gazebo::physics::TrimeshShape, 807
 - gazebo::rendering::Light, 407
 - gazebo::sensors::Sensor, 687
- FillSDF
 - gazebo::physics::CollisionState, 203
 - gazebo::physics::JointState, 397
 - gazebo::physics::LinkState, 433
 - gazebo::physics::ModelState, 499
 - gazebo::physics::WorldState, 907
- find_file
 - Common, 33
 - gazebo, 83
- find_file_path
 - Common, 33
- FindFile
 - gazebo::common::SystemPaths, 771
- FindFileURI
 - gazebo::common::SystemPaths, 771

- FindPublication
 - gazebo::transport::TopicManager, 802
- Fini
 - gazebo::common::LogRecord, 442
 - gazebo::Master, 446
 - gazebo::physics::Actor, 112
 - gazebo::physics::Base, 140
 - gazebo::physics::Collision, 195
 - gazebo::physics::Entity, 280
 - gazebo::physics::Link, 419
 - gazebo::physics::Model, 484
 - gazebo::physics::PhysicsEngine, 562
 - gazebo::physics::World, 896
 - gazebo::rendering::Camera, 168
 - gazebo::rendering::DepthCamera, 253
 - gazebo::rendering::GpuLaser, 323
 - gazebo::rendering::RenderEngine, 642
 - gazebo::rendering::RTShaderSystem, 661
 - gazebo::rendering::UserCamera, 816
 - gazebo::rendering::Visual, 875
 - gazebo::rendering::WindowManager, 890
 - gazebo::sensors::CameraSensor, 186
 - gazebo::sensors::ContactSensor, 239
 - gazebo::sensors::DepthCameraSensor, 255
 - gazebo::sensors::GpuRaySensor, 328
 - gazebo::sensors::ImuSensor, 363
 - gazebo::sensors::MultiCameraSensor, 515
 - gazebo::sensors::RaySensor, 629
 - gazebo::sensors::RFIDSensor, 645
 - gazebo::sensors::RFIDTag, 647
 - gazebo::sensors::Sensor, 687
 - gazebo::sensors::SensorManager, 696
 - gazebo::Server, 700
 - gazebo::transport::ConnectionManager, 227
 - gazebo::transport::Node, 529
 - gazebo::transport::PublicationTransport, 605
 - gazebo::transport::TopicManager, 802
- fini
 - Classes for physics and dynamics, 44
 - gazebo, 83
 - Rendering, 66
 - Sensors, 71
 - Transport, 76
- Float
 - gazebo::common::Time, 780
- FogFromSDF
 - Messages, 59
- forceApplied
 - gazebo::physics::Joint, 392
- g
 - gazebo::common::Color, 215
- GAZEBO_DEPRECATED
 - CommonTypes.hh, 937
- GAZEBO_FORCEINLINE
 - CommonTypes.hh, 937
- GOURAUD
 - gazebo::common::Material, 449
- GPtrArray
 - MeshCSG.hh, 997
- GUIFromSDF
 - Messages, 60
- GUIOverlay
 - gazebo::rendering::GUIOverlay, 342
- GUIOverlay.hh, 964
- GUIPluginPtr
 - gazebo, 83
- GZ_ALL_COLLIDE
 - PhysicsTypes.hh, 1024
- GZ_ASSERT
 - Assert.hh, 918
- GZ_DBL_MAX
 - Helpers.hh, 968
- GZ_DBL_MIN
 - Helpers.hh, 968
- GZ_DTOR
 - Angle.hh, 914
- GZ_FIXED_COLLIDE
 - PhysicsTypes.hh, 1024
- GZ_FLT_MAX
 - Helpers.hh, 968
- GZ_FLT_MIN
 - Helpers.hh, 968
- GZ_GHOST_COLLIDE
 - PhysicsTypes.hh, 1024
- GZ_LOG_VERSION
 - LogRecord.hh, 989
- GZ_MODEL_DB_MANIFEST_FILENAME
 - ModelDatabase.hh, 1002
- GZ_MODEL_MANIFEST_FILENAME
 - ModelDatabase.hh, 1002
- GZ_NONE_COLLIDE
 - PhysicsTypes.hh, 1024
- GZ_NORMALIZE
 - Angle.hh, 914
- GZ_REGISTER_MODEL_PLUGIN
 - common/Plugin.hh, 1030
- GZ_REGISTER_PHYSICS_ENGINE
 - Classes for physics and dynamics, 44
- GZ_REGISTER_SENSOR_PLUGIN
 - common/Plugin.hh, 1030
- GZ_REGISTER_STATIC_MSG
 - Messages, 56
- GZ_REGISTER_STATIC_SENSOR
 - Sensors, 70
- GZ_REGISTER_SYSTEM_PLUGIN
 - common/Plugin.hh, 1031
- GZ_REGISTER_VISUAL_PLUGIN

- common/Plugin.hh, 1031
- GZ_REGISTER_WORLD_PLUGIN
 - common/Plugin.hh, 1031
- GZ_RTOD
 - Angle.hh, 914
- GZ_SENSOR_COLLIDE
 - PhysicsTypes.hh, 1024
- GZ_VISIBILITY_ALL
 - RenderTypes.hh, 1047
- GZ_VISIBILITY_GUI
 - RenderTypes.hh, 1047
- GZ_VISIBILITY_NOT_SELECTABLE
 - RenderTypes.hh, 1047
- GZ_VISIBILITY_SELECTION
 - RenderTypes.hh, 1047
- gazebo, 81
 - add_plugin, 83
 - find_file, 83
 - fini, 83
 - GUIPluginPtr, 83
 - init, 83
 - load, 83
 - ModelPluginPtr, 83
 - print_version, 83
 - run, 83
 - SensorPluginPtr, 83
 - stop, 83
 - SystemPluginPtr, 83
 - VisualPluginPtr, 83
 - WorldPluginPtr, 83
- gazebo.hh, 958
- gazebo::Master, 445
 - ~Master, 445
 - Fini, 446
 - Init, 446
 - Master, 445
 - Run, 446
 - RunOnce, 446
 - RunThread, 446
 - Stop, 446
- gazebo::ModelPlugin, 495
 - ~ModelPlugin, 496
 - Init, 496
 - Load, 496
 - ModelPlugin, 495
 - Reset, 496
- gazebo::PluginT
 - Create, 583
 - filename, 584
 - GetFilename, 584
 - GetHandle, 584
 - GetType, 584
 - handle, 584
 - TPtr, 583
 - type, 584
- gazebo::PluginT < T >, 582
- gazebo::SensorPlugin, 698
 - ~SensorPlugin, 699
 - Init, 699
 - Load, 699
 - Reset, 699
 - SensorPlugin, 699
- gazebo::Server, 700
 - ~Server, 700
 - Fini, 700
 - GetInitialized, 700
 - Init, 700
 - LoadFile, 700
 - LoadString, 701
 - ParseArgs, 701
 - PrintUsage, 701
 - Run, 701
 - Server, 700
 - SetParams, 701
 - Stop, 701
 - systemPluginsArgc, 701
 - systemPluginsArgv, 701
- gazebo::SystemPlugin, 773
 - ~SystemPlugin, 774
 - Init, 774
 - Load, 774
 - Reset, 774
 - SystemPlugin, 774
- gazebo::VisualPlugin, 887
 - Init, 888
 - Load, 888
 - Reset, 888
 - VisualPlugin, 888
- gazebo::WorldPlugin, 904
 - ~WorldPlugin, 904
 - Init, 905
 - Load, 905
 - Reset, 905
 - WorldPlugin, 904
- gazebo::common, 83
 - AnimationPtr, 86
 - DiagnosticTimerPtr, 86
 - NodeMap, 86
 - NodeMapIter, 86
 - NumericAnimationPtr, 86
 - Param_V, 86
 - PoseAnimationPtr, 86
 - RawNodeAnim, 86
 - RawNodeWeights, 86
 - RawSkeletonAnim, 86
 - StrStr_M, 86
- gazebo::common::Animation, 123
 - ~Animation, 125

- AddTime, 125
- Animation, 125
- build, 126
- GetKeyFrame, 125
- GetKeyFrameCount, 125
- GetKeyFramesAtTime, 125
- GetLength, 126
- GetTime, 126
- KeyFrame_V, 124
- keyFrames, 126
- length, 127
- loop, 127
- name, 127
- SetLength, 126
- SetTime, 126
- timePos, 127
- gazebo::common::AssertionInternalError, 129
 - ~AssertionInternalError, 131
 - AssertionInternalError, 130
- gazebo::common::BVHLoader, 154
 - ~BVHLoader, 154
 - BVHLoader, 154
 - Load, 154
- gazebo::common::ColladaLoader, 190
 - ~ColladaLoader, 191
 - ColladaLoader, 191
 - Load, 191
- gazebo::common::Color, 205
 - ~Color, 208
 - a, 215
 - ABGR, 207
 - ARGB, 207
 - b, 215
 - BGRA, 207
 - Black, 215
 - Blue, 215
 - Color, 207, 208
 - g, 215
 - GetAsABGR, 208
 - GetAsARGB, 208
 - GetAsBGRA, 208
 - GetAsHSV, 209
 - GetAsRGBA, 209
 - GetAsYUV, 209
 - Green, 215
 - operator<<, 215
 - operator>>, 215
 - operator*, 209, 210
 - operator*=: 210
 - operator+, 210
 - operator+=, 211
 - operator-, 211
 - operator=, 211
 - operator/, 212
 - operator/=: 212
 - operator=, 212
 - operator==, 213
 - operator[], 213
 - Purple, 215
 - r, 215
 - RGBA, 207
 - Red, 216
 - Reset, 213
 - Set, 213
 - SetFromABGR, 213
 - SetFromARGB, 214
 - SetFromBGRA, 214
 - SetFromHSV, 214
 - SetFromRGBA, 214
 - SetFromYUV, 214
 - White, 216
 - Yellow, 216
- gazebo::common::Console, 231
- gazebo::common::Exception, 314
 - ~Exception, 315
 - Exception, 315
 - GetErrorFile, 315
 - GetErrorStr, 316
 - operator<<, 316
 - Print, 316
- gazebo::common::Image, 356
 - ~Image, 358
 - BAYER_GBRG8, 358
 - BAYER_GRBG8, 358
 - BAYER_RGGB8, 358
 - BAYER_RGGR8, 358
 - BGR_INT16, 357
 - BGR_INT32, 357
 - BGR_INT8, 357
 - BGRA_INT8, 357
 - ConvertPixelFormat, 358
 - GetAvgColor, 358
 - GetBPP, 359
 - GetData, 359
 - GetFilename, 359
 - GetHeight, 359
 - GetMaxColor, 359
 - GetPitch, 359
 - GetPixel, 359
 - GetPixelFormat, 360
 - GetRGBData, 360
 - GetWidth, 360
 - Image, 358
 - L_INT16, 357
 - L_INT8, 357
 - Load, 360
 - PIXEL_FORMAT_COUNT, 358
 - PixelFormat, 357

- R_FLOAT16, 358
- R_FLOAT32, 358
- RGB_FLOAT16, 358
- RGB_FLOAT32, 358
- RGB_INT16, 357
- RGB_INT32, 357
- RGB_INT8, 357
- RGBA_INT8, 357
- Rescale, 360
- SavePNG, 361
- SetFromData, 361
- UNKNOWN_PIXEL_FORMAT, 357
- Valid, 361
- gazebo::common::InternalError, 372
 - ~InternalError, 374
 - InternalError, 373
- gazebo::common::KeyFrame, 402
 - ~KeyFrame, 403
 - GetTime, 403
 - KeyFrame, 403
 - time, 403
- gazebo::common::LogPlay, 436
 - GetChunk, 437
 - GetChunkCount, 438
 - GetEncoding, 438
 - GetGazeboVersion, 438
 - GetLogVersion, 438
 - GetRandSeed, 438
 - IsOpen, 438
 - Open, 439
 - Step, 439
- gazebo::common::LogRecord, 440
 - Add, 441
 - Fini, 442
 - GetBasePath, 442
 - GetEncoding, 442
 - GetFileSize, 442
 - GetFilename, 442
 - GetFirstUpdate, 442
 - GetPaused, 443
 - GetRunTime, 443
 - GetRunning, 443
 - Init, 443
 - Remove, 443
 - SetBasePath, 444
 - SetPaused, 444
 - Start, 444
 - Stop, 444
- gazebo::common::Material, 446
 - ~Material, 449
 - ADD, 449
 - ambient, 454
 - BLEND_COUNT, 449
 - BLINN, 449
 - BlendMode, 449
 - blendMode, 454
 - BlendModeStr, 454
 - diffuse, 455
 - emissive, 455
 - FLAT, 449
 - GOURAUD, 449
 - GetAmbient, 450
 - GetBlendFactors, 450
 - GetBlendMode, 450
 - GetDepthWrite, 450
 - GetDiffuse, 450
 - GetEmissive, 450
 - GetLighting, 450
 - GetName, 451
 - GetPointSize, 451
 - GetShadeMode, 451
 - GetShininess, 451
 - GetSpecular, 451
 - GetTextureImage, 451
 - GetTransparency, 452
 - MODULATE, 449
 - Material, 449
 - name, 455
 - operator<<, 454
 - PHONG, 449
 - pointSize, 455
 - REPLACE, 449
 - SHADE_COUNT, 449
 - SetAmbient, 452
 - SetBlendFactors, 452
 - SetBlendMode, 452
 - SetDepthWrite, 452
 - SetDiffuse, 453
 - SetEmissive, 453
 - SetLighting, 453
 - SetPointSize, 453
 - SetShadeMode, 453
 - SetShininess, 453
 - SetSpecular, 453
 - SetTextureImage, 454
 - SetTransparency, 454
 - ShadeMode, 449
 - shadeMode, 455
 - ShadeModeStr, 455
 - shininess, 455
 - specular, 455
 - texImage, 455
 - transparency, 455
- gazebo::common::Mesh, 466
 - ~Mesh, 468
 - AddMaterial, 468
 - AddSubMesh, 468
 - FillArrays, 468

- GenSphericalTexCoord, 469
- GetAABB, 469
- GetIndexCount, 469
- GetMaterial, 469
- GetMaterialCount, 469
- GetMax, 470
- GetMin, 470
- GetName, 470
- GetNormalCount, 470
- GetPath, 470
- GetSkeleton, 470
- GetSubMesh, 470
- GetSubMeshCount, 471
- GetTexCoordCount, 471
- GetVertexCount, 471
- HasSkeleton, 471
- Mesh, 468
- RecalculateNormals, 471
- Scale, 471
- SetName, 472
- SetPath, 472
- SetScale, 472
- SetSkeleton, 472
- gazebo::common::MeshCSG, 472
 - ~MeshCSG, 473
 - BooleanOperation, 473
 - CreateBoolean, 473
 - DIFFERENCE, 473
 - INTERSECTION, 473
 - MeshCSG, 473
 - UNION, 473
- gazebo::common::MeshLoader, 474
 - ~MeshLoader, 474
 - Load, 475
 - MeshLoader, 474
- gazebo::common::MeshManager, 475
 - AddMesh, 476
 - CreateBox, 476
 - CreateCamera, 477
 - CreateCone, 477
 - CreateCylinder, 477
 - CreatePlane, 477, 478
 - CreateSphere, 478
 - CreateTube, 478
 - GenSphericalTexCoord, 478
 - GetMesh, 479
 - GetMeshAABB, 479
 - HasMesh, 479
 - IsValidFilename, 479
 - Load, 479
- gazebo::common::ModelDatabase, 493
- gazebo::common::MouseEvent, 503
 - alt, 505
 - button, 505
 - Buttons, 504
 - buttons, 505
 - control, 505
 - dragging, 505
 - EventType, 504
 - LEFT, 504
 - MIDDLE, 504
 - MOVE, 505
 - MouseEvent, 505
 - moveScale, 505
 - NO_BUTTON, 504
 - NO_EVENT, 505
 - PRESS, 505
 - pos, 505
 - pressPos, 505
 - prevPos, 506
 - RELEASE, 505
 - RIGHT, 504
 - SCROLL, 505
 - scroll, 506
 - shift, 506
 - type, 506
- gazebo::common::NodeAnimation, 532
 - ~NodeAnimation, 533
 - AddKeyFrame, 533, 534
 - GetFrameAt, 534
 - GetFrameCount, 534
 - GetKeyFrame, 534
 - GetLength, 535
 - GetName, 535
 - GetTimeAtX, 535
 - keyFrames, 536
 - length, 536
 - name, 536
 - NodeAnimation, 533
 - Scale, 535
 - SetName, 535
- gazebo::common::NodeAssignment, 536
 - nodeIndex, 537
 - vertexIndex, 537
 - weight, 537
- gazebo::common::NodeTransform, 537
 - ~NodeTransform, 539
 - Get, 539
 - GetSID, 539
 - GetType, 539
 - MATRIX, 538
 - NodeTransform, 539
 - operator*, 540
 - operator(), 539
 - PrintSource, 540
 - ROTATE, 538
 - RecalculateMatrix, 540
 - SCALE, 538

- Set, 540
- SetComponent, 540
- SetSID, 541
- SetSourceValues, 541
- SetType, 541
- sid, 541
- source, 541
- TRANSLATE, 538
- transform, 542
- TransformType, 538
- type, 542
- gazebo::common::NumericAnimation, 542
 - ~NumericAnimation, 543
 - CreateKeyFrame, 543
 - GetInterpolatedKeyFrame, 543
 - NumericAnimation, 543
- gazebo::common::NumericKeyFrame, 544
 - ~NumericKeyFrame, 545
 - GetValue, 545
 - NumericKeyFrame, 544
 - SetValue, 545
 - value, 545
- gazebo::common::PID, 571
 - ~PID, 572
 - GetCmd, 572
 - GetErrors, 573
 - Init, 573
 - operator=, 573
 - PID, 572
 - Reset, 573
 - SetCmd, 573
 - SetCmdMax, 574
 - SetCmdMin, 574
 - SetDGain, 574
 - SetIGain, 574
 - SetIMax, 574
 - SetIMin, 574
 - SetPGain, 575
 - Update, 575
- gazebo::common::PoseAnimation, 593
 - ~PoseAnimation, 594
 - BuildInterpolationSplines, 594
 - CreateKeyFrame, 594
 - GetInterpolatedKeyFrame, 594, 595
 - PoseAnimation, 594
- gazebo::common::PoseKeyFrame, 595
 - ~PoseKeyFrame, 596
 - GetRotation, 596
 - GetTranslation, 596
 - PoseKeyFrame, 596
 - rotate, 597
 - SetRotation, 597
 - SetTranslation, 597
 - translate, 597
- gazebo::common::STLLoader, 747
 - ~STLLoader, 748
 - Load, 748
 - STLLoader, 748
- gazebo::common::Skeleton, 712
 - ~Skeleton, 714
 - AddAnimation, 714
 - AddVertNodeWeight, 714
 - anims, 718
 - bindShapeTransform, 718
 - BuildNodeMap, 715
 - GetAnimation, 715
 - GetBindShapeTransform, 715
 - GetNodeByHandle, 715
 - GetNodeById, 715
 - GetNodeByName, 716
 - GetNodes, 716
 - GetNumAnimations, 716
 - GetNumJoints, 716
 - GetNumNodes, 716
 - GetNumVertNodeWeights, 716
 - GetRootNode, 717
 - GetVertNodeWeight, 717
 - nodes, 718
 - PrintTransforms, 717
 - rawNW, 718
 - root, 718
 - Scale, 717
 - SetBindShapeTransform, 717
 - SetNumVertAttached, 718
 - SetRootNode, 718
 - Skeleton, 714
- gazebo::common::SkeletonAnimation, 719
 - ~SkeletonAnimation, 720
 - AddKeyFrame, 720
 - animations, 722
 - GetLength, 720
 - GetName, 721
 - GetNodeCount, 721
 - GetNodePoseAt, 721
 - GetPoseAt, 721
 - GetPoseAtX, 722
 - HasNode, 722
 - length, 722
 - name, 723
 - Scale, 722
 - SetName, 722
 - SkeletonAnimation, 720
- gazebo::common::SkeletonNode, 723
 - ~SkeletonNode, 726
 - AddChild, 726
 - AddRawTransform, 726
 - children, 731
 - GetChild, 726

- GetChildById, 726
- GetChildByName, 727
- GetChildCount, 727
- GetHandle, 727
- GetId, 727
- GetInverseBindTransform, 727
- GetModelTransform, 727
- GetName, 728
- GetNumRawTrans, 728
- GetParent, 728
- GetRawTransform, 728
- GetRawTransforms, 728
- GetTransform, 729
- GetTransforms, 729
- handle, 731
- id, 731
- initialTransform, 731
- invBindTransform, 731
- IsJoint, 729
- IsRootNode, 729
- JOINT, 725
- modelTransform, 731
- NODE, 725
- name, 731
- parent, 732
- rawTransforms, 732
- Reset, 729
- SetHandle, 729
- SetId, 729
- SetInitialTransform, 730
- SetInverseBindTransform, 730
- SetModelTransform, 730
- SetName, 730
- SetParent, 730
- SetTransform, 730
- SetType, 731
- SkeletonNode, 725, 726
- SkeletonNodeType, 725
- transform, 732
- type, 732
- UpdateChildrenTransforms, 731
- gazebo::common::SubMesh, 749
 - ~SubMesh, 751
 - AddIndex, 751
 - AddNodeAssignment, 752
 - AddNormal, 752
 - AddTexCoord, 752
 - AddVertex, 752
 - CopyNormals, 753
 - CopyVertices, 753
 - FillArrays, 753
 - GenSphericalTexCoord, 753
 - GetIndex, 753
 - GetIndexCount, 754
 - GetMaterialIndex, 754
 - GetMax, 754
 - GetMaxIndex, 754
 - GetMin, 754
 - GetNodeAssignment, 754
 - GetNodeAssignmentsCount, 754
 - GetNormal, 754
 - GetNormalCount, 755
 - GetPrimitiveType, 755
 - GetTexCoord, 755
 - GetTexCoordCount, 755
 - GetVertex, 755
 - GetVertexCount, 755
 - GetVertexIndex, 756
 - HasVertex, 756
 - LINES, 751
 - LINESTRIPS, 751
 - POINTS, 751
 - PrimitiveType, 751
 - RecalculateNormals, 756
 - Scale, 756
 - SetIndexCount, 756
 - SetMaterialIndex, 756
 - SetNormal, 757
 - SetNormalCount, 757
 - SetPrimitiveType, 757
 - SetScale, 757
 - SetSubMeshCenter, 757
 - SetTexCoord, 757
 - SetTexCoordCount, 758
 - SetVertex, 758
 - SetVertexCount, 758
 - SubMesh, 751
 - TRIANGLES, 751
 - TRIFANS, 751
 - TRISTRIPS, 751
- gazebo::common::SystemPaths, 768
 - AddGazeboPaths, 770
 - AddModelPaths, 770
 - AddOgrePaths, 770
 - AddPluginPaths, 770
 - AddSearchPathSuffix, 770
 - ClearGazeboPaths, 771
 - ClearModelPaths, 771
 - ClearOgrePaths, 771
 - ClearPluginPaths, 771
 - FindFile, 771
 - FindFileURI, 771
 - gazeboPathsFromEnv, 773
 - GetGazeboPaths, 771
 - GetLogPath, 772
 - GetModelPaths, 772
 - GetOgrePaths, 772
 - GetPluginPaths, 772

- GetWorldPathExtension, 772
- modelPathsFromEnv, 773
- ogrePathsFromEnv, 773
- pluginPathsFromEnv, 773
- gazebo::common::Time, 775
 - ~Time, 779
 - Double, 780
 - Float, 780
 - GetWallTime, 780
 - GetWallTimeAsISOString, 780
 - MSleep, 781
 - MicToNano, 780
 - MilToNano, 780
 - NSleep, 781
 - nsec, 796
 - operator<, 788, 789
 - operator<<, 795
 - operator<=, 789, 790
 - operator>, 792, 793
 - operator>>, 795
 - operator>=, 793, 794
 - operator*, 782, 783
 - operator*=:, 783, 784
 - operator+, 784
 - operator+=, 785
 - operator-, 785, 786
 - operator-=, 786, 787
 - operator/, 787
 - operator/=, 788
 - operator=, 790, 791
 - operator==, 791, 792
 - sec, 796
 - SecToNano, 794
 - Set, 794, 795
 - SetToWallTime, 795
 - Sleep, 795
 - Time, 779
 - Zero, 796
- gazebo::common::Timer, 796
 - ~Timer, 797
 - GetElapsed, 798
 - GetRunning, 798
 - operator<<, 798
 - Start, 798
 - Stop, 798
 - Timer, 797
- gazebo::common::UpdateInfo, 810
 - realTime, 811
 - simTime, 811
 - worldName, 811
- gazebo::common::Video, 861
 - ~Video, 862
 - GetHeight, 862
 - GetNextFrame, 862
 - GetWidth, 862
 - Load, 862
 - Video, 862
- gazebo::event, 86
 - Connection_V, 87
 - ConnectionPtr, 87
- gazebo::event::Connection, 218
 - ~Connection, 218
 - Connection, 218
 - GetId, 219
- gazebo::event::Event, 288
 - ~Event, 290
 - Disconnect, 290
- gazebo::event::EventT
 - operator(), 308–310
 - Signal, 311–313
- gazebo::event::EventT< T >, 305
- gazebo::event::Events, 293
 - addEntity, 303
 - ConnectAddEntity, 295
 - ConnectCreateEntity, 296
 - ConnectDeleteEntity, 296
 - ConnectDiagTimerStart, 296
 - ConnectDiagTimerStop, 297
 - ConnectPause, 297
 - ConnectPostRender, 297
 - ConnectPreRender, 297
 - ConnectRender, 298
 - ConnectSetSelectedEntity, 298
 - ConnectStep, 298
 - ConnectStop, 299
 - ConnectWorldCreated, 299
 - ConnectWorldUpdateBegin, 299
 - ConnectWorldUpdateEnd, 299
 - ConnectWorldUpdateStart, 300
 - deleteEntity, 303
 - diagTimerStart, 303
 - diagTimerStop, 304
 - DisconnectAddEntity, 300
 - DisconnectCreateEntity, 300
 - DisconnectDeleteEntity, 300
 - DisconnectDiagTimerStart, 301
 - DisconnectDiagTimerStop, 301
 - DisconnectPause, 301
 - DisconnectPostRender, 301
 - DisconnectPreRender, 301
 - DisconnectRender, 302
 - DisconnectSetSelectedEntity, 302
 - DisconnectStep, 302
 - DisconnectStop, 302
 - DisconnectWorldCreated, 302
 - DisconnectWorldUpdateBegin, 303
 - DisconnectWorldUpdateEnd, 303
 - DisconnectWorldUpdateStart, 303

- entityCreated, 304
- pause, 304
- postRender, 304
- preRender, 304
- render, 304
- setSelectedEntity, 304
- step, 304
- stop, 304
- worldCreated, 305
- worldUpdateBegin, 305
- worldUpdateEnd, 305
- worldUpdateStart, 305
- gazebo::math, 87
 - GeneratorType, 89
 - NRealGen, 89
 - NormalRealDist, 89
 - UIntGen, 89
 - URealGen, 89
 - UniformIntDist, 89
 - UniformRealDist, 89
- gazebo::math::Angle, 116
 - ~Angle, 118
 - Angle, 117
 - Degree, 118
 - Normalize, 118
 - operator<, 120
 - operator<<, 122
 - operator<=, 121
 - operator>, 121
 - operator>>, 122
 - operator>=, 121
 - operator*, 118
 - operator*=:, 119
 - operator+, 119
 - operator+=, 119
 - operator-, 119
 - operator-=, 120
 - operator/, 120
 - operator/=, 120
 - operator==, 121
 - Radian, 122
 - SetFromDegree, 122
 - SetFromRadian, 122
- gazebo::math::Box, 146
 - ~Box, 148
 - Box, 148
 - GetCenter, 148
 - GetSize, 148
 - GetXLength, 148
 - GetYLength, 149
 - GetZLength, 149
 - max, 151
 - Merge, 149
 - min, 151
 - operator<<, 150
 - operator+, 149
 - operator+=, 149
 - operator-, 150
 - operator=, 150
 - operator==, 150
- gazebo::math::Matrix3, 456
 - ~Matrix3, 457
 - m, 459
 - Matrix3, 457
 - operator<<, 459
 - operator==, 457
 - operator[], 457, 458
 - SetCol, 458
 - SetFromAxes, 458
 - SetFromAxis, 458
- gazebo::math::Matrix4, 459
 - ~Matrix4, 461
 - GetAsPose, 462
 - GetEulerRotation, 462
 - GetRotation, 462
 - GetTranslation, 462
 - IDENTITY, 466
 - Inverse, 462
 - IsAffine, 462
 - m, 466
 - Matrix4, 461
 - operator<<, 466
 - operator*, 462, 463
 - operator=, 463
 - operator==, 464
 - operator[], 464
 - Set, 464
 - SetScale, 465
 - SetTranslate, 465
 - TransformAffine, 465
 - ZERO, 466
- gazebo::math::Plane, 575
 - ~Plane, 576
 - d, 577
 - Distance, 577
 - normal, 577
 - operator=, 577
 - Plane, 576
 - Set, 577
 - size, 578
- gazebo::math::Pose, 584
 - ~Pose, 587
 - CoordPoseSolve, 587
 - CoordPositionAdd, 587
 - CoordPositionSub, 588
 - CoordRotationAdd, 588
 - CoordRotationSub, 588
 - Correct, 589

- GetInverse, 589
- IsFinite, 589
- operator<<, 592
- operator>>, 592
- operator*, 589
- operator+, 589
- operator+=", 590
- operator-, 590
- operator-=, 590
- operator==, 591
- pos, 592
- Pose, 586, 587
- Reset, 591
- rot, 593
- RotatePositionAboutOrigin, 591
- Round, 591
- Set, 591, 592
- Zero, 593
- gazebo::math::Quaternion, 609
 - ~Quaternion, 613
 - Correct, 614
 - Dot, 614
 - EulerToQuaternion, 614
 - GetAsAxis, 614
 - GetAsEuler, 615
 - GetAsMatrix3, 615
 - GetAsMatrix4, 615
 - GetExp, 615
 - GetInverse, 615
 - GetLog, 615
 - GetPitch, 615
 - GetRoll, 616
 - GetXAxis, 616
 - GetYAxis, 616
 - GetYaw, 616
 - GetZAxis, 616
 - Invert, 616
 - IsFinite, 617
 - Normalize, 617
 - operator<<, 622
 - operator>>, 622
 - operator*, 617, 618
 - operator*=", 618
 - operator+, 618
 - operator+=", 618
 - operator-, 619
 - operator-=, 619
 - operator=, 619
 - operator==, 619
 - Quaternion, 612, 613
 - RotateVector, 620
 - RotateVectorReverse, 620
 - Round, 620
 - Scale, 620
 - Set, 620
 - SetFromAxis, 621
 - SetFromEuler, 621
 - SetToIdentity, 621
 - Slerp, 622
 - Squad, 622
 - w, 623
 - x, 623
 - y, 623
 - z, 623
- gazebo::math::Rand, 623
 - GetDblNormal, 624
 - GetDblUniform, 624
 - GetIntNormal, 624
 - GetIntUniform, 624
 - GetSeed, 625
 - SetSeed, 625
- gazebo::math::RotationSpline, 654
 - ~RotationSpline, 655
 - AddPoint, 655
 - autoCalc, 657
 - Clear, 655
 - GetNumPoints, 655
 - GetPoint, 655
 - Interpolate, 656
 - points, 657
 - RecalcTangents, 656
 - RotationSpline, 655
 - SetAutoCalculate, 657
 - tangents, 657
 - UpdatePoint, 657
- gazebo::math::Spline, 739
 - ~Spline, 740
 - AddPoint, 740
 - autoCalc, 743
 - Clear, 740
 - coeffs, 743
 - GetPoint, 740
 - GetPointCount, 741
 - GetTangent, 741
 - GetTension, 741
 - Interpolate, 741
 - points, 743
 - RecalcTangents, 741
 - SetAutoCalculate, 742
 - SetTension, 742
 - Spline, 740
 - tangents, 743
 - tension, 743
 - UpdatePoint, 742
- gazebo::math::Vector2d, 821
 - ~Vector2d, 823
 - Cross, 823
 - Distance, 824

- IsFinite, 824
- Normalize, 824
- operator<<, 829
- operator>>, 829
- operator*, 824
- operator*=: 825
- operator+, 825
- operator+=, 826
- operator-, 826
- operator-=, 826
- operator/, 826, 827
- operator/=, 827
- operator=, 827, 828
- operator==, 828
- operator[], 828
- Set, 828
- Vector2d, 823
- x, 829
- y, 829
- gazebo::math::Vector2i, 829
 - ~Vector2i, 832
 - Cross, 832
 - Distance, 832
 - IsFinite, 832
 - Normalize, 832
 - operator<<, 837
 - operator>>, 838
 - operator*, 833
 - operator*=: 833, 834
 - operator+, 834
 - operator+=, 834
 - operator-, 834
 - operator-=, 835
 - operator/, 835
 - operator/=, 835, 836
 - operator=, 836
 - operator==, 837
 - operator[], 837
 - Set, 837
 - Vector2i, 831
 - x, 838
 - y, 838
- gazebo::math::Vector3, 838
 - ~Vector3, 841
 - Correct, 842
 - Cross, 842
 - Distance, 842
 - Dot, 843
 - Equal, 843
 - GetAbs, 843
 - GetDistToLine, 843
 - GetLength, 843
 - GetMax, 844
 - GetMin, 844
 - GetNormal, 844
 - GetPerpendicular, 844
 - GetRounded, 844
 - GetSquaredLength, 844
 - GetSum, 845
 - IsFinite, 845
 - Normalize, 845
 - operator<<, 850
 - operator>>, 851
 - operator*, 845, 846, 850
 - operator*=: 846
 - operator+, 846
 - operator+=, 847
 - operator-, 847
 - operator-=, 847
 - operator/, 847, 848
 - operator/=, 848
 - operator=, 848, 849
 - operator==, 849
 - operator[], 849
 - Round, 849
 - Set, 850
 - SetToMax, 850
 - SetToMin, 850
 - Vector3, 841
 - x, 851
 - y, 851
 - z, 851
 - Zero, 851
- gazebo::math::Vector4, 852
 - ~Vector4, 854
 - Distance, 854
 - GetLength, 854
 - GetSquaredLength, 854
 - IsFinite, 855
 - Normalize, 855
 - operator<<, 860
 - operator>>, 860
 - operator*, 855, 856
 - operator*=: 856
 - operator+, 857
 - operator+=, 857
 - operator-, 857
 - operator-=, 857
 - operator/, 858
 - operator/=, 858, 859
 - operator=, 859
 - operator==, 859
 - operator[], 859
 - Set, 860
 - Vector4, 854
 - w, 860
 - x, 861
 - y, 861

- z, 861
- gazebo::msgs, 89
 - MsgFactoryFn, 91
- gazebo::msgs::MsgFactory, 512
 - GetMsgTypes, 512
 - NewMsg, 513
 - RegisterMsg, 513
- gazebo::physics, 91
 - Actor_V, 95
 - ActorPtr, 95
 - Base_V, 95
 - BasePtr, 95
 - BoxShapePtr, 95
 - Collision_V, 95
 - CollisionPtr, 95
 - ContactPtr, 95
 - CylinderShapePtr, 95
 - EntityPtr, 95
 - HeightmapShapePtr, 95
 - InertialPtr, 95
 - Joint_V, 95
 - JointController_V, 95
 - JointControllerPtr, 95
 - JointPtr, 95
 - Link_V, 95
 - LinkPtr, 95
 - MeshShapePtr, 95
 - Model_V, 95
 - ModelPtr, 96
 - MultiRayShapePtr, 96
 - PhysicsEnginePtr, 96
 - RayShapePtr, 96
 - RoadPtr, 96
 - ShapePtr, 96
 - SphereShapePtr, 96
 - SurfaceParamsPtr, 96
 - WorldPtr, 96
- gazebo::physics::Actor, 109
 - ~Actor, 112
 - active, 114
 - Actor, 112
 - autoStart, 114
 - bonePosePub, 114
 - Fini, 112
 - GetSDF, 112
 - Init, 113
 - interpolateX, 114
 - IsActive, 113
 - lastPos, 114
 - lastScriptTime, 114
 - lastTraj, 114
 - Load, 113
 - loop, 114
 - mainLink, 114
 - mesh, 114
 - oldAction, 114
 - pathLength, 114
 - Play, 113
 - playStartTime, 115
 - prevFrameTime, 115
 - scriptLength, 115
 - skelAnimation, 115
 - skelNodesMap, 115
 - skeleton, 115
 - skinFile, 115
 - skinScale, 115
 - startDelay, 115
 - Stop, 113
 - trajInfo, 115
 - trajectories, 115
 - Update, 113
 - UpdateParameters, 113
 - visualName, 116
- gazebo::physics::BallJoint
 - ~BallJoint, 135
 - BallJoint, 134
 - GetAngleCount, 135
 - GetHighStop, 135
 - GetLowStop, 135
 - Load, 135
 - SetAxis, 135
 - SetHighStop, 135
 - SetLowStop, 135
- gazebo::physics::BallJoint< T >, 133
- gazebo::physics::Base, 135
 - ~Base, 139
 - ACTOR, 138
 - AddChild, 139
 - AddType, 139
 - BALL_JOINT, 139
 - BASE, 138
 - BOX_SHAPE, 139
 - Base, 139
 - COLLISION, 138
 - CYLINDER_SHAPE, 139
 - children, 146
 - childrenEnd, 146
 - ENTITY, 138
 - EntityType, 138
 - Fini, 140
 - GetById, 140
 - GetByName, 140
 - GetChild, 140, 141
 - GetChildCount, 141
 - GetId, 141
 - GetName, 141
 - GetParent, 141
 - GetParentId, 141

- GetSDF, 142
- GetSaveable, 142
- GetScopedName, 142
- GetType, 142
- GetWorld, 142
- HEIGHTMAP_SHAPE, 139
- HINGE2_JOINT, 139
- HINGE_JOINT, 139
- HasType, 142
- Init, 143
- IsSelected, 143
- JOINT, 139
- LIGHT, 139
- LINK, 138
- Load, 143
- MAP_SHAPE, 139
- MODEL, 138
- MULTIRAY_SHAPE, 139
- operator==, 143
- PLANE_SHAPE, 139
- parent, 146
- Print, 144
- RAY_SHAPE, 139
- RemoveChild, 144
- RemoveChildren, 144
- Reset, 144
- SCREW_JOINT, 139
- SHAPE, 139
- SLIDER_JOINT, 139
- SPHERE_SHAPE, 139
- sdf, 146
- SetName, 145
- SetParent, 145
- SetSaveable, 145
- SetSelected, 145
- SetWorld, 145
- TRIMESH_SHAPE, 139
- UNIVERSAL_JOINT, 139
- Update, 145
- UpdateParameters, 146
- VISUAL, 139
- world, 146
- gazebo::physics::BoxShape, 151
 - ~BoxShape, 153
 - BoxShape, 153
 - FillMsg, 153
 - GetSize, 153
 - Init, 153
 - ProcessMsg, 153
 - SetSize, 153
- gazebo::physics::Collision, 192
 - ~Collision, 194
 - AddContact, 195
 - Collision, 194
 - ConnectContact, 195
 - DisconnectContact, 195
 - FillMsg, 195
 - Fini, 195
 - GetBoundingBox, 195
 - GetContactsEnabled, 195
 - GetLaserRetro, 196
 - GetLink, 196
 - GetModel, 196
 - GetRelativeAngularAccel, 196
 - GetRelativeAngularVel, 196
 - GetRelativeLinearAccel, 196
 - GetRelativeLinearVel, 197
 - GetShape, 197
 - GetShapeType, 197
 - GetState, 197
 - GetSurface, 197
 - GetWorldAngularAccel, 198
 - GetWorldAngularVel, 198
 - GetWorldLinearAccel, 198
 - GetWorldLinearVel, 198
 - Init, 198
 - IsPlaceable, 198
 - link, 200
 - Load, 199
 - placeable, 201
 - ProcessMsg, 199
 - SetCategoryBits, 199
 - SetCollideBits, 199
 - SetCollision, 199
 - SetContactsEnabled, 200
 - SetLaserRetro, 200
 - SetShape, 200
 - SetState, 200
 - shape, 201
 - UpdateParameters, 200
- gazebo::physics::CollisionState, 201
 - ~CollisionState, 203
 - CollisionState, 202
 - FillSDF, 203
 - GetPose, 203
 - IsZero, 203
 - Load, 203
 - operator<<, 204
 - operator+, 203
 - operator-, 204
 - operator=, 204
- gazebo::physics::Contact, 232
 - ~Contact, 233
 - collision1, 234
 - collision2, 234
 - Contact, 233
 - count, 234
 - DebugString, 233

- depths, 234
- FillMsg, 233
- normals, 234
- operator=, 233, 234
- positions, 235
- Reset, 234
- time, 235
- wrench, 235
- gazebo::physics::ContactManager, 235
 - ~ContactManager, 236
 - Clear, 236
 - ContactManager, 236
 - GetContact, 236
 - GetContactCount, 236
 - GetContacts, 236
 - Init, 236
 - NewContact, 237
 - PublishContacts, 237
 - ResetCount, 237
- gazebo::physics::CylinderShape, 246
 - ~CylinderShape, 248
 - CylinderShape, 248
 - FillMsg, 248
 - GetLength, 248
 - GetRadius, 248
 - Init, 248
 - ProcessMsg, 249
 - SetLength, 249
 - SetRadius, 249
 - SetSize, 249
- gazebo::physics::Entity, 277
 - ~Entity, 280
 - animation, 287
 - animationConnection, 287
 - animationStartPose, 287
 - connections, 287
 - dirtyPose, 288
 - Entity, 280
 - Fini, 280
 - GetBoundingBox, 280
 - GetChildCollision, 280
 - GetChildLink, 281
 - GetCollisionBoundingBox, 281
 - GetDirtyPose, 281
 - GetInitialRelativePose, 281
 - GetNearestEntityBelow, 281
 - GetParentModel, 282
 - GetRelativeAngularAccel, 282
 - GetRelativeAngularVel, 282
 - GetRelativeLinearAccel, 282
 - GetRelativeLinearVel, 282
 - GetRelativePose, 283
 - GetWorldAngularAccel, 283
 - GetWorldAngularVel, 283
 - GetWorldLinearAccel, 283
 - GetWorldLinearVel, 283
 - GetWorldPose, 284
 - IsCanonicalLink, 284
 - IsStatic, 284
 - Load, 284
 - node, 288
 - OnPoseChange, 284
 - parentEntity, 288
 - PlaceOnEntity, 285
 - PlaceOnNearestEntityBelow, 285
 - poseMsg, 288
 - prevAnimationTime, 288
 - requestPub, 288
 - Reset, 285
 - SetAnimation, 285
 - SetCanonicalLink, 285
 - SetInitialRelativePose, 286
 - SetName, 286
 - SetRelativePose, 286
 - SetStatic, 286
 - SetWorldPose, 286
 - SetWorldTwist, 287
 - StopAnimation, 287
 - UpdateParameters, 287
 - visPub, 288
 - visualMsg, 288
- gazebo::physics::Gripper, 340
 - ~Gripper, 340
 - Gripper, 340
 - Init, 341
 - Load, 341
- gazebo::physics::HeightmapShape, 348
 - ~HeightmapShape, 350
 - FillMsg, 350
 - GetHeight, 350
 - GetMaxHeight, 350
 - GetMinHeight, 350
 - GetPos, 350
 - GetSize, 351
 - GetSubSampling, 351
 - GetURI, 351
 - GetVertexCount, 351
 - HeightmapShape, 349
 - heights, 352
 - img, 352
 - Init, 351
 - Load, 351
 - ProcessMsg, 352
 - scale, 352
 - subSampling, 352
 - vertSize, 352
- gazebo::physics::Hinge2Joint
 - ~Hinge2Joint, 353

- GetAngleCount, 354
- Hinge2Joint, 353
- Load, 354
- gazebo::physics::Hinge2Joint< T >, 352
- gazebo::physics::HingeJoint
 - ~HingeJoint, 355
 - GetAngleCount, 355
 - HingeJoint, 355
 - Init, 355
 - Load, 355
- gazebo::physics::HingeJoint< T >, 354
- gazebo::physics::Inertial, 365
 - ~Inertial, 367
 - GetCoG, 367
 - GetIXX, 367
 - GetIXY, 367
 - GetIXZ, 367
 - GetIYY, 368
 - GetIYZ, 368
 - GetIZZ, 368
 - GetMass, 368
 - GetPose, 368
 - GetPrincipalMoments, 368
 - GetProductsofInertia, 368
 - Inertial, 366, 367
 - Load, 369
 - operator<<, 372
 - operator+, 369
 - operator+=", 369
 - operator=, 369
 - ProcessMsg, 370
 - Reset, 370
 - Rotate, 370
 - SetCoG, 370
 - SetIXX, 371
 - SetIXY, 371
 - SetIXZ, 371
 - SetIYY, 371
 - SetIYZ, 371
 - SetIZZ, 371
 - SetInertiaMatrix, 370
 - SetMass, 372
 - UpdateParameters, 372
- gazebo::physics::Joint, 375
 - ~Joint, 380
 - anchorLink, 391
 - anchorPos, 391
 - anchorPose, 392
 - ApplyDamping, 380
 - applyDamping, 392
 - AreConnected, 380
 - Attach, 380
 - Attribute, 379
 - CFM, 380
 - childLink, 392
 - ConnectJointUpdate, 381
 - dampingCoefficient, 392
 - Detach, 381
 - DisconnectJointUpdate, 381
 - ERP, 380
 - effortLimit, 392
 - FMAX, 380
 - FUDGE_FACTOR, 379
 - FillMsg, 381
 - forceApplied, 392
 - GetAnchor, 381
 - GetAngle, 382
 - GetAngleCount, 382
 - GetAngleImpl, 382
 - GetAttribute, 382
 - GetChild, 382
 - GetEffortLimit, 383
 - GetForce, 383
 - GetForceTorque, 383, 384
 - GetGlobalAxis, 384
 - GetHighStop, 384
 - GetInertiaRatio, 385
 - GetJointLink, 385
 - GetLinkForce, 385
 - GetLinkTorque, 385
 - GetLocalAxis, 386
 - GetLowStop, 386
 - GetLowerLimit, 386
 - GetMaxForce, 386
 - GetParent, 387
 - GetUpperLimit, 387
 - GetVelocity, 387
 - GetVelocityLimit, 387
 - HI_STOP, 380
 - inertiaRatio, 392
 - Init, 388
 - Joint, 380
 - LO_STOP, 380
 - Load, 388
 - lowerLimit, 392
 - model, 392
 - parentLink, 392
 - Reset, 388
 - STOP_CFM, 380
 - STOP_ERP, 380
 - SUSPENSION_CFM, 379
 - SUSPENSION_ERP, 379
 - SetAnchor, 389
 - SetAngle, 389
 - SetAttribute, 389
 - SetAxis, 389
 - SetDamping, 389
 - SetForce, 390

- SetHighStop, 390
- SetLowStop, 390
- SetMaxForce, 390
- SetModel, 390
- SetState, 391
- SetVelocity, 391
- Update, 391
- UpdateParameters, 391
- upperLimit, 393
- useCFMDamping, 393
- VEL, 380
- velocityLimit, 393
- gazebo::physics::JointController, 393
 - AddJoint, 394
 - JointController, 394
 - Reset, 394
 - SetJointPosition, 394
 - SetJointPositions, 394
 - Update, 395
- gazebo::physics::JointState, 395
 - ~JointState, 397
 - FillSDF, 397
 - GetAngle, 397
 - GetAngleCount, 397
 - GetAngles, 397
 - IsZero, 397
 - JointState, 396
 - Load, 398
 - operator<<, 399
 - operator+, 398
 - operator-, 398
 - operator=, 398
- gazebo::physics::JointWrench, 400
 - body1Force, 401
 - body1Torque, 401
 - body2Force, 402
 - body2Torque, 402
 - operator=, 401
- gazebo::physics::Link, 411
 - ~Link, 416
 - AddChildJoint, 416
 - AddForce, 416
 - AddForceAtRelativePosition, 417
 - AddForceAtWorldPosition, 417
 - AddParentJoint, 417
 - AddRelativeForce, 417
 - AddRelativeTorque, 417
 - AddTorque, 418
 - angularAccel, 430
 - AttachStaticModel, 418
 - attachedModelsOffset, 430
 - cgVisuals, 430
 - ConnectEnabled, 418
 - DetachAllStaticModels, 418
 - DetachStaticModel, 418
 - DisconnectEnabled, 418
 - FillMsg, 419
 - Fini, 419
 - GetAngularDamping, 419
 - GetBoundingBox, 419
 - GetChildJointsLinks, 419
 - GetCollision, 419, 420
 - GetCollisionById, 420
 - GetCollisions, 420
 - GetEnabled, 420
 - GetGravityMode, 421
 - GetInertial, 421
 - GetKinematic, 421
 - GetLinearDamping, 421
 - GetModel, 421
 - GetParentJointsLinks, 421
 - GetRelativeAngularAccel, 422
 - GetRelativeAngularVel, 422
 - GetRelativeForce, 422
 - GetRelativeLinearAccel, 422
 - GetRelativeLinearVel, 422
 - GetRelativeTorque, 422
 - GetSelfCollide, 423
 - GetSensorCount, 423
 - GetSensorName, 423
 - GetWorldAngularAccel, 423
 - GetWorldCoGLinearVel, 424
 - GetWorldCoGPose, 424
 - GetWorldForce, 424
 - GetWorldLinearAccel, 424
 - GetWorldLinearVel, 424, 425
 - GetWorldTorque, 425
 - inertial, 430
 - Init, 425
 - linearAccel, 430
 - Link, 416
 - Load, 425
 - OnPoseChange, 425
 - ProcessMsg, 425
 - RemoveChildJoint, 426
 - RemoveParentJoint, 426
 - Reset, 426
 - ResetPhysicsStates, 426
 - SetAngularAccel, 426
 - SetAngularDamping, 426
 - SetAngularVel, 427
 - SetAutoDisable, 427
 - SetCollideMode, 427
 - SetEnabled, 427
 - SetForce, 427
 - SetGravityMode, 427
 - SetInertial, 428
 - SetKinematic, 428

- SetLaserRetro, 428
- SetLinearAccel, 428
- SetLinearDamping, 428
- SetLinearVel, 428
- SetSelected, 429
- SetSelfCollide, 429
- SetState, 429
- SetTorque, 429
- Update, 429
- UpdateMass, 429
- UpdateParameters, 430
- UpdateSurface, 430
- visuals, 430
- gazebo::physics::LinkState, 431
 - ~LinkState, 433
 - FillSDF, 433
 - GetAcceleration, 433
 - GetCollisionState, 433
 - GetCollisionStateCount, 434
 - GetCollisionStates, 434
 - GetPose, 434
 - GetVelocity, 434
 - GetWrench, 434
 - IsZero, 435
 - LinkState, 432
 - Load, 435
 - operator<<, 436
 - operator+, 435
 - operator-, 435
 - operator=, 436
- gazebo::physics::Model, 480
 - ~Model, 483
 - AttachStaticModel, 483
 - attachedModels, 493
 - attachedModelsOffset, 493
 - DetachStaticModel, 484
 - FillMsg, 484
 - Fini, 484
 - GetAutoDisable, 484
 - GetBoundingBox, 484
 - GetJoint, 485
 - GetJointController, 485
 - GetJointCount, 485
 - GetJoints, 485
 - GetLink, 485
 - GetLinkById, 486
 - GetLinks, 486
 - GetPluginCount, 486
 - GetRelativeAngularAccel, 486
 - GetRelativeAngularVel, 486
 - GetRelativeLinearAccel, 487
 - GetRelativeLinearVel, 487
 - GetSDF, 487
 - GetSensorCount, 487
 - GetWorldAngularAccel, 487
 - GetWorldAngularVel, 488
 - GetWorldLinearAccel, 488
 - GetWorldLinearVel, 488
 - Init, 488
 - Load, 488
 - LoadJoints, 489
 - LoadPlugins, 489
 - Model, 483
 - OnPoseChange, 489
 - ProcessMsg, 489
 - RemoveChild, 489
 - Reset, 489
 - SetAngularAccel, 489
 - SetAngularVel, 490
 - SetAutoDisable, 490
 - SetCollideMode, 490
 - SetEnabled, 490
 - SetGravityMode, 490
 - SetJointAnimation, 490
 - SetJointPosition, 491
 - SetJointPositions, 491
 - SetLaserRetro, 491
 - SetLinearAccel, 491
 - SetLinearVel, 492
 - SetLinkWorldPose, 492
 - SetState, 492
 - StopAnimation, 492
 - Update, 492
 - UpdateParameters, 493
- gazebo::physics::ModelState, 496
 - ~ModelState, 499
 - FillSDF, 499
 - GetJointState, 499
 - GetJointStateCount, 500
 - GetJointStates, 500
 - GetLinkState, 500
 - GetLinkStateCount, 501
 - GetLinkStates, 501
 - GetPose, 501
 - HasJointState, 501
 - HasLinkState, 501
 - IsZero, 502
 - Load, 502
 - ModelState, 498
 - operator<<, 503
 - operator+, 502
 - operator-, 502
 - operator=, 503
- gazebo::physics::MultiRayShape, 518
 - ~MultiRayShape, 521
 - AddRay, 521
 - ConnectNewLaserScans, 521
 - DisconnectNewLaserScans, 522

- FillMsg, 522
- GetFiducial, 522
- GetMaxAngle, 522
- GetMaxRange, 522
- GetMinAngle, 523
- GetMinRange, 523
- GetRange, 523
- GetResRange, 523
- GetRetro, 523
- GetSampleCount, 524
- GetScanResolution, 524
- GetVerticalMaxAngle, 524
- GetVerticalMinAngle, 524
- GetVerticalSampleCount, 524
- GetVerticalScanResolution, 524
- horzElem, 525
- Init, 524
- MultiRayShape, 521
- newLaserScans, 525
- offset, 525
- ProcessMsg, 525
- rangeElem, 525
- rayElem, 525
- rays, 526
- scanElem, 526
- Update, 525
- UpdateRays, 525
- vertElem, 526
- gazebo::physics::PhysicsEngine, 558
 - ~PhysicsEngine, 561
 - contactManager, 569
 - CreateCollision, 561
 - CreateJoint, 562
 - CreateLink, 562
 - CreateShape, 562
 - DebugPrint, 562
 - Fini, 562
 - GetAutoDisableFlag, 562
 - GetContactManager, 563
 - GetContactMaxCorrectingVel, 563
 - GetContactSurfaceLayer, 563
 - GetGravity, 563
 - GetMaxContacts, 563
 - GetPhysicsUpdateMutex, 563
 - GetSORPGSIlters, 564
 - GetSORPGSPreconIlters, 564
 - GetSORPGSW, 564
 - GetStepTime, 564
 - GetType, 564
 - GetUpdatePeriod, 565
 - GetUpdateRate, 565
 - GetWorldCFM, 565
 - GetWorldERP, 565
 - Init, 565
 - InitForThread, 565
 - Load, 566
 - node, 569
 - OnPhysicsMsg, 566
 - OnRequest, 566
 - PhysicsEngine, 561
 - physicsSub, 569
 - physicsUpdateMutex, 569
 - requestSub, 569
 - Reset, 566
 - responsePub, 569
 - sdf, 569
 - SetAutoDisableFlag, 566
 - SetContactMaxCorrectingVel, 566
 - SetContactSurfaceLayer, 567
 - SetGravity, 567
 - SetMaxContacts, 567
 - SetSORPGSIlters, 567
 - SetSORPGSPreconIlters, 567
 - SetSORPGSW, 568
 - SetSeed, 567
 - SetStepTime, 568
 - SetUpdateRate, 568
 - SetWorldCFM, 568
 - SetWorldERP, 568
 - UpdateCollision, 569
 - UpdatePhysics, 569
 - world, 569
- gazebo::physics::PhysicsFactory, 570
 - IsRegistered, 570
 - NewPhysicsEngine, 570
 - RegisterAll, 571
 - RegisterPhysicsEngine, 571
- gazebo::physics::PlaneShape, 578
 - ~PlaneShape, 579
 - CreatePlane, 579
 - FillMsg, 579
 - GetNormal, 580
 - GetSize, 580
 - Init, 580
 - PlaneShape, 579
 - ProcessMsg, 580
 - SetAltitude, 580
 - SetNormal, 580
 - SetSize, 581
- gazebo::physics::RayShape, 634
 - ~RayShape, 636
 - contactFiducial, 639
 - contactLen, 639
 - contactRetro, 639
 - FillMsg, 636
 - GetFiducial, 636
 - GetGlobalPoints, 636
 - GetIntersection, 637

- GetLength, 637
- GetRelativePoints, 637
- GetRetro, 637
- globalEndPos, 639
- globalStartPos, 639
- Init, 637
- ProcessMsg, 638
- RayShape, 636
- relativeEndPos, 639
- relativeStartPos, 639
- SetFiducial, 638
- SetLength, 638
- SetPoints, 638
- SetRetro, 638
- Update, 638
- gazebo::physics::Road, 651
 - ~Road, 652
 - Init, 653
 - Load, 653
 - Road, 652
- gazebo::physics::ScrewJoint
 - ~ScrewJoint, 678
 - fakeAnchor, 680
 - GetAnchor, 678
 - GetAngleCount, 679
 - GetThreadPitch, 679
 - Load, 679
 - ScrewJoint, 678
 - SetAnchor, 679
 - SetThreadPitch, 679
 - threadPitch, 680
- gazebo::physics::ScrewJoint< T >, 677
- gazebo::physics::Shape, 706
 - ~Shape, 707
 - collisionParent, 708
 - FillMsg, 707
 - Init, 707
 - ProcessMsg, 707
 - Shape, 707
- gazebo::physics::SliderJoint
 - ~SliderJoint, 733
 - fakeAnchor, 734
 - GetAnchor, 733
 - GetAngleCount, 734
 - Load, 734
 - SetAnchor, 734
 - SliderJoint, 733
- gazebo::physics::SliderJoint< T >, 732
- gazebo::physics::SphereShape, 736
 - ~SphereShape, 738
 - FillMsg, 738
 - GetRadius, 738
 - Init, 738
 - ProcessMsg, 738
 - SetRadius, 738
 - SphereShape, 738
- gazebo::physics::State, 743
 - ~State, 745
 - GetName, 745
 - GetRealTime, 745
 - GetSimTime, 746
 - GetWallTime, 746
 - Load, 746
 - name, 747
 - operator-, 746
 - operator=, 746
 - realTime, 747
 - SetName, 747
 - simTime, 747
 - State, 745
 - wallTime, 747
- gazebo::physics::SurfaceParams, 764
 - ~SurfaceParams, 765
 - bounce, 765
 - bounceThreshold, 766
 - cfm, 766
 - erp, 766
 - fdir1, 766
 - FillMsg, 765
 - kd, 766
 - kp, 766
 - Load, 765
 - maxVel, 767
 - minDepth, 767
 - mu1, 767
 - mu2, 767
 - ProcessMsg, 765
 - slip1, 767
 - slip2, 768
 - SurfaceParams, 765
- gazebo::physics::TrajectoryInfo, 805
 - duration, 805
 - endTime, 805
 - id, 805
 - startTime, 805
 - translated, 805
 - type, 805
- gazebo::physics::TrimeshShape, 806
 - ~TrimeshShape, 807
 - FillMsg, 807
 - GetFilename, 808
 - GetSize, 808
 - Init, 808
 - mesh, 809
 - ProcessMsg, 808
 - SetFilename, 808
 - SetScale, 808
 - TrimeshShape, 807

- Update, 808
- gazebo::physics::UniversalJoint
 - ~UniversalJoint, 810
 - GetAngleCount, 810
 - Load, 810
 - UniversalJoint, 810
- gazebo::physics::UniversalJoint< T >, 809
- gazebo::physics::World, 892
 - ~World, 895
 - Clear, 896
 - dirtyPoses, 903
 - DisableAllModels, 896
 - EnableAllModels, 896
 - EnablePhysicsEngine, 896
 - Fini, 896
 - GetByName, 896
 - GetEnablePhysicsEngine, 896
 - GetEntity, 897
 - GetEntityBelowPoint, 897
 - GetModel, 897
 - GetModelBelowPoint, 898
 - GetModelCount, 898
 - GetModels, 898
 - GetName, 898
 - GetPauseTime, 898
 - GetPhysicsEngine, 899
 - GetRealTime, 899
 - GetSelectedEntity, 899
 - GetSetWorldPoseMutex, 899
 - GetSimTime, 899
 - GetStartTime, 899
 - Init, 900
 - InsertModelFile, 900
 - InsertModelSDF, 900
 - InsertModelString, 900
 - IsLoaded, 900
 - IsPaused, 900
 - Load, 901
 - LoadPlugin, 901
 - PrintEntityTree, 901
 - PublishModelPose, 901
 - RemovePlugin, 901
 - Reset, 902
 - ResetEntities, 902
 - ResetTime, 902
 - Run, 902
 - Save, 902
 - SetPaused, 902
 - SetSimTime, 902
 - SetState, 903
 - StepWorld, 903
 - Stop, 903
 - StripWorldName, 903
 - UpdateStateSDF, 903
 - World, 895
- gazebo::physics::WorldState, 905
 - ~WorldState, 907
 - FillSDF, 907
 - GetModelState, 907, 908
 - GetModelStateCount, 908
 - GetModelStates, 908
 - HasModelState, 908
 - IsZero, 908
 - Load, 909
 - operator<<, 910
 - operator+, 909
 - operator-, 909
 - operator=, 909
 - SetWorld, 910
 - WorldState, 907
- gazebo::rendering, 96
 - ArrowVisualPtr, 99
 - AxisVisualPtr, 99
 - COMVisualPtr, 99
 - CameraPtr, 99
 - CameraVisualPtr, 99
 - ContactVisualPtr, 99
 - DepthCameraPtr, 99
 - DynamicLinesPtr, 99
 - GpuLaserPtr, 99
 - JointVisualPtr, 99
 - LaserVisualPtr, 99
 - LightPtr, 99
 - RENDERING_LINE_LIST, 99
 - RENDERING_LINE_STRIP, 99
 - RENDERING_MESH_RESOURCE, 99
 - RENDERING_POINT_LIST, 99
 - RENDERING_TRIANGLE_FAN, 99
 - RENDERING_TRIANGLE_LIST, 99
 - RENDERING_TRIANGLE_STRIP, 99
 - RFIDTagVisualPtr, 99
 - RFIDVisualPtr, 99
 - RenderOpType, 99
 - ScenePtr, 99
 - UserCameraPtr, 99
 - VisualPtr, 99
- gazebo::rendering::ArrowVisual, 127
 - ~ArrowVisual, 129
 - ArrowVisual, 128
 - Load, 129
 - ShowRotation, 129
- gazebo::rendering::AxisVisual, 131
 - ~AxisVisual, 132
 - AxisVisual, 132
 - Load, 132
 - ScaleXAxis, 132
 - ScaleYAxis, 132
 - ScaleZAxis, 133

- SetAxisMaterial, 133
- ShowRotation, 133
- gazebo::rendering::COMVisual, 216
 - ~COMVisual, 217
 - COMVisual, 217
 - Load, 217
- gazebo::rendering::Camera, 159
 - ~Camera, 166
 - animState, 182
 - AnimationComplete, 166
 - AttachToVisual, 166
 - AttachToVisualImpl, 166, 167
 - bayerFrameBuffer, 182
 - Camera, 165
 - camera, 182
 - captureData, 182
 - ConnectNewImageFrame, 167
 - connections, 182
 - CreateRenderTexture, 167
 - DisconnectNewImageFrame, 167
 - EnableSaveFrame, 168
 - Fini, 168
 - GetAspectRatio, 168
 - GetAvgFPS, 168
 - GetCameraToViewportRay, 168
 - GetDirection, 168
 - GetFarClip, 169
 - GetFrameFilename, 169
 - GetHFOV, 169
 - GetImageByteSize, 169
 - GetImageData, 170
 - GetImageDepth, 170
 - GetImageFormat, 170
 - GetImageHeight, 170
 - GetImageWidth, 170
 - GetInitialized, 170
 - GetLastRenderWallTime, 171
 - GetName, 171
 - GetNearClip, 171
 - GetOgreCamera, 171
 - GetPitchNode, 171
 - GetRenderRate, 171
 - GetRenderTexture, 172
 - GetRight, 172
 - GetScene, 172
 - GetSceneNode, 172
 - GetTextureHeight, 172
 - GetTextureWidth, 172
 - GetTriangleCount, 173
 - GetUp, 173
 - GetVFOV, 173
 - GetViewport, 173
 - GetViewportHeight, 173
 - GetViewportWidth, 173
 - GetWindowId, 174
 - GetWorldPointOnPlane, 174
 - GetWorldPose, 174
 - GetWorldPosition, 174
 - GetWorldRotation, 174
 - GetZValue, 175
 - imageFormat, 183
 - imageHeight, 183
 - imageWidth, 183
 - Init, 175
 - initialized, 183
 - IsAnimating, 175
 - IsInitialized, 175
 - IsVisible, 175, 176
 - lastRenderWallTime, 183
 - Load, 176
 - MoveToPosition, 176
 - MoveToPositions, 176
 - name, 183
 - newData, 183
 - newImageFrame, 183
 - onAnimationComplete, 183
 - pitchNode, 183
 - PostRender, 177
 - prevAnimTime, 183
 - Render, 177
 - RenderImpl, 177
 - renderTarget, 184
 - renderTexture, 184
 - requests, 184
 - RotatePitch, 177
 - RotateYaw, 177
 - saveCount, 184
 - SaveFrame, 177, 178
 - saveFrameBuffer, 184
 - scene, 184
 - sceneNode, 184
 - sdf, 184
 - SetAspectRatio, 178
 - SetCaptureData, 178
 - SetClipDist, 178
 - SetHFOV, 179
 - SetImageHeight, 179
 - SetImageSize, 179
 - SetImageWidth, 179
 - SetName, 179
 - SetRenderRate, 179
 - SetRenderTarget, 180
 - SetSaveFramePathname, 180
 - SetScene, 180
 - SetSceneNode, 180
 - SetWindowId, 180
 - SetWorldPose, 180
 - SetWorldPosition, 181

- SetWorldRotation, 181
- ShowWireframe, 181
- textureHeight, 184
- textureWidth, 184
- ToggleShowWireframe, 181
- TrackVisual, 181
- TrackVisualImpl, 181, 182
- Translate, 182
- Update, 182
- viewport, 184
- windowId, 184
- gazebo::rendering::CameraVisual, 189
 - ~CameraVisual, 190
 - CameraVisual, 190
 - Load, 190
- gazebo::rendering::ContactVisual, 242
 - ~ContactVisual, 243
 - ContactVisual, 243
 - SetEnabled, 243
- gazebo::rendering::Conversions, 243
 - Convert, 244, 245
- gazebo::rendering::DepthCamera, 249
 - ~DepthCamera, 251
 - ConnectNewDepthFrame, 251
 - ConnectNewRGBPointCloud, 252
 - CreateDepthTexture, 252
 - DepthCamera, 251
 - depthTarget, 254
 - depthTexture, 254
 - depthViewport, 254
 - DisconnectNewDepthFrame, 252
 - DisconnectNewRGBPointCloud, 252
 - Fini, 253
 - GetDepthData, 253
 - Init, 253
 - Load, 253
 - PostRender, 253
 - SetDepthTarget, 253
- gazebo::rendering::DynamicLines, 262
 - ~DynamicLines, 263
 - AddPoint, 264
 - Clear, 264
 - CreateVertexDeclaration, 264
 - DynamicLines, 263
 - FillHardwareBuffers, 264
 - GetMovableType, 264
 - getMovableType, 264
 - GetPoint, 265
 - GetPointCount, 265
 - SetPoint, 265
 - Update, 265
- gazebo::rendering::DynamicRenderable, 265
 - ~DynamicRenderable, 267
 - CreateVertexDeclaration, 267
 - DynamicRenderable, 267
 - FillHardwareBuffers, 267
 - getBoundingRadius, 267
 - GetOperationType, 268
 - getSquaredViewDepth, 268
 - indexBufferCapacity, 269
 - Init, 268
 - PrepareHardwareBuffers, 268
 - SetOperationType, 269
 - vertexBufferCapacity, 269
- gazebo::rendering::Events, 291
 - ConnectCreateScene, 291
 - ConnectRemoveScene, 292
 - createScene, 292
 - DisconnectCreateScene, 292
 - DisconnectRemoveScene, 292
 - removeScene, 292
- gazebo::rendering::FPSViewController, 316
 - ~FPSViewController, 318
 - FPSViewController, 318
 - GetTypeString, 318
 - HandleKeyPressEvent, 318
 - HandleKeyReleaseEvent, 318
 - HandleMouseEvent, 318
 - Init, 319
 - Update, 319
- gazebo::rendering::GUIOverlay, 341
 - ~GUIOverlay, 342
 - AttachCameraToImage, 342
 - ButtonCallback, 343
 - CreateWindow, 343
 - GUIOverlay, 342
 - HandleKeyPressEvent, 343
 - HandleKeyReleaseEvent, 344
 - HandleMouseEvent, 344
 - Hide, 344
 - Init, 344
 - IsInitialized, 344
 - LoadLayout, 344
 - Resize, 345
 - Show, 345
 - Update, 345
- gazebo::rendering::GpuLaser, 320
 - ~GpuLaser, 322
 - ConnectNewLaserFrame, 322
 - CreateLaserTexture, 322
 - DisconnectNewLaserFrame, 323
 - Fini, 323
 - GetLaserData, 323
 - GpuLaser, 322
 - Init, 323
 - Load, 323
 - notifyRenderSingleObject, 323
 - PostRender, 323

- SetParentSensor, 323
- SetRangeCount, 324
- gazebo::rendering::Grid, 336
 - ~Grid, 337
 - Enable, 338
 - GetCellCount, 338
 - GetCellLength, 338
 - GetColor, 338
 - GetHeight, 338
 - GetLineWidth, 338
 - GetSceneNode, 338
 - Grid, 337
 - Init, 339
 - SetCellCount, 339
 - SetCellLength, 339
 - SetColor, 339
 - SetHeight, 339
 - SetLineWidth, 339
 - SetUserData, 340
- gazebo::rendering::GzTerrainMatGen, 345
 - ~GzTerrainMatGen, 346
 - GzTerrainMatGen, 346
- gazebo::rendering::GzTerrainMatGen::SM2Profile, 734
 - ~SM2Profile, 736
 - addTechnique, 736
 - generate, 736
 - generateForCompositeMap, 736
 - SM2Profile, 736
 - UpdateParams, 736
 - UpdateParamsForCompositeMap, 736
- gazebo::rendering::GzTerrainMatGen::SM2Profile::-
 - ShaderHelperCg, 701
 - defaultVpParams, 702
 - generateFragmentProgram, 702
 - generateVertexProgram, 702
 - generateVertexProgramSource, 702
 - generateVpDynamicShadows, 702
 - generateVpDynamicShadowsParams, 702
 - generateVpFooter, 703
 - generateVpHeader, 703
- gazebo::rendering::GzTerrainMatGen::SM2Profile::-
 - ShaderHelperGLSL, 703
 - defaultVpParams, 704
 - generateFpDynamicShadows, 704
 - generateFpDynamicShadowsHelpers, 704
 - generateFpDynamicShadowsParams, 705
 - generateFpFooter, 705
 - generateFpHeader, 705
 - generateFpLayer, 705
 - generateFragmentProgram, 705
 - generateFragmentProgramSource, 705
 - generateVertexProgram, 705
 - generateVertexProgramSource, 705
 - generateVpDynamicShadows, 705
 - generateVpDynamicShadowsParams, 705
 - generateVpFooter, 705
 - generateVpHeader, 705
 - updateParams, 705
 - updateVpParams, 706
- gazebo::rendering::Heightmap, 346
 - ~Heightmap, 347
 - GetHeight, 347
 - GetOgreTerrain, 347
 - Heightmap, 347
 - Load, 347
 - LoadFromMsg, 347
- gazebo::rendering::JointVisual, 399
 - ~JointVisual, 400
 - JointVisual, 400
 - Load, 400
- gazebo::rendering::LaserVisual, 403
 - ~LaserVisual, 405
 - LaserVisual, 404
 - SetEmissive, 405
- gazebo::rendering::Light, 405
 - ~Light, 407
 - FillMsg, 407
 - GetDiffuseColor, 407
 - GetDirection, 407
 - GetName, 407
 - GetPosition, 407
 - GetSpecularColor, 408
 - GetType, 408
 - Light, 407
 - Load, 408
 - LoadFromMsg, 408
 - OnPoseChange, 408
 - SetAttenuation, 408
 - SetCastShadows, 409
 - SetDiffuseColor, 409
 - SetDirection, 409
 - SetLightType, 409
 - SetName, 409
 - SetPosition, 409
 - SetRange, 410
 - SetSelected, 410
 - SetSpecularColor, 410
 - SetSpotFalloff, 410
 - SetSpotInnerAngle, 410
 - SetSpotOuterAngle, 410
 - ShowVisual, 411
 - ToggleShowVisual, 411
 - UpdateFromMsg, 411
- gazebo::rendering::MovableText, 506
 - ~MovableText, 508
 - _setupGeometry, 509
 - _updateColors, 509
 - GetAABB, 509

- GetBaseline, 509
- getBoundingRadius, 509
- GetCharHeight, 509
- GetColor, 509
- GetFont, 509
- getLights, 509
- getMaterial, 509
- getRenderOperation, 509
- GetShowOnTop, 510
- GetSpaceWidth, 510
- getSquaredViewDepth, 510
- GetText, 510
- getWorldTransforms, 510
- H_CENTER, 508
- H_LEFT, 508
- HorizAlign, 508
- Load, 510
- MovableText, 508
- SetBaseline, 510
- SetCharHeight, 511
- SetColor, 511
- SetFontName, 511
- SetShowOnTop, 511
- SetSpaceWidth, 511
- SetText, 511
- SetTextAlignment, 512
- Update, 512
- V_ABOVE, 508
- V_BELOW, 508
- VertAlign, 508
- visitRenderables, 512
- gazebo::rendering::OrbitViewController, 545
 - ~OrbitViewController, 547
 - GetFocalPoint, 547
 - GetTypeString, 547
 - HandleKeyPressEvent, 547
 - HandleKeyReleaseEvent, 547
 - HandleMouseEvent, 548
 - Init, 548
 - OrbitViewController, 547
 - SetDistance, 548
 - SetFocalPoint, 548
 - Update, 549
- gazebo::rendering::Projector, 597
 - ~Projector, 598
 - GetParent, 598
 - Load, 598, 599
 - Projector, 598
 - SetEnabled, 599
 - SetTexture, 599
 - Toggle, 599
- gazebo::rendering::RFIDTagVisual, 648
 - ~RFIDTagVisual, 650
 - RFIDTagVisual, 649
- gazebo::rendering::RFIDVisual, 650
 - ~RFIDVisual, 651
 - RFIDVisual, 651
- gazebo::rendering::RTShaderSystem, 658
 - AddScene, 659
 - ApplyShadows, 660
 - AttachEntity, 660
 - AttachViewport, 660
 - Clear, 660
 - DetachEntity, 660
 - DetachViewport, 660
 - Fini, 661
 - GenerateShaders, 661
 - GetPSSMShadowCameraSetup, 661
 - Init, 661
 - LightingModel, 659
 - RemoveScene, 661
 - RemoveShadows, 661
 - SSLM_NormalMapLightingObjectSpace, 659
 - SSLM_NormalMapLightingTangentSpace, 659
 - SSLM_PerPixelLighting, 659
 - SSLM_PerVertexLighting, 659
 - SetPerPixelLighting, 661
 - UpdateShaders, 662
- gazebo::rendering::RenderEngine, 639
 - AddResourcePath, 641
 - CreateScene, 641
 - DEFERRED, 641
 - dummyContext, 643
 - dummyDisplay, 643
 - dummyWindowId, 643
 - FORWARD, 641
 - Fini, 642
 - GetRenderPathType, 642
 - GetScene, 642
 - GetSceneCount, 642
 - Init, 643
 - Load, 643
 - NONE, 641
 - RENDER_PATH_COUNT, 641
 - RemoveScene, 643
 - RenderPathType, 641
 - root, 643
 - VERTEX, 641
- gazebo::rendering::Road2d, 653
 - ~Road2d, 653
 - Load, 654
 - Road2d, 653
- gazebo::rendering::Scene, 662
 - ~Scene, 665
 - AddVisual, 665
 - Clear, 666
 - CloneVisual, 666
 - CreateCamera, 666

- CreateDepthCamera, 666
- CreateGrid, 667
- CreateUserCamera, 667
- DrawLine, 667
- GetAmbientColor, 667
- GetBackgroundColor, 667
- GetCamera, 668
- GetCameraCount, 668
- GetFirstContact, 668
- GetGrid, 669
- GetGridCount, 669
- GetHeightBelowPoint, 669
- GetHeightmap, 669
- GetId, 670
- GetIdString, 670
- GetInitialized, 670
- GetLight, 670
- GetLightCount, 670
- GetManager, 671
- GetModelVisualAt, 671
- GetName, 671
- GetSelectedVisual, 671
- GetShadowsEnabled, 671
- GetUserCamera, 672
- GetUserCameraCount, 672
- GetVisual, 672
- GetVisualAt, 672
- GetVisualBelow, 673
- GetVisualsBelowPoint, 673
- GetWorldVisual, 673
- Init, 673
- Load, 673, 674
- PreRender, 674
- PrintSceneGraph, 674
- RemoveVisual, 674
- Scene, 665
- SelectVisual, 674
- SetAmbientColor, 674
- SetBackgroundColor, 674
- SetFog, 675
- SetGrid, 675
- SetShadowsEnabled, 675
- SetTransparent, 675
- SetVisible, 675
- ShowCOMs, 676
- ShowCollisions, 676
- ShowContacts, 676
- ShowJoints, 676
- skyx, 677
- SnapVisualToNearestBelow, 676
- StripSceneName, 676
- gazebo::rendering::SelectionObj, 681
 - ~SelectionObj, 682
 - Attach, 682
 - Clear, 682
 - GetVisualName, 682
 - Init, 683
 - IsActive, 683
 - SelectionObj, 682
 - SetActive, 683
 - SetHighlight, 683
- gazebo::rendering::UserCamera, 813
 - ~UserCamera, 815
 - AnimationComplete, 815
 - AttachToVisualImpl, 815
 - EnableViewController, 816
 - Finis, 816
 - GetAvgFPS, 816
 - GetGUIOverlay, 816
 - GetImageHeight, 816
 - GetImageWidth, 817
 - GetTriangleCount, 817
 - GetViewControllerTypeString, 817
 - GetVisual, 817
 - HandleKeyPressEvent, 818
 - HandleKeyReleaseEvent, 818
 - HandleMouseEvent, 818
 - Init, 818
 - Load, 818
 - MoveToPosition, 818
 - MoveToVisual, 819
 - PostRender, 819
 - Resize, 819
 - SetFocalPoint, 819
 - SetRenderTarget, 820
 - SetViewController, 820
 - SetViewportDimensions, 820
 - SetWorldPose, 820
 - TrackVisualImpl, 821
 - Update, 821
 - UserCamera, 815
- gazebo::rendering::VideoVisual, 863
 - ~VideoVisual, 864
 - VideoVisual, 864
- gazebo::rendering::ViewController, 864
 - ~ViewController, 865
 - camera, 867
 - enabled, 867
 - GetTypeString, 865
 - HandleKeyPressEvent, 866
 - HandleKeyReleaseEvent, 866
 - HandleMouseEvent, 866
 - Init, 866
 - SetEnabled, 867
 - typeString, 867
 - Update, 867
 - ViewController, 865
- gazebo::rendering::Visual, 867

- ~Visual, 873
- AttachAxes, 873
- AttachLineVertex, 873
- AttachMesh, 873
- AttachObject, 873
- AttachVisual, 873
- ClearParent, 874
- Clone, 874
- CreateDynamicLine, 874
- DeleteDynamicLine, 874
- DetachObjects, 874
- DetachVisual, 874, 875
- DisableTrackVisual, 875
- EnableTrackVisual, 875
- Fini, 875
- GetAttachedObjectCount, 875
- GetBoundingBox, 875
- GetChild, 875
- GetChildCount, 876
- GetMaterialName, 876
- GetMeshName, 876
- GetName, 876
- GetNormalMap, 876
- GetParent, 876
- GetPose, 877
- GetPosition, 877
- GetRootVisual, 877
- GetRotation, 877
- GetScale, 877
- GetScene, 877
- GetSceneNode, 878
- GetShaderType, 878
- GetTransparency, 878
- GetVisibilityFlags, 878
- GetVisible, 878
- GetWorldPose, 879
- HasAttachedObject, 879
- Init, 879
- InsertMesh, 879
- IsPlane, 879
- IsStatic, 880
- Load, 880
- LoadFromMsg, 880
- LoadPlugin, 880
- MakeStatic, 880
- MoveToPosition, 880
- MoveToPositions, 881
- parent, 886
- RemovePlugin, 881
- scene, 886
- sceneNode, 887
- SetAmbient, 881
- SetCastShadows, 881
- SetDiffuse, 881
- SetEmissive, 882
- SetHighlighted, 882
- SetMaterial, 882
- SetName, 882
- SetNormalMap, 882
- SetPose, 882
- SetPosition, 883
- SetRibbonTrail, 883
- SetRotation, 883
- SetScale, 883
- SetScene, 883
- SetShaderType, 884
- SetSkeletonPose, 884
- SetSpecular, 884
- SetTransparency, 884
- SetVisibilityFlags, 884
- SetVisible, 885
- SetWorldPose, 885
- SetWorldPosition, 885
- SetWorldRotation, 885
- ShowBoundingBox, 885
- ShowCOM, 885
- ShowCollision, 885
- ShowJoints, 886
- ShowSkeleton, 886
- ToggleVisible, 886
- Update, 886
- UpdateFromMsg, 886
- Visual, 872
- gazebo::rendering::WindowManager, 888
 - CreateWindow, 889
 - Fini, 890
 - GetAvgFPS, 890
 - GetTriangleCount, 890
 - GetWindow, 890
 - Moved, 890
 - Resize, 891
 - SetCamera, 891
- gazebo::rendering::WireBox, 891
 - ~WireBox, 892
 - Init, 892
 - SetVisible, 892
 - WireBox, 892
- gazebo::sensors, 100
 - CATEGORY_COUNT, 102
 - CameraSensor_V, 102
 - CameraSensorPtr, 102
 - ContactSensor_V, 102
 - ContactSensorPtr, 102
 - DepthCameraSensor_V, 102
 - DepthCameraSensorPtr, 102
 - GpuRaySensor_V, 102
 - GpuRaySensorPtr, 102
 - IMAGE, 102

- ImuSensor_V, 102
- ImuSensorPtr, 102
- OTHER, 102
- RAY, 102
- RFIDSensor_V, 102
- RFIDSensorPtr, 102
- RFIDTag_V, 102
- RFIDTagPtr, 102
- RaySensor_V, 102
- RaySensorPtr, 102
- Sensor_V, 102
- SensorCategory, 102
- SensorFactoryFn, 102
- SensorPtr, 102
- gazebo::sensors::CameraSensor, 185
 - ~CameraSensor, 186
 - CameraSensor, 186
 - Fini, 186
 - GetCamera, 186
 - GetImageData, 187
 - GetImageHeight, 187
 - GetImageWidth, 187
 - GetTopic, 187
 - Init, 187
 - IsActive, 187
 - Load, 188
 - SaveFrame, 188
 - SetParent, 188
 - UpdateImpl, 189
- gazebo::sensors::ContactSensor, 237
 - ~ContactSensor, 239
 - ContactSensor, 239
 - Fini, 239
 - GetCollisionContactCount, 239
 - GetCollisionCount, 239
 - GetCollisionName, 240
 - GetContacts, 240
 - Init, 241
 - IsActive, 241
 - Load, 241
 - UpdateImpl, 241
- gazebo::sensors::DepthCameraSensor, 254
 - ~DepthCameraSensor, 255
 - DepthCameraSensor, 255
 - Fini, 255
 - GetDepthCamera, 255
 - Init, 256
 - Load, 256
 - SaveFrame, 256
 - SetActive, 256
 - SetParent, 257
 - UpdateImpl, 257
- gazebo::sensors::GpuRaySensor, 324
 - ~GpuRaySensor, 327
 - cameraCount, 334
 - cameraElem, 334
 - chfov, 334
 - ConnectNewLaserFrame, 328
 - cvfov, 334
 - DisconnectNewLaserFrame, 328
 - far, 335
 - Fini, 328
 - GetAngleMax, 328
 - GetAngleMin, 328
 - GetAngleResolution, 328
 - GetCameraCount, 328
 - GetCosHorzFOV, 329
 - GetCosVertFOV, 329
 - GetFiducial, 329
 - GetHorzFOV, 329
 - GetHorzHalfAngle, 329
 - GetLaserCamera, 330
 - GetRange, 330
 - GetRangeCount, 330
 - GetRangeCountRatio, 330
 - GetRangeMax, 330
 - GetRangeMin, 331
 - GetRangeResolution, 331
 - GetRanges, 331
 - GetRayCount, 331
 - GetRayCountRatio, 331
 - GetRetro, 331
 - GetVertFOV, 332
 - GetVertHalfAngle, 332
 - GetVerticalAngleMax, 332
 - GetVerticalAngleMin, 332
 - GetVerticalRangeCount, 332
 - GetVerticalRayCount, 332
 - GpuRaySensor, 327
 - hfov, 335
 - horzElem, 335
 - horzHalfAngle, 335
 - horzRangeCount, 335
 - horzRayCount, 335
 - Init, 333
 - IsHorizontal, 333
 - isHorizontal, 335
 - Load, 333
 - near, 335
 - rangeCountRatio, 335
 - rangeElem, 335
 - rayCountRatio, 335
 - scanElem, 336
 - SetAngleMax, 333
 - SetAngleMin, 333
 - SetVerticalAngleMax, 334
 - SetVerticalAngleMin, 334
 - UpdateImpl, 334

- vertElem, 336
- vertHalfAngle, 336
- vertRangeCount, 336
- vertRayCount, 336
- vfov, 336
- gazebo::sensors::ImuSensor, 361
 - ~ImuSensor, 363
 - Fini, 363
 - GetAngularVelocity, 363
 - GetLinearAcceleration, 363
 - GetOrientation, 363
 - ImuSensor, 363
 - Init, 363
 - Load, 364
 - SetReferencePose, 364
 - UpdateImpl, 364
- gazebo::sensors::MultiCameraSensor, 513
 - ~MultiCameraSensor, 515
 - Fini, 515
 - GetCamera, 515
 - GetCameraCount, 515
 - GetImageData, 516
 - GetImageHeight, 516
 - GetImageWidth, 516
 - GetTopic, 517
 - Init, 517
 - IsActive, 517
 - Load, 517
 - MultiCameraSensor, 515
 - SaveFrame, 517
 - UpdateImpl, 518
- gazebo::sensors::RFIDSensor, 644
 - ~RFIDSensor, 645
 - AddTag, 645
 - Fini, 645
 - Init, 645
 - Load, 645
 - RFIDSensor, 645
 - UpdateImpl, 646
- gazebo::sensors::RFIDTag, 646
 - ~RFIDTag, 647
 - Fini, 647
 - GetTagPose, 647
 - Init, 648
 - Load, 648
 - RFIDTag, 647
 - UpdateImpl, 648
- gazebo::sensors::RaySensor, 627
 - ~RaySensor, 629
 - Fini, 629
 - GetAngleMax, 629
 - GetAngleMin, 629
 - GetAngleResolution, 629
 - GetFiducial, 630
 - GetLaserShape, 630
 - GetRange, 630
 - GetRangeCount, 631
 - GetRangeMax, 631
 - GetRangeMin, 631
 - GetRangeResolution, 631
 - GetRanges, 631
 - GetRayCount, 631
 - GetRetro, 632
 - GetTopic, 632
 - GetVerticalAngleMax, 632
 - GetVerticalAngleMin, 632
 - GetVerticalRangeCount, 632
 - GetVerticalRayCount, 633
 - Init, 633
 - IsActive, 633
 - Load, 633
 - RaySensor, 629
 - UpdateImpl, 633
- gazebo::sensors::Sensor, 683
 - ~Sensor, 686
 - active, 692
 - ConnectUpdated, 686
 - connections, 692
 - DisconnectUpdated, 687
 - FillMsg, 687
 - Fini, 687
 - GetCategory, 687
 - GetLastMeasurementTime, 688
 - GetLastUpdateTime, 688
 - GetName, 688
 - GetParentName, 688
 - GetPose, 688
 - GetScopedName, 688
 - GetTopic, 689
 - GetType, 689
 - GetUpdateRate, 689
 - GetVisualize, 689
 - GetWorldName, 689
 - Init, 689
 - IsActive, 690
 - lastMeasurementTime, 692
 - lastUpdateTime, 692
 - Load, 690
 - node, 692
 - parentName, 692
 - plugins, 692
 - pose, 692
 - poseSub, 692
 - ResetLastUpdateTime, 690
 - sdf, 692
 - Sensor, 686
 - SetActive, 690
 - SetParent, 691

- SetUpdateRate, 691
- Update, 691
- UpdateImpl, 691
- updatePeriod, 692
- world, 693
- gazebo::sensors::SensorFactory, 693
 - GetSensorTypes, 693
 - NewSensor, 694
 - RegisterAll, 694
 - RegisterSensor, 694
- gazebo::sensors::SensorManager, 695
 - CreateSensor, 696
 - Fini, 696
 - GetSensor, 696
 - GetSensorTypes, 697
 - GetSensors, 696
 - Init, 697
 - RemoveSensor, 697
 - RemoveSensors, 697
 - ResetLastUpdateTimes, 697
 - Run, 697
 - RunThreads, 697
 - SensorsInitialized, 697
 - Stop, 698
 - Update, 698
- gazebo::sensors::SimTimeEvent, 708
 - condition, 708
 - time, 708
- gazebo::sensors::SimTimeEventHandler, 709
 - ~SimTimeEventHandler, 709
 - AddRelativeEvent, 710
 - SimTimeEventHandler, 709
- gazebo::transport, 103
 - ConnectionPtr, 104
 - NodePtr, 104
 - PublicationPtr, 104
 - PublicationTransportPtr, 104
 - PublisherPtr, 104
 - SubscriberPtr, 104
 - SubscriptionTransportPtr, 104
- gazebo::transport::CallbackHelper, 155
 - ~CallbackHelper, 156
 - CallbackHelper, 156
 - GetId, 156
 - GetLatching, 156
 - GetMsgType, 156
 - HandleData, 157
 - IsLocal, 157
 - latching, 157
- gazebo::transport::CallbackHelperT
 - CallbackHelperT, 158
 - GetMsgType, 159
 - HandleData, 159
 - IsLocal, 159
- gazebo::transport::CallbackHelperT< M >, 157
- gazebo::transport::Connection, 219
 - ~Connection, 221
 - AcceptCallback, 221
 - AsyncRead, 221
 - Cancel, 221
 - Connect, 221
 - ConnectToShutdown, 222
 - Connection, 221
 - DisconnectShutdown, 222
 - EnqueueMsg, 222
 - GetId, 222
 - GetLocalAddress, 222
 - GetLocalHostname, 223
 - GetLocalPort, 223
 - GetLocalURI, 223
 - GetRemoteAddress, 223
 - GetRemoteHostname, 223
 - GetRemotePort, 223
 - GetRemoteURI, 224
 - IsOpen, 224
 - Listen, 224
 - ProcessWriteQueue, 224
 - Read, 224
 - ReadCallback, 221
 - Shutdown, 225
 - StartRead, 225
 - StopRead, 225
 - ValidateIP, 225
- gazebo::transport::ConnectionManager, 225
 - Advertise, 227
 - ConnectToRemoteHost, 227
 - eventConnections, 230
 - Fini, 227
 - GetAllPublishers, 227
 - GetTopicNamespaces, 228
 - Init, 228
 - IsRunning, 228
 - RegisterTopicNamespace, 228
 - RemoveConnection, 228
 - Run, 228
 - RunUpdate, 229
 - Stop, 229
 - Subscribe, 229
 - Unadvertise, 229
 - Unsubscribe, 229
- gazebo::transport::ConnectionReadTask, 230
 - ConnectionReadTask, 230
 - execute, 231
- gazebo::transport::IOManager, 374
 - ~IOManager, 374
 - DecCount, 375
 - GetCount, 375
 - GetIO, 375

- IOManager, 374
- IncCount, 375
- Stop, 375
- gazebo::transport::Node, 526
 - ~Node, 528
 - Advertise, 528
 - DecodeTopicName, 528
 - EncodeTopicName, 528
 - Fini, 529
 - GetId, 529
 - GetMsgType, 529
 - GetTopicNamespace, 529
 - HandleData, 529
 - HasLatchedSubscriber, 530
 - Init, 530
 - InsertLatchedMsg, 530
 - Node, 528
 - ProcessIncoming, 530
 - ProcessPublishers, 530
 - RemoveCallback, 530
 - Subscribe, 530–532
- gazebo::transport::Publication, 600
 - ~Publication, 601
 - AddPublisher, 601
 - AddSubscription, 601
 - AddTransport, 601
 - GetCallbackCount, 602
 - GetLocallyAdvertised, 602
 - GetMsgType, 602
 - GetNodeCount, 602
 - GetRemoteSubscriptionCount, 602
 - GetTransportCount, 602
 - HasTransport, 603
 - LocalPublish, 603
 - Publication, 601
 - Publish, 603
 - RemoveSubscription, 603
 - RemoveTransport, 604
 - SetLocallyAdvertised, 604
- gazebo::transport::PublicationTransport, 604
 - ~PublicationTransport, 605
 - AddCallback, 605
 - Fini, 605
 - GetConnection, 605
 - GetMsgType, 606
 - GetTopic, 606
 - Init, 606
 - PublicationTransport, 605
- gazebo::transport::Publisher, 606
 - ~Publisher, 607
 - GetLatching, 608
 - GetMsgType, 608
 - GetOutgoingCount, 608
 - GetPrevMsg, 608
 - GetTopic, 608
 - HasConnections, 608
 - Publish, 608, 609
 - Publisher, 607
 - SendMessage, 609
 - SetPublication, 609
 - WaitForConnection, 609
- gazebo::transport::RawCallbackHelper, 625
 - GetMsgType, 626
 - HandleData, 626
 - IsLocal, 627
 - RawCallbackHelper, 626
- gazebo::transport::SubscribeOptions, 758
 - GetLatching, 759
 - GetMsgType, 759
 - GetNode, 759
 - GetTopic, 759
 - Init, 760
 - SubscribeOptions, 759
- gazebo::transport::Subscriber, 760
 - ~Subscriber, 761
 - GetCallbackId, 761
 - GetTopic, 761
 - SetCallbackId, 761
 - Subscriber, 761
 - Unsubscribe, 761
- gazebo::transport::SubscriptionTransport, 762
 - ~SubscriptionTransport, 763
 - GetConnection, 763
 - HandleData, 763
 - Init, 763
 - IsLocal, 763
 - SubscriptionTransport, 763
- gazebo::transport::TopicManager, 798
 - AddNode, 800
 - Advertise, 801
 - ClearBuffers, 801
 - ConnectPubToSub, 801
 - ConnectSubToPub, 801
 - ConnectSubscribers, 801
 - DisconnectPubFromSub, 801
 - DisconnectSubFromPub, 802
 - FindPublication, 802
 - Fini, 802
 - GetAdvertisedTopics, 802
 - GetTopicNamespaces, 802
 - Init, 803
 - IsAdvertised, 803
 - PauseIncoming, 803
 - ProcessNodes, 803
 - Publish, 803
 - RegisterTopicNamespace, 804
 - RemoveNode, 804
 - SubNodeMap, 800

- Subscribe, 804
- Unadvertise, 804
- Unsubscribe, 804
- UpdatePublications, 805
- gazebo::util, 105
 - DiagnosticTimerPtr, 105
- gazebo::util::DiagnosticManager, 257
 - GetLabel, 258
 - GetLogPath, 258
 - GetTime, 258, 259
 - GetTimerCount, 259
 - Init, 259
 - Lap, 259
 - StartTimer, 260
 - StopTimer, 260
- gazebo::util::DiagnosticTimer, 260
 - ~DiagnosticTimer, 261
 - DiagnosticTimer, 261
 - GetName, 261
 - Lap, 261
 - Start, 261
 - Stop, 262
- gazebo_core.hh, 959
- Gazebo_parser, 68
 - ConstUrdfLinkPtr, 68
 - UrdfCollisionPtr, 68
 - UrdfLinkPtr, 68
 - UrdfVisualPtr, 68
- GazeboGenerator
 - google::protobuf::compiler::cpp::GazeboGenerator, 320
- GazeboGenerator.hh, 959
- gazeboPathsFromEnv
 - gazebo::common::SystemPaths, 773
- GenSphericalTexCoord
 - gazebo::common::Mesh, 469
 - gazebo::common::MeshManager, 478
 - gazebo::common::SubMesh, 753
- Generate
 - google::protobuf::compiler::cpp::GazeboGenerator, 320
- generate
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 736
- generateForCompositeMap
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 736
- generateFpDynamicShadows
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 704
- generateFpDynamicShadowsHelpers
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 704
- generateFpDynamicShadowsParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 705
- generateFpFooter
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 705
- generateFpHeader
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 705
- generateFpLayer
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 705
- generateFragmentProgram
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 702
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 705
- generateFragmentProgramSource
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 705
- GenerateShaders
 - gazebo::rendering::RTShaderSystem, 661
- generateVertexProgram
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 702
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 705
- generateVertexProgramSource
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 702
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 705
- generateVpDynamicShadows
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 702
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 705
- generateVpDynamicShadowsParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 702
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 705
- generateVpFooter
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 703
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 705
- generateVpHeader
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 703
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 705
- GeneratorType
 - gazebo::math, 89
- Get

- gazebo::common::NodeTransform, 539
- sdf::Param, 551, 552
- get_master_uri
 - Transport, 76
- get_scene
 - Rendering, 66
- get_sensor
 - Sensors, 71
- get_topic_namespaces
 - Transport, 76
- get_world
 - Classes for physics and dynamics, 44
- GetAABB
 - gazebo::common::Mesh, 469
 - gazebo::rendering::MovableText, 509
- GetAbs
 - gazebo::math::Vector3, 843
- GetAcceleration
 - gazebo::physics::LinkState, 433
- GetAdvertisedTopics
 - gazebo::transport::TopicManager, 802
- getAdvertisedTopics
 - Transport, 76
- GetAllPublishers
 - gazebo::transport::ConnectionManager, 227
- GetAmbient
 - gazebo::common::Material, 450
- GetAmbientColor
 - gazebo::rendering::Scene, 667
- GetAnchor
 - gazebo::physics::Joint, 381
 - gazebo::physics::ScrewJoint, 678
 - gazebo::physics::SliderJoint, 733
- GetAngle
 - gazebo::physics::Joint, 382
 - gazebo::physics::JointState, 397
- GetAngleCount
 - gazebo::physics::BallJoint, 135
 - gazebo::physics::Hinge2Joint, 354
 - gazebo::physics::HingeJoint, 355
 - gazebo::physics::Joint, 382
 - gazebo::physics::JointState, 397
 - gazebo::physics::ScrewJoint, 679
 - gazebo::physics::SliderJoint, 734
 - gazebo::physics::UniversalJoint, 810
- GetAngleImpl
 - gazebo::physics::Joint, 382
- GetAngleMax
 - gazebo::sensors::GpuRaySensor, 328
 - gazebo::sensors::RaySensor, 629
- GetAngleMin
 - gazebo::sensors::GpuRaySensor, 328
 - gazebo::sensors::RaySensor, 629
- GetAngleResolution
 - gazebo::sensors::GpuRaySensor, 328
 - gazebo::sensors::RaySensor, 629
- GetAngles
 - gazebo::physics::JointState, 397
- GetAngularDamping
 - gazebo::physics::Link, 419
- GetAngularVelocity
 - gazebo::sensors::ImuSensor, 363
- GetAnimation
 - gazebo::common::Skeleton, 715
- GetAsABGR
 - gazebo::common::Color, 208
- GetAsARGB
 - gazebo::common::Color, 208
- GetAsAxis
 - gazebo::math::Quaternion, 614
- GetAsBGRA
 - gazebo::common::Color, 208
- GetAsEuler
 - gazebo::math::Quaternion, 615
- GetAsHSV
 - gazebo::common::Color, 209
- GetAsMatrix3
 - gazebo::math::Quaternion, 615
- GetAsMatrix4
 - gazebo::math::Quaternion, 615
- GetAsPose
 - gazebo::math::Matrix4, 462
- GetAsRGBA
 - gazebo::common::Color, 209
- GetAsString
 - sdf::Param, 552
 - sdf::ParamT, 557
- GetAsYUV
 - gazebo::common::Color, 209
- GetAspectRatio
 - gazebo::rendering::Camera, 168
- GetAttachedObjectCount
 - gazebo::rendering::Visual, 875
- GetAttribute
 - gazebo::physics::Joint, 382
 - sdf::Element, 273
- GetAttributeCount
 - sdf::Element, 273
- GetAttributeSet
 - sdf::Element, 273
- GetAutoDisable
 - gazebo::physics::Model, 484
- GetAutoDisableFlag
 - gazebo::physics::PhysicsEngine, 562
- GetAvgColor
 - gazebo::common::Image, 358
- GetAvgFPS
 - gazebo::rendering::Camera, 168

- gazebo::rendering::UserCamera, 816
- gazebo::rendering::WindowManager, 890
- GetBPP
 - gazebo::common::Image, 359
- GetBackgroundColor
 - gazebo::rendering::Scene, 667
- GetBasePath
 - gazebo::common::LogRecord, 442
- GetBaseline
 - gazebo::rendering::MovableText, 509
- GetBindShapeTransform
 - gazebo::common::Skeleton, 715
- GetBlendFactors
 - gazebo::common::Material, 450
- GetBlendMode
 - gazebo::common::Material, 450
- GetBoundingBox
 - gazebo::physics::Collision, 195
 - gazebo::physics::Entity, 280
 - gazebo::physics::Link, 419
 - gazebo::physics::Model, 484
 - gazebo::rendering::Visual, 875
- getBoundingRadius
 - gazebo::rendering::DynamicRenderable, 267
 - gazebo::rendering::MovableText, 509
- GetById
 - gazebo::physics::Base, 140
- GetByName
 - gazebo::physics::Base, 140
 - gazebo::physics::World, 896
- GetCallbackCount
 - gazebo::transport::Publication, 602
- GetCallbackId
 - gazebo::transport::Subscriber, 761
- GetCamera
 - gazebo::rendering::Scene, 668
 - gazebo::sensors::CameraSensor, 186
 - gazebo::sensors::MultiCameraSensor, 515
- GetCameraCount
 - gazebo::rendering::Scene, 668
 - gazebo::sensors::GpuRaySensor, 328
 - gazebo::sensors::MultiCameraSensor, 515
- GetCameraToViewportRay
 - gazebo::rendering::Camera, 168
- GetCategory
 - gazebo::sensors::Sensor, 687
- GetCellCount
 - gazebo::rendering::Grid, 338
- GetCellLength
 - gazebo::rendering::Grid, 338
- GetCenter
 - gazebo::math::Box, 148
- GetCharHeight
 - gazebo::rendering::MovableText, 509
- GetChild
 - gazebo::common::SkeletonNode, 726
 - gazebo::physics::Base, 140, 141
 - gazebo::physics::Joint, 382
 - gazebo::rendering::Visual, 875
- GetChildById
 - gazebo::common::SkeletonNode, 726
- GetChildByName
 - gazebo::common::SkeletonNode, 727
- GetChildCollision
 - gazebo::physics::Entity, 280
- GetChildCount
 - gazebo::common::SkeletonNode, 727
 - gazebo::physics::Base, 141
 - gazebo::rendering::Visual, 876
- GetChildJointsLinks
 - gazebo::physics::Link, 419
- GetChildLink
 - gazebo::physics::Entity, 281
- GetChunk
 - gazebo::common::LogPlay, 437
- GetChunkCount
 - gazebo::common::LogPlay, 438
- GetCmd
 - gazebo::common::PID, 572
- GetCoG
 - gazebo::physics::Inertial, 367
- GetCollision
 - gazebo::physics::Link, 419, 420
- GetCollisionBoundingBox
 - gazebo::physics::Entity, 281
- GetCollisionById
 - gazebo::physics::Link, 420
- GetCollisionContactCount
 - gazebo::sensors::ContactSensor, 239
- GetCollisionCount
 - gazebo::sensors::ContactSensor, 239
- GetCollisionName
 - gazebo::sensors::ContactSensor, 240
- GetCollisionState
 - gazebo::physics::LinkState, 433
- GetCollisionStateCount
 - gazebo::physics::LinkState, 434
- GetCollisionStates
 - gazebo::physics::LinkState, 434
- GetCollisions
 - gazebo::physics::Link, 420
- GetColor
 - gazebo::rendering::Grid, 338
 - gazebo::rendering::MovableText, 509
- GetConnection
 - gazebo::transport::PublicationTransport, 605
 - gazebo::transport::SubscriptionTransport, 763
- GetContact

- gazebo::physics::ContactManager, 236
- GetContactCount
 - gazebo::physics::ContactManager, 236
- GetContactManager
 - gazebo::physics::PhysicsEngine, 563
- GetContactMaxCorrectingVel
 - gazebo::physics::PhysicsEngine, 563
- GetContactSurfaceLayer
 - gazebo::physics::PhysicsEngine, 563
- GetContacts
 - gazebo::physics::ContactManager, 236
 - gazebo::sensors::ContactSensor, 240
- GetContactsEnabled
 - gazebo::physics::Collision, 195
- GetCopyChildren
 - sdf::Element, 273
- GetCosHorzFOV
 - gazebo::sensors::GpuRaySensor, 329
- GetCosVertFOV
 - gazebo::sensors::GpuRaySensor, 329
- GetCount
 - gazebo::transport::IOManager, 375
- GetCurrentDir
 - SystemPaths.hh, 1085
- GetDBConfig
 - Common, 33
- GetData
 - gazebo::common::Image, 359
- GetDbInNormal
 - gazebo::math::Rand, 624
- GetDbUniform
 - gazebo::math::Rand, 624
- GetDefaultAsString
 - sdf::Param, 552
 - sdf::ParamT, 557
- GetDefaultValue
 - sdf::ParamT, 557
- GetDepthCamera
 - gazebo::sensors::DepthCameraSensor, 255
- GetDepthData
 - gazebo::rendering::DepthCamera, 253
- GetDepthWrite
 - gazebo::common::Material, 450
- GetDescription
 - sdf::Element, 273
 - sdf::Param, 552
- GetDiffuse
 - gazebo::common::Material, 450
- GetDiffuseColor
 - gazebo::rendering::Light, 407
- GetDirection
 - gazebo::rendering::Camera, 168
 - gazebo::rendering::Light, 407
- GetDirtyPose
 - gazebo::physics::Entity, 281
- GetDistToLine
 - gazebo::math::Vector3, 843
- GetEffortLimit
 - gazebo::physics::Joint, 383
- GetElapsed
 - gazebo::common::Timer, 798
- GetElement
 - sdf::Element, 273
- GetElementDescription
 - sdf::Element, 273
- GetElementDescriptionCount
 - sdf::Element, 273
- GetElementImpl
 - sdf::Element, 274
- GetEmissive
 - gazebo::common::Material, 450
- GetEnablePhysicsEngine
 - gazebo::physics::World, 896
- GetEnabled
 - gazebo::physics::Link, 420
- GetEncoding
 - gazebo::common::LogPlay, 438
 - gazebo::common::LogRecord, 442
- GetEntity
 - gazebo::physics::World, 897
- GetEntityBelowPoint
 - gazebo::physics::World, 897
- GetErrorFile
 - gazebo::common::Exception, 315
- GetErrorStr
 - gazebo::common::Exception, 316
- GetErrors
 - gazebo::common::PID, 573
- GetEulerRotation
 - gazebo::math::Matrix4, 462
- GetExp
 - gazebo::math::Quaternion, 615
- GetFarClip
 - gazebo::rendering::Camera, 169
- GetFiducial
 - gazebo::physics::MultiRayShape, 522
 - gazebo::physics::RayShape, 636
 - gazebo::sensors::GpuRaySensor, 329
 - gazebo::sensors::RaySensor, 630
- GetFileSize
 - gazebo::common::LogRecord, 442
- GetFilename
 - gazebo::common::Image, 359
 - gazebo::common::LogRecord, 442
 - gazebo::physics::TrimeshShape, 808
 - gazebo::PluginT, 584
- GetFirstContact
 - gazebo::rendering::Scene, 668

- GetFirstElement
 - sdf::Element, 274
- GetFirstUpdate
 - gazebo::common::LogRecord, 442
- GetFocalPoint
 - gazebo::rendering::OrbitViewController, 547
- GetFont
 - gazebo::rendering::MovableText, 509
- GetForce
 - gazebo::physics::Joint, 383
- GetForceTorque
 - gazebo::physics::Joint, 383, 384
- GetFrameAt
 - gazebo::common::NodeAnimation, 534
- GetFrameCount
 - gazebo::common::NodeAnimation, 534
- GetFrameFilename
 - gazebo::rendering::Camera, 169
- GetGUIOverlay
 - gazebo::rendering::UserCamera, 816
- GetGazeboPaths
 - gazebo::common::SystemPaths, 771
- GetGazeboVersion
 - gazebo::common::LogPlay, 438
- GetGlobalAxis
 - gazebo::physics::Joint, 384
- GetGlobalPoints
 - gazebo::physics::RayShape, 636
- GetGravity
 - gazebo::physics::PhysicsEngine, 563
- GetGravityMode
 - gazebo::physics::Link, 421
- GetGrid
 - gazebo::rendering::Scene, 669
- GetGridCount
 - gazebo::rendering::Scene, 669
- GetHFOV
 - gazebo::rendering::Camera, 169
- GetHandle
 - gazebo::common::SkeletonNode, 727
 - gazebo::PluginT, 584
- GetHeader
 - Messages, 59
- GetHeight
 - gazebo::common::Image, 359
 - gazebo::common::Video, 862
 - gazebo::physics::HeightmapShape, 350
 - gazebo::rendering::Grid, 338
 - gazebo::rendering::Heightmap, 347
- GetHeightBelowPoint
 - gazebo::rendering::Scene, 669
- GetHeightmap
 - gazebo::rendering::Scene, 669
- GetHighStop
 - gazebo::physics::BallJoint, 135
 - gazebo::physics::Joint, 384
- GetHorzFOV
 - gazebo::sensors::GpuRaySensor, 329
- GetHorzHalfAngle
 - gazebo::sensors::GpuRaySensor, 329
- GetIO
 - gazebo::transport::IOManager, 375
- GetIXX
 - gazebo::physics::Inertial, 367
- GetIXY
 - gazebo::physics::Inertial, 367
- GetIXZ
 - gazebo::physics::Inertial, 367
- GetIYY
 - gazebo::physics::Inertial, 368
- GetIYZ
 - gazebo::physics::Inertial, 368
- GetIZZ
 - gazebo::physics::Inertial, 368
- GetId
 - gazebo::common::SkeletonNode, 727
 - gazebo::event::Connection, 219
 - gazebo::physics::Base, 141
 - gazebo::rendering::Scene, 670
 - gazebo::transport::CallbackHelper, 156
 - gazebo::transport::Connection, 222
 - gazebo::transport::Node, 529
- GetIdString
 - gazebo::rendering::Scene, 670
- GetImageByteSize
 - gazebo::rendering::Camera, 169
- GetImageData
 - gazebo::rendering::Camera, 170
 - gazebo::sensors::CameraSensor, 187
 - gazebo::sensors::MultiCameraSensor, 516
- GetImageDepth
 - gazebo::rendering::Camera, 170
- GetImageFormat
 - gazebo::rendering::Camera, 170
- GetImageHeight
 - gazebo::rendering::Camera, 170
 - gazebo::rendering::UserCamera, 816
 - gazebo::sensors::CameraSensor, 187
 - gazebo::sensors::MultiCameraSensor, 516
- GetImageWidth
 - gazebo::rendering::Camera, 170
 - gazebo::rendering::UserCamera, 817
 - gazebo::sensors::CameraSensor, 187
 - gazebo::sensors::MultiCameraSensor, 516
- GetInclude
 - sdf::Element, 274
- GetIndex
 - gazebo::common::SubMesh, 753

- GetIndexCount
 - gazebo::common::Mesh, 469
 - gazebo::common::SubMesh, 754
- GetInertiaRatio
 - gazebo::physics::Joint, 385
- GetInertial
 - gazebo::physics::Link, 421
- GetInitialRelativePose
 - gazebo::physics::Entity, 281
- GetInitialized
 - gazebo::rendering::Camera, 170
 - gazebo::rendering::Scene, 670
 - gazebo::Server, 700
- GetIntNormal
 - gazebo::math::Rand, 624
- GetIntUniform
 - gazebo::math::Rand, 624
- GetInterpolatedKeyFrame
 - gazebo::common::NumericAnimation, 543
 - gazebo::common::PoseAnimation, 594, 595
- GetIntersection
 - gazebo::physics::RayShape, 637
- GetInverse
 - gazebo::math::Pose, 589
 - gazebo::math::Quaternion, 615
- GetInverseBindTransform
 - gazebo::common::SkeletonNode, 727
- GetJoint
 - gazebo::physics::Model, 485
- GetJointController
 - gazebo::physics::Model, 485
- GetJointCount
 - gazebo::physics::Model, 485
- GetJointLink
 - gazebo::physics::Joint, 385
- GetJointState
 - gazebo::physics::ModelState, 499
- GetJointStateCount
 - gazebo::physics::ModelState, 500
- GetJointStates
 - gazebo::physics::ModelState, 500
- GetJoints
 - gazebo::physics::Model, 485
- GetKey
 - sdf::Param, 552
- GetKeyFrame
 - gazebo::common::Animation, 125
 - gazebo::common::NodeAnimation, 534
- GetKeyFrameCount
 - gazebo::common::Animation, 125
- GetKeyFramesATime
 - gazebo::common::Animation, 125
- GetKinematic
 - gazebo::physics::Link, 421
- GetLabel
 - gazebo::util::DiagnosticManager, 258
- GetLaserCamera
 - gazebo::sensors::GpuRaySensor, 330
- GetLaserData
 - gazebo::rendering::GpuLaser, 323
- GetLaserRetro
 - gazebo::physics::Collision, 196
- GetLaserShape
 - gazebo::sensors::RaySensor, 630
- GetLastMeasurementTime
 - gazebo::sensors::Sensor, 688
- GetLastRenderWallTime
 - gazebo::rendering::Camera, 171
- GetLastUpdateTime
 - gazebo::sensors::Sensor, 688
- GetLatching
 - gazebo::transport::CallbackHelper, 156
 - gazebo::transport::Publisher, 608
 - gazebo::transport::SubscribeOptions, 759
- GetLength
 - gazebo::common::Animation, 126
 - gazebo::common::NodeAnimation, 535
 - gazebo::common::SkeletonAnimation, 720
 - gazebo::math::Vector3, 843
 - gazebo::math::Vector4, 854
 - gazebo::physics::CylinderShape, 248
 - gazebo::physics::RayShape, 637
- GetLight
 - gazebo::rendering::Scene, 670
- GetLightCount
 - gazebo::rendering::Scene, 670
- GetLighting
 - gazebo::common::Material, 450
- getLights
 - gazebo::rendering::MovableText, 509
- GetLineWidth
 - gazebo::rendering::Grid, 338
- GetLinearAcceleration
 - gazebo::sensors::ImuSensor, 363
- GetLinearDamping
 - gazebo::physics::Link, 421
- GetLink
 - gazebo::physics::Collision, 196
 - gazebo::physics::Model, 485
- GetLinkByld
 - gazebo::physics::Model, 486
- GetLinkForce
 - gazebo::physics::Joint, 385
- GetLinkState
 - gazebo::physics::ModelState, 500
- GetLinkStateCount
 - gazebo::physics::ModelState, 501
- GetLinkStates

- gazebo::physics::ModelState, 501
- GetLinkTorque
 - gazebo::physics::Joint, 385
- GetLinks
 - gazebo::physics::Model, 486
- GetLocalAddress
 - gazebo::transport::Connection, 222
- GetLocalAxis
 - gazebo::physics::Joint, 386
- GetLocalHostname
 - gazebo::transport::Connection, 223
- GetLocalPort
 - gazebo::transport::Connection, 223
- GetLocalURI
 - gazebo::transport::Connection, 223
- GetLocallyAdvertised
 - gazebo::transport::Publication, 602
- GetLog
 - gazebo::math::Quaternion, 615
- GetLogPath
 - gazebo::common::SystemPaths, 772
 - gazebo::util::DiagnosticManager, 258
- GetLogVersion
 - gazebo::common::LogPlay, 438
- GetLowStop
 - gazebo::physics::BallJoint, 135
 - gazebo::physics::Joint, 386
- GetLowerLimit
 - gazebo::physics::Joint, 386
- GetManager
 - gazebo::rendering::Scene, 671
- GetManifest
 - Common, 34
- GetMass
 - gazebo::physics::Inertial, 368
- GetMaterial
 - gazebo::common::Mesh, 469
- getMaterial
 - gazebo::rendering::MovableText, 509
- GetMaterialCount
 - gazebo::common::Mesh, 469
- GetMaterialIndex
 - gazebo::common::SubMesh, 754
- GetMaterialName
 - gazebo::rendering::Visual, 876
- GetMax
 - gazebo::common::Mesh, 470
 - gazebo::common::SubMesh, 754
 - gazebo::math::Vector3, 844
- GetMaxAngle
 - gazebo::physics::MultiRayShape, 522
- GetMaxColor
 - gazebo::common::Image, 359
- GetMaxContacts
 - gazebo::physics::PhysicsEngine, 563
- GetMaxForce
 - gazebo::physics::Joint, 386
- GetMaxHeight
 - gazebo::physics::HeightmapShape, 350
- GetMaxIndex
 - gazebo::common::SubMesh, 754
- GetMaxRange
 - gazebo::physics::MultiRayShape, 522
- GetMesh
 - gazebo::common::MeshManager, 479
- GetMeshAABB
 - gazebo::common::MeshManager, 479
- GetMeshName
 - gazebo::rendering::Visual, 876
- GetMin
 - gazebo::common::Mesh, 470
 - gazebo::common::SubMesh, 754
 - gazebo::math::Vector3, 844
- GetMinAngle
 - gazebo::physics::MultiRayShape, 523
- GetMinHeight
 - gazebo::physics::HeightmapShape, 350
- GetMinRange
 - gazebo::physics::MultiRayShape, 523
- GetModel
 - gazebo::physics::Collision, 196
 - gazebo::physics::Link, 421
 - gazebo::physics::World, 897
- GetModelBelowPoint
 - gazebo::physics::World, 898
- GetModelConfig
 - Common, 34
- GetModelCount
 - gazebo::physics::World, 898
- GetModelFile
 - Common, 34
- GetModelName
 - Common, 34
- GetModelPath
 - Common, 35
- GetModelPaths
 - gazebo::common::SystemPaths, 772
- GetModelState
 - gazebo::physics::WorldState, 907, 908
- GetModelStateCount
 - gazebo::physics::WorldState, 908
- GetModelStates
 - gazebo::physics::WorldState, 908
- GetModelTransform
 - gazebo::common::SkeletonNode, 727
- GetModelVisualAt
 - gazebo::rendering::Scene, 671
- GetModels

- Common, 35
- gazebo::physics::World, 898
- GetMovableType
 - gazebo::rendering::DynamicLines, 264
- getMovableType
 - gazebo::rendering::DynamicLines, 264
- GetMsgType
 - gazebo::transport::CallbackHelper, 156
 - gazebo::transport::CallbackHelperT, 159
 - gazebo::transport::Node, 529
 - gazebo::transport::Publication, 602
 - gazebo::transport::PublicationTransport, 606
 - gazebo::transport::Publisher, 608
 - gazebo::transport::RawCallbackHelper, 626
 - gazebo::transport::SubscribeOptions, 759
- GetMsgTypes
 - gazebo::msgs::MsgFactory, 512
- GetName
 - gazebo::common::Material, 451
 - gazebo::common::Mesh, 470
 - gazebo::common::NodeAnimation, 535
 - gazebo::common::SkeletonAnimation, 721
 - gazebo::common::SkeletonNode, 728
 - gazebo::physics::Base, 141
 - gazebo::physics::State, 745
 - gazebo::physics::World, 898
 - gazebo::rendering::Camera, 171
 - gazebo::rendering::Light, 407
 - gazebo::rendering::Scene, 671
 - gazebo::rendering::Visual, 876
 - gazebo::sensors::Sensor, 688
 - gazebo::util::DiagnosticTimer, 261
 - sdf::Element, 274
- GetNearClip
 - gazebo::rendering::Camera, 171
- GetNearestEntityBelow
 - gazebo::physics::Entity, 281
- GetNextElement
 - sdf::Element, 274
- GetNextFrame
 - gazebo::common::Video, 862
- GetNode
 - gazebo::transport::SubscribeOptions, 759
- GetNodeAssignment
 - gazebo::common::SubMesh, 754
- GetNodeAssignmentsCount
 - gazebo::common::SubMesh, 754
- GetNodeByHandle
 - gazebo::common::Skeleton, 715
- GetNodeById
 - gazebo::common::Skeleton, 715
- GetNodeByName
 - gazebo::common::Skeleton, 716
- GetNodeCount
 - gazebo::common::SkeletonAnimation, 721
 - gazebo::transport::Publication, 602
- GetNodePoseAt
 - gazebo::common::SkeletonAnimation, 721
- GetNodes
 - gazebo::common::Skeleton, 716
- GetNormal
 - gazebo::common::SubMesh, 754
 - gazebo::math::Vector3, 844
 - gazebo::physics::PlaneShape, 580
- GetNormalCount
 - gazebo::common::Mesh, 470
 - gazebo::common::SubMesh, 755
- GetNormalMap
 - gazebo::rendering::Visual, 876
- GetNumAnimations
 - gazebo::common::Skeleton, 716
- GetNumJoints
 - gazebo::common::Skeleton, 716
- GetNumNodes
 - gazebo::common::Skeleton, 716
- GetNumPoints
 - gazebo::math::RotationSpline, 655
- GetNumRawTrans
 - gazebo::common::SkeletonNode, 728
- GetNumVertNodeWeights
 - gazebo::common::Skeleton, 716
- GetOgreCamera
 - gazebo::rendering::Camera, 171
- GetOgrePaths
 - gazebo::common::SystemPaths, 772
- GetOgreTerrain
 - gazebo::rendering::Heightmap, 347
- GetOperationType
 - gazebo::rendering::DynamicRenderable, 268
- GetOrientation
 - gazebo::sensors::ImuSensor, 363
- GetOutgoingCount
 - gazebo::transport::Publisher, 608
- GetPSSMShadowCameraSetup
 - gazebo::rendering::RTShaderSystem, 661
- GetParent
 - gazebo::common::SkeletonNode, 728
 - gazebo::physics::Base, 141
 - gazebo::physics::Joint, 387
 - gazebo::rendering::Projector, 598
 - gazebo::rendering::Visual, 876
 - sdf::Element, 274
- GetParentId
 - gazebo::physics::Base, 141
- GetParentJointsLinks
 - gazebo::physics::Link, 421
- GetParentModel
 - gazebo::physics::Entity, 282

- GetParentName
 - gazebo::sensors::Sensor, 688
- GetPath
 - gazebo::common::Mesh, 470
- GetPauseTime
 - gazebo::physics::World, 898
- GetPaused
 - gazebo::common::LogRecord, 443
- GetPerpendicular
 - gazebo::math::Vector3, 844
- GetPhysicsEngine
 - gazebo::physics::World, 899
- GetPhysicsUpdateMutex
 - gazebo::physics::PhysicsEngine, 563
- GetPitch
 - gazebo::common::Image, 359
 - gazebo::math::Quaternion, 615
- GetPitchNode
 - gazebo::rendering::Camera, 171
- GetPixel
 - gazebo::common::Image, 359
- GetPixelFormat
 - gazebo::common::Image, 360
- GetPluginCount
 - gazebo::physics::Model, 486
- GetPluginPaths
 - gazebo::common::SystemPaths, 772
- GetPoint
 - gazebo::math::RotationSpline, 655
 - gazebo::math::Spline, 740
 - gazebo::rendering::DynamicLines, 265
- GetPointCount
 - gazebo::math::Spline, 741
 - gazebo::rendering::DynamicLines, 265
- GetPointSize
 - gazebo::common::Material, 451
- GetPos
 - gazebo::physics::HeightmapShape, 350
- GetPose
 - gazebo::physics::CollisionState, 203
 - gazebo::physics::Inertial, 368
 - gazebo::physics::LinkState, 434
 - gazebo::physics::ModelState, 501
 - gazebo::rendering::Visual, 877
 - gazebo::sensors::Sensor, 688
- GetPoseAt
 - gazebo::common::SkeletonAnimation, 721
- GetPoseAtX
 - gazebo::common::SkeletonAnimation, 722
- GetPosition
 - gazebo::rendering::Light, 407
 - gazebo::rendering::Visual, 877
- GetPrevMsg
 - gazebo::transport::Publisher, 608
- GetPrimitiveType
 - gazebo::common::SubMesh, 755
- GetPrincipalMoments
 - gazebo::physics::Inertial, 368
- GetProductsofInertia
 - gazebo::physics::Inertial, 368
- GetRGBData
 - gazebo::common::Image, 360
- GetRadius
 - gazebo::physics::CylinderShape, 248
 - gazebo::physics::SphereShape, 738
- GetRandSeed
 - gazebo::common::LogPlay, 438
- GetRange
 - gazebo::physics::MultiRayShape, 523
 - gazebo::sensors::GpuRaySensor, 330
 - gazebo::sensors::RaySensor, 630
- GetRangeCount
 - gazebo::sensors::GpuRaySensor, 330
 - gazebo::sensors::RaySensor, 631
- GetRangeCountRatio
 - gazebo::sensors::GpuRaySensor, 330
- GetRangeMax
 - gazebo::sensors::GpuRaySensor, 330
 - gazebo::sensors::RaySensor, 631
- GetRangeMin
 - gazebo::sensors::GpuRaySensor, 331
 - gazebo::sensors::RaySensor, 631
- GetRangeResolution
 - gazebo::sensors::GpuRaySensor, 331
 - gazebo::sensors::RaySensor, 631
- GetRanges
 - gazebo::sensors::GpuRaySensor, 331
 - gazebo::sensors::RaySensor, 631
- GetRawTransform
 - gazebo::common::SkeletonNode, 728
- GetRawTransforms
 - gazebo::common::SkeletonNode, 728
- GetRayCount
 - gazebo::sensors::GpuRaySensor, 331
 - gazebo::sensors::RaySensor, 631
- GetRayCountRatio
 - gazebo::sensors::GpuRaySensor, 331
- GetRealTime
 - gazebo::physics::State, 745
 - gazebo::physics::World, 899
- GetRelativeAngularAccel
 - gazebo::physics::Collision, 196
 - gazebo::physics::Entity, 282
 - gazebo::physics::Link, 422
 - gazebo::physics::Model, 486
- GetRelativeAngularVel
 - gazebo::physics::Collision, 196
 - gazebo::physics::Entity, 282

- gazebo::physics::Link, 422
- gazebo::physics::Model, 486
- GetRelativeForce
 - gazebo::physics::Link, 422
- GetRelativeLinearAccel
 - gazebo::physics::Collision, 196
 - gazebo::physics::Entity, 282
 - gazebo::physics::Link, 422
 - gazebo::physics::Model, 487
- GetRelativeLinearVel
 - gazebo::physics::Collision, 197
 - gazebo::physics::Entity, 282
 - gazebo::physics::Link, 422
 - gazebo::physics::Model, 487
- GetRelativePoints
 - gazebo::physics::RayShape, 637
- GetRelativePose
 - gazebo::physics::Entity, 283
- GetRelativeTorque
 - gazebo::physics::Link, 422
- GetRemoteAddress
 - gazebo::transport::Connection, 223
- GetRemoteHostname
 - gazebo::transport::Connection, 223
- GetRemotePort
 - gazebo::transport::Connection, 223
- GetRemoteSubscriptionCount
 - gazebo::transport::Publication, 602
- GetRemoteURI
 - gazebo::transport::Connection, 224
- getRenderOperation
 - gazebo::rendering::MovableText, 509
- GetRenderPathType
 - gazebo::rendering::RenderEngine, 642
- GetRenderRate
 - gazebo::rendering::Camera, 171
- GetRenderTexture
 - gazebo::rendering::Camera, 172
- GetRequired
 - sdf::Element, 274
 - sdf::Param, 552
- GetResRange
 - gazebo::physics::MultiRayShape, 523
- GetRetro
 - gazebo::physics::MultiRayShape, 523
 - gazebo::physics::RayShape, 637
 - gazebo::sensors::GpuRaySensor, 331
 - gazebo::sensors::RaySensor, 632
- GetRight
 - gazebo::rendering::Camera, 172
- GetRoll
 - gazebo::math::Quaternion, 616
- GetRootNode
 - gazebo::common::Skeleton, 717
- GetRootVisual
 - gazebo::rendering::Visual, 877
- GetRotation
 - gazebo::common::PoseKeyFrame, 596
 - gazebo::math::Matrix4, 462
 - gazebo::rendering::Visual, 877
- GetRounded
 - gazebo::math::Vector3, 844
- GetRunTime
 - gazebo::common::LogRecord, 443
- GetRunning
 - gazebo::common::LogRecord, 443
 - gazebo::common::Timer, 798
- GetSDF
 - gazebo::physics::Actor, 112
 - gazebo::physics::Base, 142
 - gazebo::physics::Model, 487
- GetSID
 - gazebo::common::NodeTransform, 539
- GetSORPGSIters
 - gazebo::physics::PhysicsEngine, 564
- GetSORPGSPreconIters
 - gazebo::physics::PhysicsEngine, 564
- GetSORPGSW
 - gazebo::physics::PhysicsEngine, 564
- GetSampleCount
 - gazebo::physics::MultiRayShape, 524
- GetSaveable
 - gazebo::physics::Base, 142
- GetScale
 - gazebo::rendering::Visual, 877
- GetScanResolution
 - gazebo::physics::MultiRayShape, 524
- GetScene
 - gazebo::rendering::Camera, 172
 - gazebo::rendering::RenderEngine, 642
 - gazebo::rendering::Visual, 877
- GetSceneCount
 - gazebo::rendering::RenderEngine, 642
- GetSceneNode
 - gazebo::rendering::Camera, 172
 - gazebo::rendering::Grid, 338
 - gazebo::rendering::Visual, 878
- GetScopedName
 - gazebo::physics::Base, 142
 - gazebo::sensors::Sensor, 688
- GetSeed
 - gazebo::math::Rand, 625
- GetSelectedEntity
 - gazebo::physics::World, 899
- GetSelectedVisual
 - gazebo::rendering::Scene, 671
- GetSelfCollide
 - gazebo::physics::Link, 423

- GetSensor
 - gazebo::sensors::SensorManager, 696
- GetSensorCount
 - gazebo::physics::Link, 423
 - gazebo::physics::Model, 487
- GetSensorName
 - gazebo::physics::Link, 423
- GetSensorTypes
 - gazebo::sensors::SensorFactory, 693
 - gazebo::sensors::SensorManager, 697
- GetSensors
 - gazebo::sensors::SensorManager, 696
- GetSet
 - sdf::Param, 552
- GetSetWorldPoseMutex
 - gazebo::physics::World, 899
- GetShadeMode
 - gazebo::common::Material, 451
- GetShaderType
 - gazebo::rendering::Visual, 878
- GetShadowsEnabled
 - gazebo::rendering::Scene, 671
- GetShape
 - gazebo::physics::Collision, 197
- GetShapeType
 - gazebo::physics::Collision, 197
- GetShininess
 - gazebo::common::Material, 451
- GetShowOnTop
 - gazebo::rendering::MovableText, 510
- GetSimTime
 - gazebo::physics::State, 746
 - gazebo::physics::World, 899
- GetSize
 - gazebo::math::Box, 148
 - gazebo::physics::BoxShape, 153
 - gazebo::physics::HeightmapShape, 351
 - gazebo::physics::PlaneShape, 580
 - gazebo::physics::TrimeshShape, 808
- GetSkeleton
 - gazebo::common::Mesh, 470
- GetSpaceWidth
 - gazebo::rendering::MovableText, 510
- GetSpecular
 - gazebo::common::Material, 451
- GetSpecularColor
 - gazebo::rendering::Light, 408
- GetSquaredLength
 - gazebo::math::Vector3, 844
 - gazebo::math::Vector4, 854
- getSquaredViewDepth
 - gazebo::rendering::DynamicRenderable, 268
 - gazebo::rendering::MovableText, 510
- GetStartTime
 - gazebo::physics::World, 899
- GetState
 - gazebo::physics::Collision, 197
- GetStepTime
 - gazebo::physics::PhysicsEngine, 564
- GetSubMesh
 - gazebo::common::Mesh, 470
- GetSubMeshCount
 - gazebo::common::Mesh, 471
- GetSubSampling
 - gazebo::physics::HeightmapShape, 351
- GetSum
 - gazebo::math::Vector3, 845
- GetSurface
 - gazebo::physics::Collision, 197
- GetTagPose
 - gazebo::sensors::RFIDTag, 647
- GetTangent
 - gazebo::math::Spline, 741
- GetTension
 - gazebo::math::Spline, 741
- GetTexCoord
 - gazebo::common::SubMesh, 755
- GetTexCoordCount
 - gazebo::common::Mesh, 471
 - gazebo::common::SubMesh, 755
- GetText
 - gazebo::rendering::MovableText, 510
- GetTextureHeight
 - gazebo::rendering::Camera, 172
- GetTextureImage
 - gazebo::common::Material, 451
- GetTextureWidth
 - gazebo::rendering::Camera, 172
- GetThreadPitch
 - gazebo::physics::ScrewJoint, 679
- GetTime
 - gazebo::common::Animation, 126
 - gazebo::common::KeyFrame, 403
 - gazebo::util::DiagnosticManager, 258, 259
- GetTimeAtX
 - gazebo::common::NodeAnimation, 535
- GetTimerCount
 - gazebo::util::DiagnosticManager, 259
- GetTopic
 - gazebo::sensors::CameraSensor, 187
 - gazebo::sensors::MultiCameraSensor, 517
 - gazebo::sensors::RaySensor, 632
 - gazebo::sensors::Sensor, 689
 - gazebo::transport::PublicationTransport, 606
 - gazebo::transport::Publisher, 608
 - gazebo::transport::SubscribeOptions, 759
 - gazebo::transport::Subscriber, 761
- getTopicMsgType

- Transport, 77
- GetTopicNamespace
 - gazebo::transport::Node, 529
- GetTopicNamespaces
 - gazebo::transport::ConnectionManager, 228
 - gazebo::transport::TopicManager, 802
- GetTransform
 - gazebo::common::SkeletonNode, 729
- GetTransforms
 - gazebo::common::SkeletonNode, 729
- GetTranslation
 - gazebo::common::PoseKeyFrame, 596
 - gazebo::math::Matrix4, 462
- GetTransparency
 - gazebo::common::Material, 452
 - gazebo::rendering::Visual, 878
- GetTransportCount
 - gazebo::transport::Publication, 602
- GetTriangleCount
 - gazebo::rendering::Camera, 173
 - gazebo::rendering::UserCamera, 817
 - gazebo::rendering::WindowManager, 890
- GetType
 - gazebo::common::NodeTransform, 539
 - gazebo::physics::Base, 142
 - gazebo::physics::PhysicsEngine, 564
 - gazebo::PluginT, 584
 - gazebo::rendering::Light, 408
 - gazebo::sensors::Sensor, 689
- GetTypeName
 - sdf::Param, 552
- GetTypeString
 - gazebo::rendering::FPSViewController, 318
 - gazebo::rendering::OrbitViewController, 547
 - gazebo::rendering::ViewController, 865
- GetURI
 - Common, 35
 - gazebo::physics::HeightmapShape, 351
- GetUp
 - gazebo::rendering::Camera, 173
- GetUpdatePeriod
 - gazebo::physics::PhysicsEngine, 565
- GetUpdateRate
 - gazebo::physics::PhysicsEngine, 565
 - gazebo::sensors::Sensor, 689
- GetUpperLimit
 - gazebo::physics::Joint, 387
- GetUserCamera
 - gazebo::rendering::Scene, 672
- GetUserCameraCount
 - gazebo::rendering::Scene, 672
- GetVFOV
 - gazebo::rendering::Camera, 173
- GetValue
 - gazebo::common::NumericKeyFrame, 545
 - sdf::Element, 274
 - sdf::ParamT, 557
- GetValueBool
 - sdf::Element, 274
- GetValueChar
 - sdf::Element, 274
- GetValueColor
 - sdf::Element, 274
- GetValueDouble
 - sdf::Element, 274
- GetValueFloat
 - sdf::Element, 274
- GetValueInt
 - sdf::Element, 274
- GetValuePose
 - sdf::Element, 274
- GetValueQuaternion
 - sdf::Element, 274
- GetValueString
 - sdf::Element, 274
- GetValueTime
 - sdf::Element, 274
- GetValueUInt
 - sdf::Element, 274
- GetValueVector2d
 - sdf::Element, 274
- GetValueVector3
 - sdf::Element, 274
- GetVelocity
 - gazebo::physics::Joint, 387
 - gazebo::physics::LinkState, 434
- GetVelocityLimit
 - gazebo::physics::Joint, 387
- GetVertFOV
 - gazebo::sensors::GpuRaySensor, 332
- GetVertHalfAngle
 - gazebo::sensors::GpuRaySensor, 332
- GetVertNodeWeight
 - gazebo::common::Skeleton, 717
- GetVertex
 - gazebo::common::SubMesh, 755
- GetVertexCount
 - gazebo::common::Mesh, 471
 - gazebo::common::SubMesh, 755
 - gazebo::physics::HeightmapShape, 351
- GetVertexIndex
 - gazebo::common::SubMesh, 756
- GetVerticalAngleMax
 - gazebo::sensors::GpuRaySensor, 332
 - gazebo::sensors::RaySensor, 632
- GetVerticalAngleMin
 - gazebo::sensors::GpuRaySensor, 332
 - gazebo::sensors::RaySensor, 632

- GetVerticalMaxAngle
 - gazebo::physics::MultiRayShape, 524
- GetVerticalMinAngle
 - gazebo::physics::MultiRayShape, 524
- GetVerticalRangeCount
 - gazebo::sensors::GpuRaySensor, 332
 - gazebo::sensors::RaySensor, 632
- GetVerticalRayCount
 - gazebo::sensors::GpuRaySensor, 332
 - gazebo::sensors::RaySensor, 633
- GetVerticalSampleCount
 - gazebo::physics::MultiRayShape, 524
- GetVerticalScanResolution
 - gazebo::physics::MultiRayShape, 524
- GetViewControllerTypeString
 - gazebo::rendering::UserCamera, 817
- GetViewport
 - gazebo::rendering::Camera, 173
- GetViewportHeight
 - gazebo::rendering::Camera, 173
- GetViewportWidth
 - gazebo::rendering::Camera, 173
- GetVisibilityFlags
 - gazebo::rendering::Visual, 878
- GetVisible
 - gazebo::rendering::Visual, 878
- GetVisual
 - gazebo::rendering::Scene, 672
 - gazebo::rendering::UserCamera, 817
- GetVisualAt
 - gazebo::rendering::Scene, 672
- GetVisualBelow
 - gazebo::rendering::Scene, 673
- GetVisualName
 - gazebo::rendering::SelectionObj, 682
- GetVisualize
 - gazebo::sensors::Sensor, 689
- GetVisualsBelowPoint
 - gazebo::rendering::Scene, 673
- GetWallTime
 - gazebo::common::Time, 780
 - gazebo::physics::State, 746
- GetWallTimeAsISOString
 - gazebo::common::Time, 780
- GetWidth
 - gazebo::common::Image, 360
 - gazebo::common::Video, 862
- GetWindow
 - gazebo::rendering::WindowManager, 890
- GetWindowId
 - gazebo::rendering::Camera, 174
- GetWorld
 - gazebo::physics::Base, 142
- GetWorldAngularAccel
 - gazebo::physics::Collision, 198
 - gazebo::physics::Entity, 283
 - gazebo::physics::Link, 423
 - gazebo::physics::Model, 487
- GetWorldAngularVel
 - gazebo::physics::Collision, 198
 - gazebo::physics::Entity, 283
 - gazebo::physics::Model, 488
- GetWorldCFM
 - gazebo::physics::PhysicsEngine, 565
- GetWorldCoGLinearVel
 - gazebo::physics::Link, 424
- GetWorldCoGPose
 - gazebo::physics::Link, 424
- GetWorldERP
 - gazebo::physics::PhysicsEngine, 565
- GetWorldForce
 - gazebo::physics::Link, 424
- GetWorldLinearAccel
 - gazebo::physics::Collision, 198
 - gazebo::physics::Entity, 283
 - gazebo::physics::Link, 424
 - gazebo::physics::Model, 488
- GetWorldLinearVel
 - gazebo::physics::Collision, 198
 - gazebo::physics::Entity, 283
 - gazebo::physics::Link, 424, 425
 - gazebo::physics::Model, 488
- GetWorldName
 - gazebo::sensors::Sensor, 689
- GetWorldPathExtension
 - gazebo::common::SystemPaths, 772
- GetWorldPointOnPlane
 - gazebo::rendering::Camera, 174
- GetWorldPose
 - gazebo::physics::Entity, 284
 - gazebo::rendering::Camera, 174
 - gazebo::rendering::Visual, 879
- GetWorldPosition
 - gazebo::rendering::Camera, 174
- GetWorldRotation
 - gazebo::rendering::Camera, 174
- GetWorldTorque
 - gazebo::physics::Link, 425
- getWorldTransforms
 - gazebo::rendering::MovableText, 510
- GetWorldVisual
 - gazebo::rendering::Scene, 673
- GetWrench
 - gazebo::physics::LinkState, 434
- GetXAxis
 - gazebo::math::Quaternion, 616
- GetXLength
 - gazebo::math::Box, 148

- GetYAxis
 - gazebo::math::Quaternion, 616
- GetYLength
 - gazebo::math::Box, 149
- GetYaw
 - gazebo::math::Quaternion, 616
- GetZAxis
 - gazebo::math::Quaternion, 616
- GetZLength
 - gazebo::math::Box, 149
- GetZValue
 - gazebo::rendering::Camera, 175
- globalEndPos
 - gazebo::physics::RayShape, 639
- globalStartPos
 - gazebo::physics::RayShape, 639
- google, 105
- google::protobuf, 105
- google::protobuf::compiler, 105
- google::protobuf::compiler::cpp, 105
- google::protobuf::compiler::cpp::GazeboGenerator, 319
 - ~GazeboGenerator, 320
 - GazeboGenerator, 320
 - Generate, 320
- GpuLaser
 - gazebo::rendering::GpuLaser, 322
- GpuLaser.hh, 960
- GpuLaserPtr
 - gazebo::rendering, 99
- GpuRaySensor
 - gazebo::sensors::GpuRaySensor, 327
- GpuRaySensor.hh, 961
- GpuRaySensor_V
 - gazebo::sensors, 102
- GpuRaySensorPtr
 - gazebo::sensors, 102
- Green
 - gazebo::common::Color, 215
- Grid
 - gazebo::rendering::Grid, 337
- Grid.hh, 962
- Gripper
 - gazebo::physics::Gripper, 340
- Gripper.hh, 962
- GtsSurface
 - MeshCSG.hh, 997
- GzTerrainMatGen
 - gazebo::rendering::GzTerrainMatGen, 346
- gzclr_end
 - Common, 31
- gzclr_start
 - Common, 31
- gzdbg
 - Common, 31
- gzerr
 - Common, 31
- gzlog
 - Common, 31
- gzmsg
 - Common, 31
- gzthrow
 - Common, 31
- gzwarn
 - Common, 32
- H_CENTER
 - gazebo::rendering::MovableText, 508
- H_LEFT
 - gazebo::rendering::MovableText, 508
- HEADER_LENGTH
 - Connection.hh, 940
- HEIGHTMAP_SHAPE
 - gazebo::physics::Base, 139
- HI_STOP
 - gazebo::physics::Joint, 380
- HINGE2_JOINT
 - gazebo::physics::Base, 139
- HINGE_JOINT
 - gazebo::physics::Base, 139
- handle
 - gazebo::common::SkeletonNode, 731
 - gazebo::PluginT, 584
- HandleData
 - gazebo::transport::CallbackHelper, 157
 - gazebo::transport::CallbackHelperT, 159
 - gazebo::transport::Node, 529
 - gazebo::transport::RawCallbackHelper, 626
 - gazebo::transport::SubscriptionTransport, 763
- HandleKeyPressEvent
 - gazebo::rendering::FPSViewController, 318
 - gazebo::rendering::GUIOverlay, 343
 - gazebo::rendering::OrbitViewController, 547
 - gazebo::rendering::UserCamera, 818
 - gazebo::rendering::ViewController, 866
- HandleKeyReleaseEvent
 - gazebo::rendering::FPSViewController, 318
 - gazebo::rendering::GUIOverlay, 344
 - gazebo::rendering::OrbitViewController, 547
 - gazebo::rendering::UserCamera, 818
 - gazebo::rendering::ViewController, 866
- HandleMouseEvent
 - gazebo::rendering::FPSViewController, 318
 - gazebo::rendering::GUIOverlay, 344
 - gazebo::rendering::OrbitViewController, 548
 - gazebo::rendering::UserCamera, 818
 - gazebo::rendering::ViewController, 866
- HasAttachedObject
 - gazebo::rendering::Visual, 879

- HasAttribute
 - sdf::Element, 274
- HasConnections
 - gazebo::transport::Publisher, 608
- HasElement
 - sdf::Element, 275
- HasElementDescription
 - sdf::Element, 275
- HasJointState
 - gazebo::physics::ModelState, 501
- HasLatchedSubscriber
 - gazebo::transport::Node, 530
- HasLinkState
 - gazebo::physics::ModelState, 501
- HasMesh
 - gazebo::common::MeshManager, 479
- HasModel
 - Common, 35
- HasModelState
 - gazebo::physics::WorldState, 908
- HasNode
 - gazebo::common::SkeletonAnimation, 722
- HasSkeleton
 - gazebo::common::Mesh, 471
- HasTransport
 - gazebo::transport::Publication, 603
- HasType
 - gazebo::physics::Base, 142
- HasVertex
 - gazebo::common::SubMesh, 756
- Heightmap
 - gazebo::rendering::Heightmap, 347
- Heightmap.hh, 964
- HeightmapShape
 - gazebo::physics::HeightmapShape, 349
- HeightmapShape.hh, 965
- HeightmapShapePtr
 - gazebo::physics, 95
- heights
 - gazebo::physics::HeightmapShape, 352
- Helpers.hh, 966
 - GZ_DBL_MAX, 968
 - GZ_DBL_MIN, 968
 - GZ_FLT_MAX, 968
 - GZ_FLT_MIN, 968
- hfov
 - gazebo::sensors::GpuRaySensor, 335
- Hide
 - gazebo::rendering::GUIOverlay, 344
- Hinge2Joint
 - gazebo::physics::Hinge2Joint, 353
- Hinge2Joint.hh, 968
- HingeJoint
 - gazebo::physics::HingeJoint, 355
- HingeJoint.hh, 970
- HorizAlign
 - gazebo::rendering::MovableText, 508
- horzElem
 - gazebo::physics::MultiRayShape, 525
 - gazebo::sensors::GpuRaySensor, 335
- horzHalfAngle
 - gazebo::sensors::GpuRaySensor, 335
- horzRangeCount
 - gazebo::sensors::GpuRaySensor, 335
- horzRayCount
 - gazebo::sensors::GpuRaySensor, 335
- IDENTITY
 - gazebo::math::Matrix4, 466
- IMAGE
 - gazebo::sensors, 102
- INTERSECTION
 - gazebo::common::MeshCSG, 473
- IOManager
 - gazebo::transport::IOManager, 374
- IOManager.hh, 974
- id
 - gazebo::common::SkeletonNode, 731
 - gazebo::physics::TrajectoryInfo, 805
- Image
 - gazebo::common::Image, 358
- Image.hh, 971
- imageFormat
 - gazebo::rendering::Camera, 183
- imageHeight
 - gazebo::rendering::Camera, 183
- imageWidth
 - gazebo::rendering::Camera, 183
- img
 - gazebo::physics::HeightmapShape, 352
- ImuSensor
 - gazebo::sensors::ImuSensor, 363
- ImuSensor.hh, 972
- ImuSensor_V
 - gazebo::sensors, 102
- ImuSensorPtr
 - gazebo::sensors, 102
- IncCount
 - gazebo::transport::IOManager, 375
- indexBufferCapacity
 - gazebo::rendering::DynamicRenderable, 269
- inertiaRatio
 - gazebo::physics::Joint, 392
- Inertial
 - gazebo::physics::Inertial, 366, 367
- inertial
 - gazebo::physics::Link, 430
- Inertial.hh, 973

InertialPtr
 gazebo::physics, 95

Init
 Common, 36
 gazebo::common::LogRecord, 443
 gazebo::common::PID, 573
 gazebo::Master, 446
 gazebo::ModelPlugin, 496
 gazebo::physics::Actor, 113
 gazebo::physics::Base, 143
 gazebo::physics::BoxShape, 153
 gazebo::physics::Collision, 198
 gazebo::physics::ContactManager, 236
 gazebo::physics::CylinderShape, 248
 gazebo::physics::Gripper, 341
 gazebo::physics::HeightmapShape, 351
 gazebo::physics::HingeJoint, 355
 gazebo::physics::Joint, 388
 gazebo::physics::Link, 425
 gazebo::physics::Model, 488
 gazebo::physics::MultiRayShape, 524
 gazebo::physics::PhysicsEngine, 565
 gazebo::physics::PlaneShape, 580
 gazebo::physics::RayShape, 637
 gazebo::physics::Road, 653
 gazebo::physics::Shape, 707
 gazebo::physics::SphereShape, 738
 gazebo::physics::TrimeshShape, 808
 gazebo::physics::World, 900
 gazebo::rendering::Camera, 175
 gazebo::rendering::DepthCamera, 253
 gazebo::rendering::DynamicRenderable, 268
 gazebo::rendering::FPSViewController, 319
 gazebo::rendering::GpuLaser, 323
 gazebo::rendering::Grid, 339
 gazebo::rendering::GUIOverlay, 344
 gazebo::rendering::OrbitViewController, 548
 gazebo::rendering::RenderEngine, 643
 gazebo::rendering::RTShaderSystem, 661
 gazebo::rendering::Scene, 673
 gazebo::rendering::SelectionObj, 683
 gazebo::rendering::UserCamera, 818
 gazebo::rendering::ViewController, 866
 gazebo::rendering::Visual, 879
 gazebo::rendering::WireBox, 892
 gazebo::SensorPlugin, 699
 gazebo::sensors::CameraSensor, 187
 gazebo::sensors::ContactSensor, 241
 gazebo::sensors::DepthCameraSensor, 256
 gazebo::sensors::GpuRaySensor, 333
 gazebo::sensors::ImuSensor, 363
 gazebo::sensors::MultiCameraSensor, 517
 gazebo::sensors::RaySensor, 633
 gazebo::sensors::RFIDSensor, 645
 gazebo::sensors::RFIDTag, 648
 gazebo::sensors::Sensor, 689
 gazebo::sensors::SensorManager, 697
 gazebo::Server, 700
 gazebo::SystemPlugin, 774
 gazebo::transport::ConnectionManager, 228
 gazebo::transport::Node, 530
 gazebo::transport::PublicationTransport, 606
 gazebo::transport::SubscribeOptions, 760
 gazebo::transport::SubscriptionTransport, 763
 gazebo::transport::TopicManager, 803
 gazebo::util::DiagnosticManager, 259
 gazebo::VisualPlugin, 888
 gazebo::WorldPlugin, 905
 Messages, 60

init
 gazebo, 83
 Rendering, 66
 sdf, 107
 Sensors, 71
 Transport, 77

init_world
 Classes for physics and dynamics, 45

init_worlds
 Classes for physics and dynamics, 45

initDoc
 sdf, 107

initFile
 sdf, 107

InitForThread
 gazebo::physics::PhysicsEngine, 565

InitModelDoc
 urdf2gazebo::URDF2Gazebo, 812

InitModelFile
 urdf2gazebo::URDF2Gazebo, 812

InitModelString
 urdf2gazebo::URDF2Gazebo, 812

initString
 sdf, 107

initXml
 sdf, 107

initialTransform
 gazebo::common::SkeletonNode, 731

initialized
 gazebo::rendering::Camera, 183

InsertElement
 sdf::Element, 275

InsertLatchedMsg
 gazebo::transport::Node, 530

InsertMesh
 gazebo::rendering::Visual, 879

InsertModelFile
 gazebo::physics::World, 900

InsertModelSDF

- gazebo::physics::World, 900
- InsertModelString
 - gazebo::physics::World, 900
- Instance
 - SingletonT, 712
- InternalError
 - gazebo::common::InternalError, 373
- Interpolate
 - gazebo::math::RotationSpline, 656
 - gazebo::math::Spline, 741
- interpolateX
 - gazebo::physics::Actor, 114
- invBindTransform
 - gazebo::common::SkeletonNode, 731
- Inverse
 - gazebo::math::Matrix4, 462
- Invert
 - gazebo::math::Quaternion, 616
- is_stopped
 - Transport, 77
- IsActive
 - gazebo::physics::Actor, 113
 - gazebo::rendering::SelectionObj, 683
 - gazebo::sensors::CameraSensor, 187
 - gazebo::sensors::ContactSensor, 241
 - gazebo::sensors::MultiCameraSensor, 517
 - gazebo::sensors::RaySensor, 633
 - gazebo::sensors::Sensor, 690
- IsAdvertised
 - gazebo::transport::TopicManager, 803
- IsAffine
 - gazebo::math::Matrix4, 462
- IsAnimating
 - gazebo::rendering::Camera, 175
- IsBool
 - sdf::Param, 552
- IsCanonicalLink
 - gazebo::physics::Entity, 284
- IsChar
 - sdf::Param, 552
- IsColor
 - sdf::Param, 552
- IsDouble
 - sdf::Param, 553
- IsFinite
 - gazebo::math::Pose, 589
 - gazebo::math::Quaternion, 617
 - gazebo::math::Vector2d, 824
 - gazebo::math::Vector2i, 832
 - gazebo::math::Vector3, 845
 - gazebo::math::Vector4, 855
- IsFloat
 - sdf::Param, 553
- IsHorizontal
 - gazebo::sensors::GpuRaySensor, 333
- isHorizontal
 - gazebo::sensors::GpuRaySensor, 335
- IsInitialized
 - gazebo::rendering::Camera, 175
 - gazebo::rendering::GUIOverlay, 344
- IsInt
 - sdf::Param, 553
- IsJoint
 - gazebo::common::SkeletonNode, 729
- IsLoaded
 - gazebo::physics::World, 900
- IsLocal
 - gazebo::transport::CallbackHelper, 157
 - gazebo::transport::CallbackHelperT, 159
 - gazebo::transport::RawCallbackHelper, 627
 - gazebo::transport::SubscriptionTransport, 763
- IsOpen
 - gazebo::common::LogPlay, 438
 - gazebo::transport::Connection, 224
- IsPaused
 - gazebo::physics::World, 900
- IsPlaceable
 - gazebo::physics::Collision, 198
- IsPlane
 - gazebo::rendering::Visual, 879
- IsPose
 - sdf::Param, 553
- isPowerOfTwo
 - Math, 51
- IsQuaternion
 - sdf::Param, 553
- IsRegistered
 - gazebo::physics::PhysicsFactory, 570
- IsRootNode
 - gazebo::common::SkeletonNode, 729
- IsRunning
 - gazebo::transport::ConnectionManager, 228
- IsSelected
 - gazebo::physics::Base, 143
- IsStatic
 - gazebo::physics::Entity, 284
 - gazebo::rendering::Visual, 880
- IsStr
 - sdf::Param, 553
- IsTime
 - sdf::Param, 553
- IsUInt
 - sdf::Param, 553
- IsValidFilename
 - gazebo::common::MeshManager, 479
- IsVector2d
 - sdf::Param, 553
- IsVector2i

- sdf::Param, 553
- IsVector3
 - sdf::Param, 553
- IsVisible
 - gazebo::rendering::Camera, 175, 176
- IsZero
 - gazebo::physics::CollisionState, 203
 - gazebo::physics::JointState, 397
 - gazebo::physics::LinkState, 435
 - gazebo::physics::ModelState, 502
 - gazebo::physics::WorldState, 908
- isnan
 - Math, 50
- JOINT
 - gazebo::common::SkeletonNode, 725
 - gazebo::physics::Base, 139
- Joint
 - gazebo::physics::Joint, 380
- Joint.hh, 975
 - MAX_JOINT_AXIS, 976
- Joint_V
 - gazebo::physics, 95
- JointController
 - gazebo::physics::JointController, 394
- JointController.hh, 976
- JointController_V
 - gazebo::physics, 95
- JointControllerPtr
 - gazebo::physics, 95
- JointPtr
 - gazebo::physics, 95
- JointState
 - gazebo::physics::JointState, 396
- JointState.hh, 977
- JointVisual
 - gazebo::rendering::JointVisual, 400
- JointVisual.hh, 978
- JointVisualPtr
 - gazebo::rendering, 99
- JointWrench.hh, 979
- kd
 - gazebo::physics::SurfaceParams, 766
- key
 - sdf::Param, 554
- KeyFrame
 - gazebo::common::KeyFrame, 403
- KeyFrame.hh, 980
- KeyFrame_V
 - gazebo::common::Animation, 124
- keyFrames
 - gazebo::common::Animation, 126
 - gazebo::common::NodeAnimation, 536
- kp
 - gazebo::physics::SurfaceParams, 766
- L_INT16
 - gazebo::common::Image, 357
- L_INT8
 - gazebo::common::Image, 357
- LEFT
 - gazebo::common::MouseEvent, 504
- LIGHT
 - gazebo::physics::Base, 139
- LINE_MAX_LEN
 - STLLoader.hh, 1078
- LINES
 - gazebo::common::SubMesh, 751
- LINESTRIPS
 - gazebo::common::SubMesh, 751
- LINK
 - gazebo::physics::Base, 138
- LINUX
 - SystemPaths.hh, 1085
- LO_STOP
 - gazebo::physics::Joint, 380
- Lap
 - gazebo::util::DiagnosticManager, 259
 - gazebo::util::DiagnosticTimer, 261
- LaserVisual
 - gazebo::rendering::LaserVisual, 404
- LaserVisual.hh, 981
- LaserVisualPtr
 - gazebo::rendering, 99
- lastMeasurementTime
 - gazebo::sensors::Sensor, 692
- lastPos
 - gazebo::physics::Actor, 114
- lastRenderWallTime
 - gazebo::rendering::Camera, 183
- lastScriptTime
 - gazebo::physics::Actor, 114
- lastTraj
 - gazebo::physics::Actor, 114
- lastUpdateTime
 - gazebo::sensors::Sensor, 692
- latching
 - gazebo::transport::CallbackHelper, 157
- length
 - gazebo::common::Animation, 127
 - gazebo::common::NodeAnimation, 536
 - gazebo::common::SkeletonAnimation, 722
- Light
 - gazebo::rendering::Light, 407
- Light.hh, 982
- LightFromSDF
 - Messages, 60
- LightPtr

- gazebo::rendering, 99
- LightingModel
 - gazebo::rendering::RTShaderSystem, 659
- linearAccel
 - gazebo::physics::Link, 430
- Link
 - gazebo::physics::Link, 416
- link
 - gazebo::physics::Collision, 200
- Link.hh, 983
- Link_V
 - gazebo::physics, 95
- LinkPtr
 - gazebo::physics, 95
- LinkState
 - gazebo::physics::LinkState, 432
- LinkState.hh, 984
- Listen
 - gazebo::transport::Connection, 224
- Load
 - gazebo::common::BVHLoader, 154
 - gazebo::common::ColladaLoader, 191
 - gazebo::common::Image, 360
 - gazebo::common::MeshLoader, 475
 - gazebo::common::MeshManager, 479
 - gazebo::common::STLLoader, 748
 - gazebo::common::Video, 862
 - gazebo::ModelPlugin, 496
 - gazebo::physics::Actor, 113
 - gazebo::physics::BallJoint, 135
 - gazebo::physics::Base, 143
 - gazebo::physics::Collision, 199
 - gazebo::physics::CollisionState, 203
 - gazebo::physics::Entity, 284
 - gazebo::physics::Gripper, 341
 - gazebo::physics::HeightmapShape, 351
 - gazebo::physics::Hinge2Joint, 354
 - gazebo::physics::HingeJoint, 355
 - gazebo::physics::Inertial, 369
 - gazebo::physics::Joint, 388
 - gazebo::physics::JointState, 398
 - gazebo::physics::Link, 425
 - gazebo::physics::LinkState, 435
 - gazebo::physics::Model, 488
 - gazebo::physics::ModelState, 502
 - gazebo::physics::PhysicsEngine, 566
 - gazebo::physics::Road, 653
 - gazebo::physics::ScrewJoint, 679
 - gazebo::physics::SliderJoint, 734
 - gazebo::physics::State, 746
 - gazebo::physics::SurfaceParams, 765
 - gazebo::physics::UniversalJoint, 810
 - gazebo::physics::World, 901
 - gazebo::physics::WorldState, 909
 - gazebo::rendering::ArrowVisual, 129
 - gazebo::rendering::AxisVisual, 132
 - gazebo::rendering::Camera, 176
 - gazebo::rendering::CameraVisual, 190
 - gazebo::rendering::COMVisual, 217
 - gazebo::rendering::DepthCamera, 253
 - gazebo::rendering::GpuLaser, 323
 - gazebo::rendering::Heightmap, 347
 - gazebo::rendering::JointVisual, 400
 - gazebo::rendering::Light, 408
 - gazebo::rendering::MovableText, 510
 - gazebo::rendering::Projector, 598, 599
 - gazebo::rendering::RenderEngine, 643
 - gazebo::rendering::Road2d, 654
 - gazebo::rendering::Scene, 673, 674
 - gazebo::rendering::UserCamera, 818
 - gazebo::rendering::Visual, 880
 - gazebo::SensorPlugin, 699
 - gazebo::sensors::CameraSensor, 188
 - gazebo::sensors::ContactSensor, 241
 - gazebo::sensors::DepthCameraSensor, 256
 - gazebo::sensors::GpuRaySensor, 333
 - gazebo::sensors::ImuSensor, 364
 - gazebo::sensors::MultiCameraSensor, 517
 - gazebo::sensors::RaySensor, 633
 - gazebo::sensors::RFIDSensor, 645
 - gazebo::sensors::RFIDTag, 648
 - gazebo::sensors::Sensor, 690
 - gazebo::SystemPlugin, 774
 - gazebo::VisualPlugin, 888
 - gazebo::WorldPlugin, 905
- load
 - Classes for physics and dynamics, 45
 - gazebo, 83
 - Rendering, 66
 - Sensors, 72
- load_world
 - Classes for physics and dynamics, 45
- load_worlds
 - Classes for physics and dynamics, 45
- LoadFile
 - gazebo::Server, 700
- LoadFromMsg
 - gazebo::rendering::Heightmap, 347
 - gazebo::rendering::Light, 408
 - gazebo::rendering::Visual, 880
- LoadJoints
 - gazebo::physics::Model, 489
- LoadLayout
 - gazebo::rendering::GUIOverlay, 344
- LoadPlugin
 - gazebo::physics::World, 901
 - gazebo::rendering::Visual, 880
- LoadPlugins

- gazebo::physics::Model, 489
- LoadString
 - gazebo::Server, 701
- LocalPublish
 - gazebo::transport::Publication, 603
- Log
 - Common, 36
- LogPlay.hh, 986
- LogRecord.hh, 987
 - GZ_LOG_VERSION, 989
- Logplay, 439
- loop
 - gazebo::common::Animation, 127
 - gazebo::physics::Actor, 114
- lowerLimit
 - gazebo::physics::Joint, 392
- m
 - gazebo::math::Matrix3, 459
 - gazebo::math::Matrix4, 466
- MAP_SHAPE
 - gazebo::physics::Base, 139
- MATRIX
 - gazebo::common::NodeTransform, 538
- MAX_COLLIDE_RETURNS
 - Contact.hh, 944
- MAX_CONTACT_JOINTS
 - Contact.hh, 944
- MAX_JOINT_AXIS
 - Joint.hh, 976
- MIDDLE
 - gazebo::common::MouseEvent, 504
- MODEL
 - gazebo::physics::Base, 138
- MODEL_PLUGIN
 - Common, 32
- MODULATE
 - gazebo::common::Material, 449
- MOVE
 - gazebo::common::MouseEvent, 505
- MSleep
 - gazebo::common::Time, 781
- MULTIRAY_SHAPE
 - gazebo::physics::Base, 139
- mainLink
 - gazebo::physics::Actor, 114
- mainpage.html, 989
- MakeStatic
 - gazebo::rendering::Visual, 880
- MapShape.hh, 989
- Master
 - gazebo::Master, 445
- Master.hh, 990
- Material
 - gazebo::common::Material, 449
- Material.hh, 991, 993
- Math, 48
 - clamp, 50
 - equal, 50
 - isPowerOfTwo, 51
 - isnan, 50
 - max, 51
 - mean, 51
 - min, 51
 - NAN_D, 53
 - NAN_I, 53
 - parseFloat, 52
 - parseInt, 52
 - precision, 52
 - variance, 52
- MathTypes.hh, 993
- Matrix3
 - gazebo::math::Matrix3, 457
- Matrix3.hh, 994
- Matrix4
 - gazebo::math::Matrix4, 461
- Matrix4.hh, 994
- max
 - gazebo::math::Box, 151
 - Math, 51
- maxVel
 - gazebo::physics::SurfaceParams, 767
- mean
 - Math, 51
- Merge
 - gazebo::math::Box, 149
- Mesh
 - gazebo::common::Mesh, 468
- mesh
 - gazebo::physics::Actor, 114
 - gazebo::physics::TrimeshShape, 809
- Mesh.hh, 995
- MeshCSG
 - gazebo::common::MeshCSG, 473
- MeshCSG.hh, 997
 - GPtrArray, 997
 - GtsSurface, 997
- MeshLoader
 - gazebo::common::MeshLoader, 474
- MeshLoader.hh, 998
- MeshManager.hh, 999
- MeshShapePtr
 - gazebo::physics, 95
- Messages, 54
 - Convert, 56–59
 - CreateRequest, 59
 - FogFromSDF, 59
 - GUIFromSDF, 60

- GZ_REGISTER_STATIC_MSG, 56
- GetHeader, 59
- Init, 60
- LightFromSDF, 60
- SceneFromSDF, 60
- Set, 61, 62
- Stamp, 62
- TrackVisualFromSDF, 63
- VisualFromSDF, 63
- MicToNano
 - gazebo::common::Time, 780
- MilToNano
 - gazebo::common::Time, 780
- min
 - gazebo::math::Box, 151
 - Math, 51
- minDepth
 - gazebo::physics::SurfaceParams, 767
- Model
 - gazebo::physics::Model, 483
- model
 - gazebo::physics::Joint, 392
- Model.hh, 1000
- Model_V
 - gazebo::physics, 95
- ModelDatabase.hh, 1002
 - GZ_MODEL_DB_MANIFEST_FILENAME, 1002
 - GZ_MODEL_MANIFEST_FILENAME, 1002
- modelPathsFromEnv
 - gazebo::common::SystemPaths, 773
- ModelPlugin
 - gazebo::ModelPlugin, 495
- ModelPluginPtr
 - gazebo, 83
- ModelPtr
 - gazebo::physics, 96
- ModelState
 - gazebo::physics::ModelState, 498
- ModelState.hh, 1003
- modelTransform
 - gazebo::common::SkeletonNode, 731
- MouseEvent
 - gazebo::common::MouseEvent, 505
- MouseEvent.hh, 1005
- MovableText
 - gazebo::rendering::MovableText, 508
- MovableText.hh, 1006
- moveScale
 - gazebo::common::MouseEvent, 505
- MoveToPosition
 - gazebo::rendering::Camera, 176
 - gazebo::rendering::UserCamera, 818
 - gazebo::rendering::Visual, 880
- MoveToPositions
 - gazebo::rendering::Camera, 176
 - gazebo::rendering::Visual, 881
- MoveToVisual
 - gazebo::rendering::UserCamera, 819
- Moved
 - gazebo::rendering::WindowManager, 890
- MsgFactory.hh, 1006
- MsgFactoryFn
 - gazebo::msgs, 91
- msgs.hh, 1007
- mu1
 - gazebo::physics::SurfaceParams, 767
- mu2
 - gazebo::physics::SurfaceParams, 767
- MultiCameraSensor
 - gazebo::sensors::MultiCameraSensor, 515
- MultiCameraSensor.hh, 1010
- MultiRayShape
 - gazebo::physics::MultiRayShape, 521
- MultiRayShape.hh, 1010
- MultiRayShapePtr
 - gazebo::physics, 96
- NAN_D
 - Math, 53
- NAN_I
 - Math, 53
- NO_BUTTON
 - gazebo::common::MouseEvent, 504
- NO_EVENT
 - gazebo::common::MouseEvent, 505
- NODE
 - gazebo::common::SkeletonNode, 725
- NONE
 - gazebo::rendering::RenderEngine, 641
- NRealGen
 - gazebo::math, 89
- NSleep
 - gazebo::common::Time, 781
- NULL
 - CommonTypes.hh, 937
- name
 - gazebo::common::Animation, 127
 - gazebo::common::Material, 455
 - gazebo::common::NodeAnimation, 536
 - gazebo::common::SkeletonAnimation, 723
 - gazebo::common::SkeletonNode, 731
 - gazebo::physics::State, 747
 - gazebo::rendering::Camera, 183
 - sdf::Plugin, 582
- near
 - gazebo::sensors::GpuRaySensor, 335
- NewContact
 - gazebo::physics::ContactManager, 237

- newData
 - gazebo::rendering::Camera, 183
- newImageFrame
 - gazebo::rendering::Camera, 183
- newLaserScans
 - gazebo::physics::MultiRayShape, 525
- NewMsg
 - gazebo::msgs::MsgFactory, 513
- NewPhysicsEngine
 - gazebo::physics::PhysicsFactory, 570
- NewSensor
 - gazebo::sensors::SensorFactory, 694
- Node
 - gazebo::transport::Node, 528
- node
 - gazebo::physics::Entity, 288
 - gazebo::physics::PhysicsEngine, 569
 - gazebo::sensors::Sensor, 692
- Node.hh, 1012
- NodeAnimation
 - gazebo::common::NodeAnimation, 533
- nodeIndex
 - gazebo::common::NodeAssignment, 537
- NodeMap
 - gazebo::common, 86
- NodeMapIter
 - gazebo::common, 86
- NodePtr
 - gazebo::transport, 104
- NodeTransform
 - gazebo::common::NodeTransform, 539
- nodes
 - gazebo::common::Skeleton, 718
- normal
 - gazebo::math::Plane, 577
- NormalRealDist
 - gazebo::math, 89
- Normalize
 - gazebo::math::Angle, 118
 - gazebo::math::Quaternion, 617
 - gazebo::math::Vector2d, 824
 - gazebo::math::Vector2i, 832
 - gazebo::math::Vector3, 845
 - gazebo::math::Vector4, 855
- normals
 - gazebo::physics::Contact, 234
- notifyRenderSingleObject
 - gazebo::rendering::GpuLaser, 323
- nsec
 - gazebo::common::Time, 796
- NullStream
 - Common, 32
- NumericAnimation
 - gazebo::common::NumericAnimation, 543
- NumericAnimationPtr
 - gazebo::common, 86
- NumericKeyFrame
 - gazebo::common::NumericKeyFrame, 544
- ORDER_MAX
 - STLLoader.hh, 1078
- OTHER
 - gazebo::sensors, 102
- offset
 - gazebo::physics::MultiRayShape, 525
- Ogre, 106
- ogre, 106
- ogre_gazebo.h, 1013
- ogrePathsFromEnv
 - gazebo::common::SystemPaths, 773
- oldAction
 - gazebo::physics::Actor, 114
- onAnimationComplete
 - gazebo::rendering::Camera, 183
- OnPhysicsMsg
 - gazebo::physics::PhysicsEngine, 566
- OnPoseChange
 - gazebo::physics::Entity, 284
 - gazebo::physics::Link, 425
 - gazebo::physics::Model, 489
 - gazebo::rendering::Light, 408
- OnRequest
 - gazebo::physics::PhysicsEngine, 566
- Open
 - gazebo::common::LogPlay, 439
- operator<
 - gazebo::common::Time, 788, 789
 - gazebo::math::Angle, 120
- operator<<
 - gazebo::common::Color, 215
 - gazebo::common::Exception, 316
 - gazebo::common::Material, 454
 - gazebo::common::Time, 795
 - gazebo::common::Timer, 798
 - gazebo::math::Angle, 122
 - gazebo::math::Box, 150
 - gazebo::math::Matrix3, 459
 - gazebo::math::Matrix4, 466
 - gazebo::math::Pose, 592
 - gazebo::math::Quaternion, 622
 - gazebo::math::Vector2d, 829
 - gazebo::math::Vector2i, 837
 - gazebo::math::Vector3, 850
 - gazebo::math::Vector4, 860
 - gazebo::physics::CollisionState, 204
 - gazebo::physics::Inertial, 372
 - gazebo::physics::JointState, 399
 - gazebo::physics::LinkState, 436

- gazebo::physics::ModelState, 503
- gazebo::physics::WorldState, 910
- sdf::ParamT, 558
- operator<=
 - gazebo::common::Time, 789, 790
 - gazebo::math::Angle, 121
- operator>
 - gazebo::common::Time, 792, 793
 - gazebo::math::Angle, 121
- operator>>
 - gazebo::common::Color, 215
 - gazebo::common::Time, 795
 - gazebo::math::Angle, 122
 - gazebo::math::Pose, 592
 - gazebo::math::Quaternion, 622
 - gazebo::math::Vector2d, 829
 - gazebo::math::Vector2i, 838
 - gazebo::math::Vector3, 851
 - gazebo::math::Vector4, 860
- operator>=
 - gazebo::common::Time, 793, 794
 - gazebo::math::Angle, 121
- operator*
 - gazebo::common::Color, 209, 210
 - gazebo::common::NodeTransform, 540
 - gazebo::common::Time, 782, 783
 - gazebo::math::Angle, 118
 - gazebo::math::Matrix4, 462, 463
 - gazebo::math::Pose, 589
 - gazebo::math::Quaternion, 617, 618
 - gazebo::math::Vector2d, 824
 - gazebo::math::Vector2i, 833
 - gazebo::math::Vector3, 845, 846, 850
 - gazebo::math::Vector4, 855, 856
 - sdf::ParamT, 557
- operator*=
 - gazebo::common::Color, 210
 - gazebo::common::Time, 783, 784
 - gazebo::math::Angle, 119
 - gazebo::math::Quaternion, 618
 - gazebo::math::Vector2d, 825
 - gazebo::math::Vector2i, 833, 834
 - gazebo::math::Vector3, 846
 - gazebo::math::Vector4, 856
- operator()
 - gazebo::common::NodeTransform, 539
 - gazebo::event::EventT, 308–310
- operator+
 - gazebo::common::Color, 210
 - gazebo::common::Time, 784
 - gazebo::math::Angle, 119
 - gazebo::math::Box, 149
 - gazebo::math::Pose, 589
 - gazebo::math::Quaternion, 618
- gazebo::math::Vector2d, 825
- gazebo::math::Vector2i, 834
- gazebo::math::Vector3, 846
- gazebo::math::Vector4, 857
- gazebo::physics::CollisionState, 203
- gazebo::physics::Inertial, 369
- gazebo::physics::JointState, 398
- gazebo::physics::LinkState, 435
- gazebo::physics::ModelState, 502
- gazebo::physics::WorldState, 909
- operator+=
 - gazebo::common::Color, 211
 - gazebo::common::Time, 785
 - gazebo::math::Angle, 119
 - gazebo::math::Box, 149
 - gazebo::math::Pose, 590
 - gazebo::math::Quaternion, 618
 - gazebo::math::Vector2d, 826
 - gazebo::math::Vector2i, 834
 - gazebo::math::Vector3, 847
 - gazebo::math::Vector4, 857
 - gazebo::physics::Inertial, 369
- operator-
 - gazebo::common::Color, 211
 - gazebo::common::Time, 785, 786
 - gazebo::math::Angle, 119
 - gazebo::math::Box, 150
 - gazebo::math::Pose, 590
 - gazebo::math::Quaternion, 619
 - gazebo::math::Vector2d, 826
 - gazebo::math::Vector2i, 834
 - gazebo::math::Vector3, 847
 - gazebo::math::Vector4, 857
 - gazebo::physics::CollisionState, 204
 - gazebo::physics::JointState, 398
 - gazebo::physics::LinkState, 435
 - gazebo::physics::ModelState, 502
 - gazebo::physics::State, 746
 - gazebo::physics::WorldState, 909
- operator-=
 - gazebo::common::Color, 211
 - gazebo::common::Time, 786, 787
 - gazebo::math::Angle, 120
 - gazebo::math::Pose, 590
 - gazebo::math::Quaternion, 619
 - gazebo::math::Vector2d, 826
 - gazebo::math::Vector2i, 835
 - gazebo::math::Vector3, 847
 - gazebo::math::Vector4, 857
- operator/
 - gazebo::common::Color, 212
 - gazebo::common::Time, 787
 - gazebo::math::Angle, 120
 - gazebo::math::Vector2d, 826, 827

- gazebo::math::Vector2i, 835
- gazebo::math::Vector3, 847, 848
- gazebo::math::Vector4, 858
- operator/=
 - gazebo::common::Color, 212
 - gazebo::common::Time, 788
 - gazebo::math::Angle, 120
 - gazebo::math::Vector2d, 827
 - gazebo::math::Vector2i, 835, 836
 - gazebo::math::Vector3, 848
 - gazebo::math::Vector4, 858, 859
- operator=
 - gazebo::common::Color, 212
 - gazebo::common::PID, 573
 - gazebo::common::Time, 790, 791
 - gazebo::math::Box, 150
 - gazebo::math::Matrix4, 463
 - gazebo::math::Plane, 577
 - gazebo::math::Quaternion, 619
 - gazebo::math::Vector2d, 827, 828
 - gazebo::math::Vector2i, 836
 - gazebo::math::Vector3, 848, 849
 - gazebo::math::Vector4, 859
 - gazebo::physics::CollisionState, 204
 - gazebo::physics::Contact, 233, 234
 - gazebo::physics::Inertial, 369
 - gazebo::physics::JointState, 398
 - gazebo::physics::JointWrench, 401
 - gazebo::physics::LinkState, 436
 - gazebo::physics::ModelState, 503
 - gazebo::physics::State, 746
 - gazebo::physics::WorldState, 909
- operator==
 - gazebo::common::Color, 213
 - gazebo::common::Time, 791, 792
 - gazebo::math::Angle, 121
 - gazebo::math::Box, 150
 - gazebo::math::Matrix3, 457
 - gazebo::math::Matrix4, 464
 - gazebo::math::Pose, 591
 - gazebo::math::Quaternion, 619
 - gazebo::math::Vector2d, 828
 - gazebo::math::Vector2i, 837
 - gazebo::math::Vector3, 849
 - gazebo::math::Vector4, 859
 - gazebo::physics::Base, 143
- operator[]
 - gazebo::common::Color, 213
 - gazebo::math::Matrix3, 457, 458
 - gazebo::math::Matrix4, 464
 - gazebo::math::Vector2d, 828
 - gazebo::math::Vector2i, 837
 - gazebo::math::Vector3, 849
 - gazebo::math::Vector4, 859
- OrbitViewController
 - gazebo::rendering::OrbitViewController, 547
- OrbitViewController.hh, 1014
- PHONG
 - gazebo::common::Material, 449
- PID
 - gazebo::common::PID, 572
- PID.hh, 1024
- PIXEL_FORMAT_COUNT
 - gazebo::common::Image, 358
- PLANE_SHAPE
 - gazebo::physics::Base, 139
- POINTS
 - gazebo::common::SubMesh, 751
- PRESS
 - gazebo::common::MouseEvent, 505
- Param
 - sdf::Param, 551
- Param.hh, 1014
- Param_V
 - gazebo::common, 86
 - sdf, 107
- ParamPtr
 - sdf, 107
- ParamT
 - sdf::ParamT, 556
- ParamT< T >, 555
- parent
 - gazebo::common::SkeletonNode, 732
 - gazebo::physics::Base, 146
 - gazebo::rendering::Visual, 886
- parentEntity
 - gazebo::physics::Entity, 288
- parentLink
 - gazebo::physics::Joint, 392
- parentName
 - gazebo::sensors::Sensor, 692
- ParseArgs
 - gazebo::Server, 701
- parseFloat
 - Math, 52
- parseInt
 - Math, 52
- parser.hh, 1016
- parser_urdf.hh, 1017
- pathLength
 - gazebo::physics::Actor, 114
- pause
 - gazebo::event::Events, 304
- pause_incoming
 - Transport, 77
- pause_world
 - Classes for physics and dynamics, 45

- pause_worlds
 - Classes for physics and dynamics, 46
- PauseIncoming
 - gazebo::transport::TopicManager, 803
- Physics.hh, 1018
- PhysicsEngine
 - gazebo::physics::PhysicsEngine, 561
- PhysicsEngine.hh, 1019
- PhysicsEnginePtr
 - gazebo::physics, 96
- PhysicsFactory.hh, 1021
- PhysicsFactoryFn
 - Classes for physics and dynamics, 44
- physicsSub
 - gazebo::physics::PhysicsEngine, 569
- PhysicsTypes.hh, 1022
 - GZ_ALL_COLLIDE, 1024
 - GZ_FIXED_COLLIDE, 1024
 - GZ_GHOST_COLLIDE, 1024
 - GZ_NONE_COLLIDE, 1024
 - GZ_SENSOR_COLLIDE, 1024
- physicsUpdateMutex
 - gazebo::physics::PhysicsEngine, 569
- pitchNode
 - gazebo::rendering::Camera, 183
- PixelFormat
 - gazebo::common::Image, 357
- PixelFormatNames
 - Common, 36
- PlaceOnEntity
 - gazebo::physics::Entity, 285
- PlaceOnNearestEntityBelow
 - gazebo::physics::Entity, 285
- placeable
 - gazebo::physics::Collision, 201
- Plane
 - gazebo::math::Plane, 576
- Plane.hh, 1025
- PlaneShape
 - gazebo::physics::PlaneShape, 579
- PlaneShape.hh, 1026
- Play
 - gazebo::physics::Actor, 113
- playStartTime
 - gazebo::physics::Actor, 115
- Plugin
 - sdf::Plugin, 582
- Plugin.hh, 1028, 1032
- pluginPathsFromEnv
 - gazebo::common::SystemPaths, 773
- PluginType
 - Common, 32
- plugins
 - gazebo::sensors::Sensor, 692
- pointSize
 - gazebo::common::Material, 455
- points
 - gazebo::math::RotationSpline, 657
 - gazebo::math::Spline, 743
- pos
 - gazebo::common::MouseEvent, 505
 - gazebo::math::Pose, 592
- Pose
 - gazebo::math::Pose, 586, 587
- pose
 - gazebo::sensors::Sensor, 692
- Pose.hh, 1032
- PoseAnimation
 - gazebo::common::PoseAnimation, 594
- PoseAnimationPtr
 - gazebo::common, 86
- PoseKeyFrame
 - gazebo::common::PoseKeyFrame, 596
- poseMsg
 - gazebo::physics::Entity, 288
- poseSub
 - gazebo::sensors::Sensor, 692
- positions
 - gazebo::physics::Contact, 235
- PostRender
 - gazebo::rendering::Camera, 177
 - gazebo::rendering::DepthCamera, 253
 - gazebo::rendering::GpuLaser, 323
 - gazebo::rendering::UserCamera, 819
- postRender
 - gazebo::event::Events, 304
- PreRender
 - gazebo::rendering::Scene, 674
- preRender
 - gazebo::event::Events, 304
- precision
 - Math, 52
- PrepareHardwareBuffers
 - gazebo::rendering::DynamicRenderable, 268
- pressPos
 - gazebo::common::MouseEvent, 505
- prevAnimTime
 - gazebo::rendering::Camera, 183
- prevAnimationTime
 - gazebo::physics::Entity, 288
- prevFrameTime
 - gazebo::physics::Actor, 115
- prevPos
 - gazebo::common::MouseEvent, 506
- PrimitiveType
 - gazebo::common::SubMesh, 751
- Print
 - gazebo::common::Exception, 316

- gazebo::physics::Base, 144
- sdf::Plugin, 582
- print_version
 - gazebo, 83
- PrintDescription
 - sdf::Element, 275
 - sdf::SDF, 681
- PrintDoc
 - sdf::SDF, 681
- PrintDocLeftPane
 - sdf::Element, 275
- PrintDocRightPane
 - sdf::Element, 275
- PrintEntityTree
 - gazebo::physics::World, 901
- PrintSceneGraph
 - gazebo::rendering::Scene, 674
- PrintSource
 - gazebo::common::NodeTransform, 540
- PrintTransforms
 - gazebo::common::Skeleton, 717
- PrintUsage
 - gazebo::Server, 701
- PrintValues
 - sdf::Element, 275
 - sdf::SDF, 681
- PrintWiki
 - sdf::Element, 275
 - sdf::SDF, 681
- ProcessIncoming
 - gazebo::transport::Node, 530
- ProcessMsg
 - gazebo::physics::BoxShape, 153
 - gazebo::physics::Collision, 199
 - gazebo::physics::CylinderShape, 249
 - gazebo::physics::HeightmapShape, 352
 - gazebo::physics::Inertial, 370
 - gazebo::physics::Link, 425
 - gazebo::physics::Model, 489
 - gazebo::physics::MultiRayShape, 525
 - gazebo::physics::PlaneShape, 580
 - gazebo::physics::RayShape, 638
 - gazebo::physics::Shape, 707
 - gazebo::physics::SphereShape, 738
 - gazebo::physics::SurfaceParams, 765
 - gazebo::physics::TrimeshShape, 808
- ProcessNodes
 - gazebo::transport::TopicManager, 803
- ProcessPublishers
 - gazebo::transport::Node, 530
- ProcessWriteQueue
 - gazebo::transport::Connection, 224
- Projector
 - gazebo::rendering::Projector, 598
- Projector.hh, 1033
- Publication
 - gazebo::transport::Publication, 601
- Publication.hh, 1034
- PublicationPtr
 - gazebo::transport, 104
- PublicationTransport
 - gazebo::transport::PublicationTransport, 605
- PublicationTransport.hh, 1035
- PublicationTransportPtr
 - gazebo::transport, 104
- Publish
 - gazebo::transport::Publication, 603
 - gazebo::transport::Publisher, 608, 609
 - gazebo::transport::TopicManager, 803
- PublishContacts
 - gazebo::physics::ContactManager, 237
- PublishModelPose
 - gazebo::physics::World, 901
- Publisher
 - gazebo::transport::Publisher, 607
- Publisher.hh, 1037
- PublisherPtr
 - gazebo::transport, 104
- Purple
 - gazebo::common::Color, 215
- Quaternion
 - gazebo::math::Quaternion, 612, 613
- Quaternion.hh, 1039
- r
 - gazebo::common::Color, 215
- R_FLOAT16
 - gazebo::common::Image, 358
- R_FLOAT32
 - gazebo::common::Image, 358
- RAY
 - gazebo::sensors, 102
- RAY_SHAPE
 - gazebo::physics::Base, 139
- RELEASE
 - gazebo::common::MouseEvent, 505
- RENDER_PATH_COUNT
 - gazebo::rendering::RenderEngine, 641
- RENDERING_LINE_LIST
 - gazebo::rendering, 99
- RENDERING_LINE_STRIP
 - gazebo::rendering, 99
- RENDERING_MESH_RESOURCE
 - gazebo::rendering, 99
- RENDERING_POINT_LIST
 - gazebo::rendering, 99
- RENDERING_TRIANGLE_FAN
 - gazebo::rendering, 99

RENDERING_TRIANGLE_LIST
 gazebo::rendering, 99
 RENDERING_TRIANGLE_STRIP
 gazebo::rendering, 99
 REPLACE
 gazebo::common::Material, 449
 RFIDSensor
 gazebo::sensors::RFIDSensor, 645
 RFIDSensor.hh, 1048
 RFIDSensor_V
 gazebo::sensors, 102
 RFIDSensorPtr
 gazebo::sensors, 102
 RFIDTag
 gazebo::sensors::RFIDTag, 647
 RFIDTag.hh, 1048
 RFIDTag_V
 gazebo::sensors, 102
 RFIDTagPtr
 gazebo::sensors, 102
 RFIDTagVisual
 gazebo::rendering::RFIDTagVisual, 649
 RFIDTagVisual.hh, 1049
 RFIDTagVisualPtr
 gazebo::rendering, 99
 RFIDVisual
 gazebo::rendering::RFIDVisual, 651
 RFIDVisual.hh, 1050
 RFIDVisualPtr
 gazebo::rendering, 99
 RGB_FLOAT16
 gazebo::common::Image, 358
 RGB_FLOAT32
 gazebo::common::Image, 358
 RGB_INT16
 gazebo::common::Image, 357
 RGB_INT32
 gazebo::common::Image, 357
 RGB_INT8
 gazebo::common::Image, 357
 RGBA
 gazebo::common::Color, 207
 RGBA_INT8
 gazebo::common::Image, 357
 RIGHT
 gazebo::common::MouseEvent, 504
 ROTATE
 gazebo::common::NodeTransform, 538
 RTShaderSystem.hh, 1053
 Radian
 gazebo::math::Angle, 122
 Rand.hh, 1040
 rangeCountRatio
 gazebo::sensors::GpuRaySensor, 335
 rangeElem
 gazebo::physics::MultiRayShape, 525
 gazebo::sensors::GpuRaySensor, 335
 RawCallbackHelper
 gazebo::transport::RawCallbackHelper, 626
 rawNW
 gazebo::common::Skeleton, 718
 RawNodeAnim
 gazebo::common, 86
 RawNodeWeights
 gazebo::common, 86
 RawSkeletonAnim
 gazebo::common, 86
 rawTransforms
 gazebo::common::SkeletonNode, 732
 rayCountRatio
 gazebo::sensors::GpuRaySensor, 335
 rayElem
 gazebo::physics::MultiRayShape, 525
 RaySensor
 gazebo::sensors::RaySensor, 629
 RaySensor.hh, 1041
 RaySensor_V
 gazebo::sensors, 102
 RaySensorPtr
 gazebo::sensors, 102
 RayShape
 gazebo::physics::RayShape, 636
 RayShape.hh, 1042
 RayShapePtr
 gazebo::physics, 96
 rays
 gazebo::physics::MultiRayShape, 526
 Read
 gazebo::transport::Connection, 224
 ReadCallback
 gazebo::transport::Connection, 221
 readDoc
 sdf, 107
 readFile
 sdf, 108
 readString
 sdf, 108
 readXml
 sdf, 108
 realTime
 gazebo::common::UpdateInfo, 811
 gazebo::physics::State, 747
 RecalcTangents
 gazebo::math::RotationSpline, 656
 gazebo::math::Spline, 741
 RecalculateMatrix
 gazebo::common::NodeTransform, 540
 RecalculateNormals

- gazebo::common::Mesh, 471
- gazebo::common::SubMesh, 756
- Red
 - gazebo::common::Color, 216
- RegisterAll
 - gazebo::physics::PhysicsFactory, 571
 - gazebo::sensors::SensorFactory, 694
- RegisterMsg
 - gazebo::msgs::MsgFactory, 513
- RegisterPhysicsEngine
 - gazebo::physics::PhysicsFactory, 571
- RegisterSensor
 - gazebo::sensors::SensorFactory, 694
- RegisterTopicNamespace
 - gazebo::transport::ConnectionManager, 228
 - gazebo::transport::TopicManager, 804
- relativeEndPos
 - gazebo::physics::RayShape, 639
- relativeStartPos
 - gazebo::physics::RayShape, 639
- Remove
 - gazebo::common::LogRecord, 443
- remove_scene
 - Rendering, 66
- remove_sensor
 - Sensors, 72
- remove_sensors
 - Sensors, 72
- remove_worlds
 - Classes for physics and dynamics, 46
- RemoveCallback
 - gazebo::transport::Node, 530
- RemoveChild
 - gazebo::physics::Base, 144
 - gazebo::physics::Model, 489
 - sdf::Element, 275
- RemoveChildJoint
 - gazebo::physics::Link, 426
- RemoveChildren
 - gazebo::physics::Base, 144
- RemoveConnection
 - gazebo::transport::ConnectionManager, 228
- RemoveFromParent
 - sdf::Element, 275
- RemoveNode
 - gazebo::transport::TopicManager, 804
- RemoveParentJoint
 - gazebo::physics::Link, 426
- RemovePlugin
 - gazebo::physics::World, 901
 - gazebo::rendering::Visual, 881
- RemoveScene
 - gazebo::rendering::RenderEngine, 643
 - gazebo::rendering::RTShaderSystem, 661
- removeScene
 - gazebo::rendering::Events, 292
- RemoveSensor
 - gazebo::sensors::SensorManager, 697
- RemoveSensors
 - gazebo::sensors::SensorManager, 697
- RemoveShadows
 - gazebo::rendering::RTShaderSystem, 661
- RemoveSubscription
 - gazebo::transport::Publication, 603
- RemoveTransport
 - gazebo::transport::Publication, 604
- RemoveVisual
 - gazebo::rendering::Scene, 674
- Render
 - gazebo::rendering::Camera, 177
- render
 - gazebo::event::Events, 304
- RenderEngine.hh, 1043
- RenderEvents.hh, 1044
- RenderImpl
 - gazebo::rendering::Camera, 177
- RenderOpType
 - gazebo::rendering, 99
- RenderPathType
 - gazebo::rendering::RenderEngine, 641
- renderTarget
 - gazebo::rendering::Camera, 184
- renderTexture
 - gazebo::rendering::Camera, 184
- RenderTypes.hh, 1046
 - GZ_VISIBILITY_ALL, 1047
 - GZ_VISIBILITY_GUI, 1047
 - GZ_VISIBILITY_NOT_SELECTABLE, 1047
 - GZ_VISIBILITY_SELECTION, 1047
- Rendering, 64
 - create_scene, 66
 - fini, 66
 - get_scene, 66
 - init, 66
 - load, 66
 - remove_scene, 66
- Rendering.hh, 1045
- request
 - Transport, 77
- requestNoReply
 - Transport, 78
- requestPub
 - gazebo::physics::Entity, 288
- requestSub
 - gazebo::physics::PhysicsEngine, 569
- requests
 - gazebo::rendering::Camera, 184
- required

- sdf::Param, 554
- Rescale
 - gazebo::common::Image, 360
- Reset
 - gazebo::common::Color, 213
 - gazebo::common::PID, 573
 - gazebo::common::SkeletonNode, 729
 - gazebo::math::Pose, 591
 - gazebo::ModelPlugin, 496
 - gazebo::physics::Base, 144
 - gazebo::physics::Contact, 234
 - gazebo::physics::Entity, 285
 - gazebo::physics::Inertial, 370
 - gazebo::physics::Joint, 388
 - gazebo::physics::JointController, 394
 - gazebo::physics::Link, 426
 - gazebo::physics::Model, 489
 - gazebo::physics::PhysicsEngine, 566
 - gazebo::physics::World, 902
 - gazebo::SensorPlugin, 699
 - gazebo::SystemPlugin, 774
 - gazebo::VisualPlugin, 888
 - gazebo::WorldPlugin, 905
 - sdf::Element, 276
 - sdf::Param, 553
 - sdf::ParamT, 557
- ResetCount
 - gazebo::physics::ContactManager, 237
- ResetEntities
 - gazebo::physics::World, 902
- ResetLastUpdateTime
 - gazebo::sensors::Sensor, 690
- ResetLastUpdateTimeS
 - gazebo::sensors::SensorManager, 697
- ResetPhysicsStates
 - gazebo::physics::Link, 426
- ResetTime
 - gazebo::physics::World, 902
- Resize
 - gazebo::rendering::GUIOverlay, 345
 - gazebo::rendering::UserCamera, 819
 - gazebo::rendering::WindowManager, 891
- responsePub
 - gazebo::physics::PhysicsEngine, 569
- Road, 651
 - gazebo::physics::Road, 652
- Road.hh, 1050
- Road2d
 - gazebo::rendering::Road2d, 653
- Road2d.hh, 1052
- RoadPtr
 - gazebo::physics, 96
- root
 - gazebo::common::Skeleton, 718
 - gazebo::rendering::RenderEngine, 643
 - sdf::SDF, 681
- rot
 - gazebo::math::Pose, 593
- Rotate
 - gazebo::physics::Inertial, 370
- rotate
 - gazebo::common::PoseKeyFrame, 597
- RotatePitch
 - gazebo::rendering::Camera, 177
- RotatePositionAboutOrigin
 - gazebo::math::Pose, 591
- RotateVector
 - gazebo::math::Quaternion, 620
- RotateVectorReverse
 - gazebo::math::Quaternion, 620
- RotateYaw
 - gazebo::rendering::Camera, 177
- RotationSpline
 - gazebo::math::RotationSpline, 655
- RotationSpline.hh, 1052
- Round
 - gazebo::math::Pose, 591
 - gazebo::math::Quaternion, 620
 - gazebo::math::Vector3, 849
- Run
 - gazebo::Master, 446
 - gazebo::physics::World, 902
 - gazebo::sensors::SensorManager, 697
 - gazebo::Server, 701
 - gazebo::transport::ConnectionManager, 228
- run
 - gazebo, 83
 - Sensors, 72
 - Transport, 78
- run_once
 - Sensors, 72
- run_threads
 - Sensors, 72
- run_world
 - Classes for physics and dynamics, 46
- run_worlds
 - Classes for physics and dynamics, 46
- RunOnce
 - gazebo::Master, 446
- RunThread
 - gazebo::Master, 446
- RunThreads
 - gazebo::sensors::SensorManager, 697
- RunUpdate
 - gazebo::transport::ConnectionManager, 229
- SCALE
 - gazebo::common::NodeTransform, 538

- SCREW_JOINT
 - gazebo::physics::Base, 139
- SCROLL
 - gazebo::common::MouseEvent, 505
- SDF
 - sdf::SDF, 681
- SDF.hh, 1057
 - SDF_VERSION, 1058
- SDF_VERSION
 - SDF.hh, 1058
- SDFPtr
 - sdf, 107
- SENSOR_PLUGIN
 - Common, 32
- SHADE_COUNT
 - gazebo::common::Material, 449
- SHAPE
 - gazebo::physics::Base, 139
- SLIDER_JOINT
 - gazebo::physics::Base, 139
- SM2Profile
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 736
- SPHERE_SHAPE
 - gazebo::physics::Base, 139
- SSLM_NormalMapLightingObjectSpace
 - gazebo::rendering::RTShaderSystem, 659
- SSLM_NormalMapLightingTangentSpace
 - gazebo::rendering::RTShaderSystem, 659
- SSLM_PerPixelLighting
 - gazebo::rendering::RTShaderSystem, 659
- SSLM_PerVertexLighting
 - gazebo::rendering::RTShaderSystem, 659
- STLLoader
 - gazebo::common::STLLoader, 748
- STLLoader.hh, 1076
 - COR3_MAX, 1077
 - FACE_MAX, 1077
 - LINE_MAX_LEN, 1078
 - ORDER_MAX, 1078
- STOP_CFM
 - gazebo::physics::Joint, 380
- STOP_ERP
 - gazebo::physics::Joint, 380
- SUSPENSION_CFM
 - gazebo::physics::Joint, 379
- SUSPENSION_ERP
 - gazebo::physics::Joint, 379
- SYSTEM_PLUGIN
 - Common, 32
- Save
 - gazebo::physics::World, 902
- saveCount
 - gazebo::rendering::Camera, 184
- SaveFrame
 - gazebo::rendering::Camera, 177, 178
 - gazebo::sensors::CameraSensor, 188
 - gazebo::sensors::DepthCameraSensor, 256
 - gazebo::sensors::MultiCameraSensor, 517
- saveFrameBuffer
 - gazebo::rendering::Camera, 184
- SavePNG
 - gazebo::common::Image, 361
- Scale
 - gazebo::common::Mesh, 471
 - gazebo::common::NodeAnimation, 535
 - gazebo::common::Skeleton, 717
 - gazebo::common::SkeletonAnimation, 722
 - gazebo::common::SubMesh, 756
 - gazebo::math::Quaternion, 620
- scale
 - gazebo::physics::HeightmapShape, 352
- ScaleXAxis
 - gazebo::rendering::AxisVisual, 132
- ScaleYAxis
 - gazebo::rendering::AxisVisual, 132
- ScaleZAxis
 - gazebo::rendering::AxisVisual, 133
- scanElem
 - gazebo::physics::MultiRayShape, 526
 - gazebo::sensors::GpuRaySensor, 336
- Scene
 - gazebo::rendering::Scene, 665
- scene
 - gazebo::rendering::Camera, 184
 - gazebo::rendering::Visual, 886
- Scene.hh, 1054
- SceneFromSDF
 - Messages, 60
- sceneNode
 - gazebo::rendering::Camera, 184
 - gazebo::rendering::Visual, 887
- ScenePtr
 - gazebo::rendering, 99
- ScrewJoint
 - gazebo::physics::ScrewJoint, 678
- ScrewJoint.hh, 1055
- scriptLength
 - gazebo::physics::Actor, 115
- scroll
 - gazebo::common::MouseEvent, 506
- sdf, 106
 - addNestedModel, 107
 - copyChildren, 107
 - ElementPtr, 107
 - ElementPtr_V, 107
 - gazebo::physics::Base, 146
 - gazebo::physics::PhysicsEngine, 569

- gazebo::rendering::Camera, 184
- gazebo::sensors::Sensor, 692
- init, 107
- initDoc, 107
- initFile, 107
- initString, 107
- initXml, 107
- Param_V, 107
- ParamPtr, 107
- readDoc, 107
- readFile, 108
- readString, 108
- readXml, 108
- SDFPtr, 107
- sdf.hh, 1056
- sdf::Converter, 245
 - Convert, 246
- sdf::Element, 269
 - ~Element, 272
 - AddAttribute, 272
 - AddElement, 272
 - AddElementDescription, 272
 - AddValue, 272
 - ClearElements, 272
 - Clone, 272
 - Copy, 273
 - Element, 272
 - GetAttribute, 273
 - GetAttributeCount, 273
 - GetAttributeSet, 273
 - GetCopyChildren, 273
 - GetDescription, 273
 - GetElement, 273
 - GetElementDescription, 273
 - GetElementDescriptionCount, 273
 - GetElementImpl, 274
 - GetFirstElement, 274
 - GetInclude, 274
 - GetName, 274
 - GetNextElement, 274
 - GetParent, 274
 - GetRequired, 274
 - GetValue, 274
 - GetValueBool, 274
 - GetValueChar, 274
 - GetValueColor, 274
 - GetValueDouble, 274
 - GetValueFloat, 274
 - GetValueInt, 274
 - GetValuePose, 274
 - GetValueQuaternion, 274
 - GetValueString, 274
 - GetValueTime, 274
 - GetValueUInt, 274
 - GetValueVector2d, 274
 - GetValueVector3, 274
 - HasAttribute, 274
 - HasElement, 275
 - HasElementDescription, 275
 - InsertElement, 275
 - PrintDescription, 275
 - PrintDocLeftPane, 275
 - PrintDocRightPane, 275
 - PrintValues, 275
 - PrintWiki, 275
 - RemoveChild, 275
 - RemoveFromParent, 275
 - Reset, 276
 - Set, 276
 - SetCopyChildren, 276
 - SetDescription, 276
 - SetInclude, 276
 - SetName, 276
 - SetParent, 276
 - SetRequired, 276
 - ToString, 276
 - Update, 277
- sdf::Param, 549
 - ~Param, 551
 - Clone, 551
 - description, 554
 - Get, 551, 552
 - GetAsString, 552
 - GetDefaultAsString, 552
 - GetDescription, 552
 - GetKey, 552
 - GetRequired, 552
 - GetSet, 552
 - GetTypeName, 552
 - IsBool, 552
 - IsChar, 552
 - IsColor, 552
 - IsDouble, 553
 - IsFloat, 553
 - IsInt, 553
 - IsPose, 553
 - IsQuaternion, 553
 - IsStr, 553
 - IsTime, 553
 - IsUInt, 553
 - IsVector2d, 553
 - IsVector2i, 553
 - IsVector3, 553
 - key, 554
 - Param, 551
 - required, 554
 - Reset, 553
 - Set, 553, 554

- set, 554
- SetDescription, 554
- SetFromString, 554
- SetUpdateFunc, 554
- typeName, 554
- Update, 554
- updateFunc, 555
- sdf::ParamT
 - ~ParamT, 556
 - Clone, 557
 - defaultValue, 558
 - GetAsString, 557
 - GetDefaultAsString, 557
 - GetDefaultValue, 557
 - GetValue, 557
 - operator<<, 558
 - operator*, 557
 - ParamT, 556
 - Reset, 557
 - Set, 557
 - SetFromString, 557
 - SetValue, 557
 - Update, 558
 - value, 558
- sdf::ParamT< T >, 555
- sdf::Plugin, 581
 - Clear, 582
 - data, 582
 - filename, 582
 - name, 582
 - Plugin, 582
 - Print, 582
- sdf::SDF, 680
 - PrintDescription, 681
 - PrintDoc, 681
 - PrintValues, 681
 - PrintWiki, 681
 - root, 681
 - SDF, 681
 - SetFromString, 681
 - ToString, 681
 - version, 681
 - Write, 681
- sec
 - gazebo::common::Time, 796
- SecToNano
 - gazebo::common::Time, 794
- SelectVisual
 - gazebo::rendering::Scene, 674
- SelectionObj
 - gazebo::rendering::SelectionObj, 682
- SelectionObj.hh, 1058
- SendMessage
 - gazebo::transport::Publisher, 609
- Sensor
 - gazebo::sensors::Sensor, 686
- Sensor.hh, 1059
- Sensor_V
 - gazebo::sensors, 102
- SensorCategory
 - gazebo::sensors, 102
- SensorFactor, 693
- SensorFactory.hh, 1060
- SensorFactoryFn
 - gazebo::sensors, 102
- SensorManager.hh, 1061
- SensorPlugin
 - gazebo::SensorPlugin, 699
- SensorPluginPtr
 - gazebo, 83
- SensorPtr
 - gazebo::sensors, 102
- SensorTypes.hh, 1064
- Sensors, 69
 - create_sensor, 71
 - fini, 71
 - GZ_REGISTER_STATIC_SENSOR, 70
 - get_sensor, 71
 - init, 71
 - load, 72
 - remove_sensor, 72
 - remove_sensors, 72
 - run, 72
 - run_once, 72
 - run_threads, 72
 - stop, 72
- Sensors.hh, 1062
- SensorsInitialized
 - gazebo::sensors::SensorManager, 697
- Server
 - gazebo::Server, 700
- Server.hh, 1065
- Set
 - gazebo::common::Color, 213
 - gazebo::common::NodeTransform, 540
 - gazebo::common::Time, 794, 795
 - gazebo::math::Matrix4, 464
 - gazebo::math::Plane, 577
 - gazebo::math::Pose, 591, 592
 - gazebo::math::Quaternion, 620
 - gazebo::math::Vector2d, 828
 - gazebo::math::Vector2i, 837
 - gazebo::math::Vector3, 850
 - gazebo::math::Vector4, 860
- Messages, 61, 62
- sdf::Element, 276
- sdf::Param, 553, 554
- sdf::ParamT, 557

- set
 - sdf::Param, 554
- SetActive
 - gazebo::rendering::SelectionObj, 683
 - gazebo::sensors::DepthCameraSensor, 256
 - gazebo::sensors::Sensor, 690
- SetAltitude
 - gazebo::physics::PlaneShape, 580
- SetAmbient
 - gazebo::common::Material, 452
 - gazebo::rendering::Visual, 881
- SetAmbientColor
 - gazebo::rendering::Scene, 674
- SetAnchor
 - gazebo::physics::Joint, 389
 - gazebo::physics::ScrewJoint, 679
 - gazebo::physics::SliderJoint, 734
- SetAngle
 - gazebo::physics::Joint, 389
- SetAngleMax
 - gazebo::sensors::GpuRaySensor, 333
- SetAngleMin
 - gazebo::sensors::GpuRaySensor, 333
- SetAngularAccel
 - gazebo::physics::Link, 426
 - gazebo::physics::Model, 489
- SetAngularDamping
 - gazebo::physics::Link, 426
- SetAngularVel
 - gazebo::physics::Link, 427
 - gazebo::physics::Model, 490
- SetAnimation
 - gazebo::physics::Entity, 285
- SetAspectRatio
 - gazebo::rendering::Camera, 178
- SetAttenuation
 - gazebo::rendering::Light, 408
- SetAttribute
 - gazebo::physics::Joint, 389
- SetAutoCalculate
 - gazebo::math::RotationSpline, 657
 - gazebo::math::Spline, 742
- SetAutoDisable
 - gazebo::physics::Link, 427
 - gazebo::physics::Model, 490
- SetAutoDisableFlag
 - gazebo::physics::PhysicsEngine, 566
- SetAxis
 - gazebo::physics::BallJoint, 135
 - gazebo::physics::Joint, 389
- SetAxisMaterial
 - gazebo::rendering::AxisVisual, 133
- SetBackgroundColor
 - gazebo::rendering::Scene, 674
- SetBasePath
 - gazebo::common::LogRecord, 444
- SetBaseline
 - gazebo::rendering::MovableText, 510
- SetBindShapeTransform
 - gazebo::common::Skeleton, 717
- SetBlendFactors
 - gazebo::common::Material, 452
- SetBlendMode
 - gazebo::common::Material, 452
- SetCallbackId
 - gazebo::transport::Subscriber, 761
- SetCamera
 - gazebo::rendering::WindowManager, 891
- SetCanonicalLink
 - gazebo::physics::Entity, 285
- SetCaptureData
 - gazebo::rendering::Camera, 178
- SetCastShadows
 - gazebo::rendering::Light, 409
 - gazebo::rendering::Visual, 881
- SetCategoryBits
 - gazebo::physics::Collision, 199
- SetCellCount
 - gazebo::rendering::Grid, 339
- SetCellLength
 - gazebo::rendering::Grid, 339
- SetCharHeight
 - gazebo::rendering::MovableText, 511
- SetClipDist
 - gazebo::rendering::Camera, 178
- SetCmd
 - gazebo::common::PID, 573
- SetCmdMax
 - gazebo::common::PID, 574
- SetCmdMin
 - gazebo::common::PID, 574
- SetCoG
 - gazebo::physics::Inertial, 370
- SetCol
 - gazebo::math::Matrix3, 458
- SetCollideBits
 - gazebo::physics::Collision, 199
- SetCollideMode
 - gazebo::physics::Link, 427
 - gazebo::physics::Model, 490
- SetCollision
 - gazebo::physics::Collision, 199
- SetColor
 - gazebo::rendering::Grid, 339
 - gazebo::rendering::MovableText, 511
- SetComponent
 - gazebo::common::NodeTransform, 540
- SetContactMaxCorrectingVel

- gazebo::physics::PhysicsEngine, 566
- SetContactSurfaceLayer
 - gazebo::physics::PhysicsEngine, 567
- SetContactsEnabled
 - gazebo::physics::Collision, 200
- SetCopyChildren
 - sdf::Element, 276
- SetDGain
 - gazebo::common::PID, 574
- SetDamping
 - gazebo::physics::Joint, 389
- SetDepthTarget
 - gazebo::rendering::DepthCamera, 253
- SetDepthWrite
 - gazebo::common::Material, 452
- SetDescription
 - sdf::Element, 276
 - sdf::Param, 554
- SetDiffuse
 - gazebo::common::Material, 453
 - gazebo::rendering::Visual, 881
- SetDiffuseColor
 - gazebo::rendering::Light, 409
- SetDirection
 - gazebo::rendering::Light, 409
- SetDistance
 - gazebo::rendering::OrbitViewController, 548
- SetEmissive
 - gazebo::common::Material, 453
 - gazebo::rendering::LaserVisual, 405
 - gazebo::rendering::Visual, 882
- SetEnabled
 - gazebo::physics::Link, 427
 - gazebo::physics::Model, 490
 - gazebo::rendering::ContactVisual, 243
 - gazebo::rendering::Projector, 599
 - gazebo::rendering::ViewController, 867
- SetFiducial
 - gazebo::physics::RayShape, 638
- SetFilename
 - gazebo::physics::TrimeshShape, 808
- SetFocalPoint
 - gazebo::rendering::OrbitViewController, 548
 - gazebo::rendering::UserCamera, 819
- SetFog
 - gazebo::rendering::Scene, 675
- SetFontName
 - gazebo::rendering::MovableText, 511
- SetForce
 - gazebo::physics::Joint, 390
 - gazebo::physics::Link, 427
- SetFromABGR
 - gazebo::common::Color, 213
- SetFromARGB
 - gazebo::common::Color, 214
- SetFromAxes
 - gazebo::math::Matrix3, 458
- SetFromAxis
 - gazebo::math::Matrix3, 458
 - gazebo::math::Quaternion, 621
- SetFromBGRA
 - gazebo::common::Color, 214
- SetFromData
 - gazebo::common::Image, 361
- SetFromDegree
 - gazebo::math::Angle, 122
- SetFromEuler
 - gazebo::math::Quaternion, 621
- SetFromHSV
 - gazebo::common::Color, 214
- SetFromRGBA
 - gazebo::common::Color, 214
- SetFromRadian
 - gazebo::math::Angle, 122
- SetFromString
 - sdf::Param, 554
 - sdf::ParamT, 557
 - sdf::SDF, 681
- SetFromYUV
 - gazebo::common::Color, 214
- SetGravity
 - gazebo::physics::PhysicsEngine, 567
- SetGravityMode
 - gazebo::physics::Link, 427
 - gazebo::physics::Model, 490
- SetGrid
 - gazebo::rendering::Scene, 675
- SetHFOV
 - gazebo::rendering::Camera, 179
- SetHandle
 - gazebo::common::SkeletonNode, 729
- SetHeight
 - gazebo::rendering::Grid, 339
- SetHighStop
 - gazebo::physics::BallJoint, 135
 - gazebo::physics::Joint, 390
- SetHighlight
 - gazebo::rendering::SelectionObj, 683
- SetHighlighted
 - gazebo::rendering::Visual, 882
- SetIGain
 - gazebo::common::PID, 574
- SetIMax
 - gazebo::common::PID, 574
- SetIMin
 - gazebo::common::PID, 574
- SetIXX
 - gazebo::physics::Inertial, 371

- SetIXY
 - gazebo::physics::Inertial, 371
- SetIXZ
 - gazebo::physics::Inertial, 371
- SetIYY
 - gazebo::physics::Inertial, 371
- SetIYZ
 - gazebo::physics::Inertial, 371
- SetIZZ
 - gazebo::physics::Inertial, 371
- SetId
 - gazebo::common::SkeletonNode, 729
- SetImageHeight
 - gazebo::rendering::Camera, 179
- SetImageSize
 - gazebo::rendering::Camera, 179
- SetImageWidth
 - gazebo::rendering::Camera, 179
- SetInclude
 - sdf::Element, 276
- SetIndexCount
 - gazebo::common::SubMesh, 756
- SetInertiaMatrix
 - gazebo::physics::Inertial, 370
- SetInertial
 - gazebo::physics::Link, 428
- SetInitialRelativePose
 - gazebo::physics::Entity, 286
- SetInitialTransform
 - gazebo::common::SkeletonNode, 730
- SetInverseBindTransform
 - gazebo::common::SkeletonNode, 730
- SetJointAnimation
 - gazebo::physics::Model, 490
- SetJointPosition
 - gazebo::physics::JointController, 394
 - gazebo::physics::Model, 491
- SetJointPositions
 - gazebo::physics::JointController, 394
 - gazebo::physics::Model, 491
- SetKinematic
 - gazebo::physics::Link, 428
- SetLaserRetro
 - gazebo::physics::Collision, 200
 - gazebo::physics::Link, 428
 - gazebo::physics::Model, 491
- SetLength
 - gazebo::common::Animation, 126
 - gazebo::physics::CylinderShape, 249
 - gazebo::physics::RayShape, 638
- SetLightType
 - gazebo::rendering::Light, 409
- SetLighting
 - gazebo::common::Material, 453
- SetLineWidth
 - gazebo::rendering::Grid, 339
- SetLinearAccel
 - gazebo::physics::Link, 428
 - gazebo::physics::Model, 491
- SetLinearDamping
 - gazebo::physics::Link, 428
- SetLinearVel
 - gazebo::physics::Link, 428
 - gazebo::physics::Model, 492
- SetLinkWorldPose
 - gazebo::physics::Model, 492
- SetLocallyAdvertised
 - gazebo::transport::Publication, 604
- SetLowStop
 - gazebo::physics::BallJoint, 135
 - gazebo::physics::Joint, 390
- SetMass
 - gazebo::physics::Inertial, 372
- SetMaterial
 - gazebo::rendering::Visual, 882
- SetMaterialIndex
 - gazebo::common::SubMesh, 756
- SetMaxContacts
 - gazebo::physics::PhysicsEngine, 567
- SetMaxForce
 - gazebo::physics::Joint, 390
- SetModel
 - gazebo::physics::Joint, 390
- SetModelTransform
 - gazebo::common::SkeletonNode, 730
- SetName
 - gazebo::common::Mesh, 472
 - gazebo::common::NodeAnimation, 535
 - gazebo::common::SkeletonAnimation, 722
 - gazebo::common::SkeletonNode, 730
 - gazebo::physics::Base, 145
 - gazebo::physics::Entity, 286
 - gazebo::physics::State, 747
 - gazebo::rendering::Camera, 179
 - gazebo::rendering::Light, 409
 - gazebo::rendering::Visual, 882
 - sdf::Element, 276
- SetNormal
 - gazebo::common::SubMesh, 757
 - gazebo::physics::PlaneShape, 580
- SetNormalCount
 - gazebo::common::SubMesh, 757
- SetNormalMap
 - gazebo::rendering::Visual, 882
- SetNumVertAttached
 - gazebo::common::Skeleton, 718
- SetOperationType
 - gazebo::rendering::DynamicRenderable, 269

- SetPGain
 - gazebo::common::PID, 575
- SetParams
 - gazebo::Server, 701
- SetParent
 - gazebo::common::SkeletonNode, 730
 - gazebo::physics::Base, 145
 - gazebo::sensors::CameraSensor, 188
 - gazebo::sensors::DepthCameraSensor, 257
 - gazebo::sensors::Sensor, 691
 - sdf::Element, 276
- SetParentSensor
 - gazebo::rendering::GpuLaser, 323
- SetPath
 - gazebo::common::Mesh, 472
- SetPaused
 - gazebo::common::LogRecord, 444
 - gazebo::physics::World, 902
- SetPerPixelLighting
 - gazebo::rendering::RTShaderSystem, 661
- SetPoint
 - gazebo::rendering::DynamicLines, 265
- SetPointSize
 - gazebo::common::Material, 453
- SetPoints
 - gazebo::physics::RayShape, 638
- SetPose
 - gazebo::rendering::Visual, 882
- SetPosition
 - gazebo::rendering::Light, 409
 - gazebo::rendering::Visual, 883
- SetPrimitiveType
 - gazebo::common::SubMesh, 757
- SetPublication
 - gazebo::transport::Publisher, 609
- SetQuiet
 - Common, 36
- SetRadius
 - gazebo::physics::CylinderShape, 249
 - gazebo::physics::SphereShape, 738
- SetRange
 - gazebo::rendering::Light, 410
- SetRangeCount
 - gazebo::rendering::GpuLaser, 324
- SetReferencePose
 - gazebo::sensors::ImuSensor, 364
- SetRelativePose
 - gazebo::physics::Entity, 286
- SetRenderRate
 - gazebo::rendering::Camera, 179
- SetRenderTarget
 - gazebo::rendering::Camera, 180
 - gazebo::rendering::UserCamera, 820
- SetRequired
 - sdf::Element, 276
- SetRetro
 - gazebo::physics::RayShape, 638
- SetRibbonTrail
 - gazebo::rendering::Visual, 883
- SetRootNode
 - gazebo::common::Skeleton, 718
- SetRotation
 - gazebo::common::PoseKeyFrame, 597
 - gazebo::rendering::Visual, 883
- SetSID
 - gazebo::common::NodeTransform, 541
- SetSORPGSIters
 - gazebo::physics::PhysicsEngine, 567
- SetSORPGSPreconIters
 - gazebo::physics::PhysicsEngine, 567
- SetSORPGSW
 - gazebo::physics::PhysicsEngine, 568
- SetSaveFramePathname
 - gazebo::rendering::Camera, 180
- SetSaveable
 - gazebo::physics::Base, 145
- SetScale
 - gazebo::common::Mesh, 472
 - gazebo::common::SubMesh, 757
 - gazebo::math::Matrix4, 465
 - gazebo::physics::TrimeshShape, 808
 - gazebo::rendering::Visual, 883
- SetScene
 - gazebo::rendering::Camera, 180
 - gazebo::rendering::Visual, 883
- SetSceneNode
 - gazebo::rendering::Camera, 180
- SetSeed
 - gazebo::math::Rand, 625
 - gazebo::physics::PhysicsEngine, 567
- SetSelected
 - gazebo::physics::Base, 145
 - gazebo::physics::Link, 429
 - gazebo::rendering::Light, 410
- setSelectedEntity
 - gazebo::event::Events, 304
- SetSelfCollide
 - gazebo::physics::Link, 429
- SetShadeMode
 - gazebo::common::Material, 453
- SetShaderType
 - gazebo::rendering::Visual, 884
- SetShadowsEnabled
 - gazebo::rendering::Scene, 675
- SetShape
 - gazebo::physics::Collision, 200
- SetShininess
 - gazebo::common::Material, 453

- SetShowOnTop
 - gazebo::rendering::MovableText, 511
- SetSimTime
 - gazebo::physics::World, 902
- SetSize
 - gazebo::physics::BoxShape, 153
 - gazebo::physics::CylinderShape, 249
 - gazebo::physics::PlaneShape, 581
- SetSkeleton
 - gazebo::common::Mesh, 472
- SetSkeletonPose
 - gazebo::rendering::Visual, 884
- SetSourceValues
 - gazebo::common::NodeTransform, 541
- SetSpaceWidth
 - gazebo::rendering::MovableText, 511
- SetSpecular
 - gazebo::common::Material, 453
 - gazebo::rendering::Visual, 884
- SetSpecularColor
 - gazebo::rendering::Light, 410
- SetSpotFalloff
 - gazebo::rendering::Light, 410
- SetSpotInnerAngle
 - gazebo::rendering::Light, 410
- SetSpotOuterAngle
 - gazebo::rendering::Light, 410
- SetState
 - gazebo::physics::Collision, 200
 - gazebo::physics::Joint, 391
 - gazebo::physics::Link, 429
 - gazebo::physics::Model, 492
 - gazebo::physics::World, 903
- SetStatic
 - gazebo::physics::Entity, 286
- SetStepTime
 - gazebo::physics::PhysicsEngine, 568
- SetSubMeshCenter
 - gazebo::common::SubMesh, 757
- SetTension
 - gazebo::math::Spline, 742
- SetTexCoord
 - gazebo::common::SubMesh, 757
- SetTexCoordCount
 - gazebo::common::SubMesh, 758
- SetText
 - gazebo::rendering::MovableText, 511
- SetTextAlignment
 - gazebo::rendering::MovableText, 512
- SetTexture
 - gazebo::rendering::Projector, 599
- SetTextureImage
 - gazebo::common::Material, 454
- SetThreadPitch
 - gazebo::physics::ScrewJoint, 679
- SetTime
 - gazebo::common::Animation, 126
- SetTolIdentity
 - gazebo::math::Quaternion, 621
- SetToMax
 - gazebo::math::Vector3, 850
- SetToMin
 - gazebo::math::Vector3, 850
- SetToWallTime
 - gazebo::common::Time, 795
- SetTorque
 - gazebo::physics::Link, 429
- SetTransform
 - gazebo::common::SkeletonNode, 730
- SetTranslate
 - gazebo::math::Matrix4, 465
- SetTranslation
 - gazebo::common::PoseKeyFrame, 597
- SetTransparency
 - gazebo::common::Material, 454
 - gazebo::rendering::Visual, 884
- SetTransparent
 - gazebo::rendering::Scene, 675
- SetType
 - gazebo::common::NodeTransform, 541
 - gazebo::common::SkeletonNode, 731
- SetUpdateFunc
 - sdf::Param, 554
- SetUpdateRate
 - gazebo::physics::PhysicsEngine, 568
 - gazebo::sensors::Sensor, 691
- SetUserData
 - gazebo::rendering::Grid, 340
- SetValue
 - gazebo::common::NumericKeyFrame, 545
 - sdf::ParamT, 557
- SetVelocity
 - gazebo::physics::Joint, 391
- SetVertex
 - gazebo::common::SubMesh, 758
- SetVertexCount
 - gazebo::common::SubMesh, 758
- SetVerticalAngleMax
 - gazebo::sensors::GpuRaySensor, 334
- SetVerticalAngleMin
 - gazebo::sensors::GpuRaySensor, 334
- SetViewController
 - gazebo::rendering::UserCamera, 820
- SetViewportDimensions
 - gazebo::rendering::UserCamera, 820
- SetVisibilityFlags
 - gazebo::rendering::Visual, 884
- SetVisible

- gazebo::rendering::Scene, 675
- gazebo::rendering::Visual, 885
- gazebo::rendering::WireBox, 892
- SetWindowId
 - gazebo::rendering::Camera, 180
- SetWorld
 - gazebo::physics::Base, 145
 - gazebo::physics::WorldState, 910
- SetWorldCFM
 - gazebo::physics::PhysicsEngine, 568
- SetWorldERP
 - gazebo::physics::PhysicsEngine, 568
- SetWorldPose
 - gazebo::physics::Entity, 286
 - gazebo::rendering::Camera, 180
 - gazebo::rendering::UserCamera, 820
 - gazebo::rendering::Visual, 885
- SetWorldPosition
 - gazebo::rendering::Camera, 181
 - gazebo::rendering::Visual, 885
- SetWorldRotation
 - gazebo::rendering::Camera, 181
 - gazebo::rendering::Visual, 885
- SetWorldTwist
 - gazebo::physics::Entity, 287
- ShadeMode
 - gazebo::common::Material, 449
- shadeMode
 - gazebo::common::Material, 455
- ShadeModeStr
 - gazebo::common::Material, 455
- Shape
 - gazebo::physics::Shape, 707
- shape
 - gazebo::physics::Collision, 201
- Shape.hh, 1066
- ShapePtr
 - gazebo::physics, 96
- shift
 - gazebo::common::MouseEvent, 506
- shininess
 - gazebo::common::Material, 455
- Show
 - gazebo::rendering::GUIOverlay, 345
- ShowBoundingBox
 - gazebo::rendering::Visual, 885
- ShowCOM
 - gazebo::rendering::Visual, 885
- ShowCOMs
 - gazebo::rendering::Scene, 676
- ShowCollision
 - gazebo::rendering::Visual, 885
- ShowCollisions
 - gazebo::rendering::Scene, 676
- ShowContacts
 - gazebo::rendering::Scene, 676
- ShowJoints
 - gazebo::rendering::Scene, 676
 - gazebo::rendering::Visual, 886
- ShowRotation
 - gazebo::rendering::ArrowVisual, 129
 - gazebo::rendering::AxisVisual, 133
- ShowSkeleton
 - gazebo::rendering::Visual, 886
- ShowVisual
 - gazebo::rendering::Light, 411
- ShowWireframe
 - gazebo::rendering::Camera, 181
- Shutdown
 - gazebo::transport::Connection, 225
- sid
 - gazebo::common::NodeTransform, 541
- Signal
 - gazebo::event::EventT, 311–313
- simTime
 - gazebo::common::UpdateInfo, 811
 - gazebo::physics::State, 747
- SimTimeEventHandler
 - gazebo::sensors::SimTimeEventHandler, 709
- SingletonT
 - ~SingletonT, 712
 - Instance, 712
 - SingletonT, 712
 - SingletonT, 712
- SingletonT< T >, 710
- SingletonT.hh, 1068
- size
 - gazebo::math::Plane, 578
- skelAnimation
 - gazebo::physics::Actor, 115
- skelNodesMap
 - gazebo::physics::Actor, 115
- Skeleton
 - gazebo::common::Skeleton, 714
- skeleton
 - gazebo::physics::Actor, 115
- Skeleton.hh, 1068
- SkeletonAnimation
 - gazebo::common::SkeletonAnimation, 720
- SkeletonAnimation.hh, 1070
- SkeletonNode
 - gazebo::common::SkeletonNode, 725, 726
- SkeletonNodeType
 - gazebo::common::SkeletonNode, 725
- skinFile
 - gazebo::physics::Actor, 115
- skinScale
 - gazebo::physics::Actor, 115

- SkyX, 108
- skyx
 - gazebo::rendering::Scene, 677
- Sleep
 - gazebo::common::Time, 795
- Slerp
 - gazebo::math::Quaternion, 622
- SliderJoint
 - gazebo::physics::SliderJoint, 733
- SliderJoint.hh, 1071
- slip1
 - gazebo::physics::SurfaceParams, 767
- slip2
 - gazebo::physics::SurfaceParams, 768
- SnapVisualToNearestBelow
 - gazebo::rendering::Scene, 676
- source
 - gazebo::common::NodeTransform, 541
- specular
 - gazebo::common::Material, 455
- SphereShape
 - gazebo::physics::SphereShape, 738
- SphereShape.hh, 1073
- SphereShapePtr
 - gazebo::physics, 96
- Spline
 - gazebo::math::Spline, 740
- Spline.hh, 1074
- Squad
 - gazebo::math::Quaternion, 622
- Stamp
 - Messages, 62
- Start
 - gazebo::common::LogRecord, 444
 - gazebo::common::Timer, 798
 - gazebo::util::DiagnosticTimer, 261
- startDelay
 - gazebo::physics::Actor, 115
- StartRead
 - gazebo::transport::Connection, 225
- startTime
 - gazebo::physics::TrajectoryInfo, 805
- StartTimer
 - gazebo::util::DiagnosticManager, 260
- State
 - gazebo::physics::State, 745
- State.hh, 1075
- Step
 - gazebo::common::LogPlay, 439
- step
 - gazebo::event::Events, 304
- StepWorld
 - gazebo::physics::World, 903
- Stop
 - gazebo::common::LogRecord, 444
 - gazebo::common::Timer, 798
 - gazebo::Master, 446
 - gazebo::physics::Actor, 113
 - gazebo::physics::World, 903
 - gazebo::sensors::SensorManager, 698
 - gazebo::Server, 701
 - gazebo::transport::ConnectionManager, 229
 - gazebo::transport::IOManager, 375
 - gazebo::util::DiagnosticTimer, 262
- stop
 - gazebo, 83
 - gazebo::event::Events, 304
 - Sensors, 72
 - Transport, 78
- stop_world
 - Classes for physics and dynamics, 46
- stop_worlds
 - Classes for physics and dynamics, 46
- StopAnimation
 - gazebo::physics::Entity, 287
 - gazebo::physics::Model, 492
- StopRead
 - gazebo::transport::Connection, 225
- StopTimer
 - gazebo::util::DiagnosticManager, 260
- StrStr_M
 - gazebo::common, 86
- StripSceneName
 - gazebo::rendering::Scene, 676
- StripWorldName
 - gazebo::physics::World, 903
- SubMesh
 - gazebo::common::SubMesh, 751
- SubNodeMap
 - gazebo::transport::TopicManager, 800
- subSampling
 - gazebo::physics::HeightmapShape, 352
- Subscribe
 - gazebo::transport::ConnectionManager, 229
 - gazebo::transport::Node, 530–532
 - gazebo::transport::TopicManager, 804
- SubscribeOptions
 - gazebo::transport::SubscribeOptions, 759
- SubscribeOptions.hh, 1078
- Subscriber
 - gazebo::transport::Subscriber, 761
- Subscriber.hh, 1079
- SubscriberPtr
 - gazebo::transport, 104
- SubscriptionTransport
 - gazebo::transport::SubscriptionTransport, 763
- SubscriptionTransport.hh, 1081
- SubscriptionTransportPtr

- gazebo::transport, 104
- SurfaceParams
 - gazebo::physics::SurfaceParams, 765
- SurfaceParams.hh, 1082
- SurfaceParamsPtr
 - gazebo::physics, 96
- SystemPaths.hh, 1084
 - GetCurrentDir, 1085
 - LINUX, 1085
- SystemPlugin
 - gazebo::SystemPlugin, 774
- SystemPluginPtr
 - gazebo, 83
- systemPluginsArgc
 - gazebo::Server, 701
- systemPluginsArgv
 - gazebo::Server, 701

- TPtr
 - gazebo::PluginT, 583
- TRANSLATE
 - gazebo::common::NodeTransform, 538
- TRIANGLES
 - gazebo::common::SubMesh, 751
- TRIFANS
 - gazebo::common::SubMesh, 751
- TRIMESH_SHAPE
 - gazebo::physics::Base, 139
- TRISTRIPS
 - gazebo::common::SubMesh, 751
- tangents
 - gazebo::math::RotationSpline, 657
 - gazebo::math::Spline, 743
- tension
 - gazebo::math::Spline, 743
- texImage
 - gazebo::common::Material, 455
- textureHeight
 - gazebo::rendering::Camera, 184
- textureWidth
 - gazebo::rendering::Camera, 184
- threadPitch
 - gazebo::physics::ScrewJoint, 680
- Time
 - gazebo::common::Time, 779
- time
 - gazebo::common::KeyFrame, 403
 - gazebo::physics::Contact, 235
 - gazebo::sensors::SimTimeEvent, 708
- Time.hh, 1085
- timePos
 - gazebo::common::Animation, 127
- Timer
 - gazebo::common::Timer, 797
- Timer.hh, 1086
- ToStdString
 - sdf::Element, 276
 - sdf::SDF, 681
- Toggle
 - gazebo::rendering::Projector, 599
- ToggleShowVisual
 - gazebo::rendering::Light, 411
- ToggleShowWireframe
 - gazebo::rendering::Camera, 181
- ToggleVisible
 - gazebo::rendering::Visual, 886
- TopicManager.hh, 1087
- TrackVisual
 - gazebo::rendering::Camera, 181
- TrackVisualFromSDF
 - Messages, 63
- TrackVisualImpl
 - gazebo::rendering::Camera, 181, 182
 - gazebo::rendering::UserCamera, 821
- trajInfo
 - gazebo::physics::Actor, 115
- trajectories
 - gazebo::physics::Actor, 115
- transform
 - gazebo::common::NodeTransform, 542
 - gazebo::common::SkeletonNode, 732
- TransformAffine
 - gazebo::math::Matrix4, 465
- TransformType
 - gazebo::common::NodeTransform, 538
- Translate
 - gazebo::rendering::Camera, 182
- translate
 - gazebo::common::PoseKeyFrame, 597
- translated
 - gazebo::physics::TrajectoryInfo, 805
- transparency
 - gazebo::common::Material, 455
- Transport, 74
 - CallbackHelperPtr, 75
 - clear_buffers, 76
 - fini, 76
 - get_master_uri, 76
 - get_topic_namespaces, 76
 - getAdvertisedTopics, 76
 - getTopicMsgType, 77
 - init, 77
 - is_stopped, 77
 - pause_incoming, 77
 - request, 77
 - requestNoReply, 78
 - run, 78
 - stop, 78

- Transport.hh, 1089
- TransportTypes.hh, 1091
- TrimeshShape
 - gazebo::physics::TrimeshShape, 807
- TrimeshShape.hh, 1093
- type
 - gazebo::common::MouseEvent, 506
 - gazebo::common::NodeTransform, 542
 - gazebo::common::SkeletonNode, 732
 - gazebo::physics::TrajectoryInfo, 805
 - gazebo::PluginT, 584
- typeName
 - sdf::Param, 554
- typeString
 - gazebo::rendering::ViewController, 867
- UIntGen
 - gazebo::math, 89
- UNION
 - gazebo::common::MeshCSG, 473
- UNIVERSAL_JOINT
 - gazebo::physics::Base, 139
- UNKNOWN_PIXEL_FORMAT
 - gazebo::common::Image, 357
- URDF2Gazebo
 - urdf2gazebo::URDF2Gazebo, 812
- URealGen
 - gazebo::math, 89
- Unadvertise
 - gazebo::transport::ConnectionManager, 229
 - gazebo::transport::TopicManager, 804
- UniformIntDist
 - gazebo::math, 89
- UniformRealDist
 - gazebo::math, 89
- UniversalJoint
 - gazebo::physics::UniversalJoint, 810
- UniversalJoint.hh, 1094
- Unsubscribe
 - gazebo::transport::ConnectionManager, 229
 - gazebo::transport::Subscriber, 761
 - gazebo::transport::TopicManager, 804
- Update
 - gazebo::common::PID, 575
 - gazebo::physics::Actor, 113
 - gazebo::physics::Base, 145
 - gazebo::physics::Joint, 391
 - gazebo::physics::JointController, 395
 - gazebo::physics::Link, 429
 - gazebo::physics::Model, 492
 - gazebo::physics::MultiRayShape, 525
 - gazebo::physics::RayShape, 638
 - gazebo::physics::TrimeshShape, 808
 - gazebo::rendering::Camera, 182
 - gazebo::rendering::DynamicLines, 265
 - gazebo::rendering::FPSViewController, 319
 - gazebo::rendering::GUIOverlay, 345
 - gazebo::rendering::MovableText, 512
 - gazebo::rendering::OrbitViewController, 549
 - gazebo::rendering::UserCamera, 821
 - gazebo::rendering::ViewController, 867
 - gazebo::rendering::Visual, 886
 - gazebo::sensors::Sensor, 691
 - gazebo::sensors::SensorManager, 698
 - sdf::Element, 277
 - sdf::Param, 554
 - sdf::ParamT, 558
- UpdateChildrenTransforms
 - gazebo::common::SkeletonNode, 731
- UpdateCollision
 - gazebo::physics::PhysicsEngine, 569
- UpdateFromMsg
 - gazebo::rendering::Light, 411
 - gazebo::rendering::Visual, 886
- updateFunc
 - sdf::Param, 555
- UpdateImpl
 - gazebo::sensors::CameraSensor, 189
 - gazebo::sensors::ContactSensor, 241
 - gazebo::sensors::DepthCameraSensor, 257
 - gazebo::sensors::GpuRaySensor, 334
 - gazebo::sensors::ImuSensor, 364
 - gazebo::sensors::MultiCameraSensor, 518
 - gazebo::sensors::RaySensor, 633
 - gazebo::sensors::RFIDSensor, 646
 - gazebo::sensors::RFIDTag, 648
 - gazebo::sensors::Sensor, 691
- UpdateInfo.hh, 1095
- UpdateMass
 - gazebo::physics::Link, 429
- UpdateParameters
 - gazebo::physics::Actor, 113
 - gazebo::physics::Base, 146
 - gazebo::physics::Collision, 200
 - gazebo::physics::Entity, 287
 - gazebo::physics::Inertial, 372
 - gazebo::physics::Joint, 391
 - gazebo::physics::Link, 430
 - gazebo::physics::Model, 493
- UpdateParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 736
- updateParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 705
- UpdateParamsForCompositeMap
 - gazebo::rendering::GzTerrainMatGen::SM2Profile, 736

- updatePeriod
 - gazebo::sensors::Sensor, 692
- UpdatePhysics
 - gazebo::physics::PhysicsEngine, 569
- UpdatePoint
 - gazebo::math::RotationSpline, 657
 - gazebo::math::Spline, 742
- UpdatePublications
 - gazebo::transport::TopicManager, 805
- UpdateRays
 - gazebo::physics::MultiRayShape, 525
- UpdateShaders
 - gazebo::rendering::RTShaderSystem, 662
- UpdateStateSDF
 - gazebo::physics::World, 903
- UpdateSurface
 - gazebo::physics::Link, 430
- updateVpParams
 - gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 706
- upperLimit
 - gazebo::physics::Joint, 393
- urdf2gazebo, 108
- urdf2gazebo::GazeboExtension, 319
- urdf2gazebo::URDF2Gazebo, 811
 - ~URDF2Gazebo, 812
 - InitModelDoc, 812
 - InitModelFile, 812
 - InitModelString, 812
 - URDF2Gazebo, 812
- UrdfCollisionPtr
 - Gazebo_parser, 68
- UrdfLinkPtr
 - Gazebo_parser, 68
- UrdfVisualPtr
 - Gazebo_parser, 68
- useCFMDamping
 - gazebo::physics::Joint, 393
- UserCamera
 - gazebo::rendering::UserCamera, 815
- UserCamera.hh, 1096
- UserCameraPtr
 - gazebo::rendering, 99
- UtilTypes.hh, 1096
- Utility, 79
 - DIAG_TIMER_LAP, 79
 - DIAG_TIMER_START, 79
 - DIAG_TIMER_STOP, 79
- V_ABOVE
 - gazebo::rendering::MovableText, 508
- V_BELOW
 - gazebo::rendering::MovableText, 508
- VEL
 - gazebo::physics::Joint, 380
- VERTEX
 - gazebo::rendering::RenderEngine, 641
- VISUAL
 - gazebo::physics::Base, 139
- VISUAL_PLUGIN
 - Common, 32
- Valid
 - gazebo::common::Image, 361
- ValidateIP
 - gazebo::transport::Connection, 225
- value
 - gazebo::common::NumericKeyFrame, 545
 - sdf::ParamT, 558
- variance
 - Math, 52
- Vector2d
 - gazebo::math::Vector2d, 823
- Vector2d.hh, 1097
- Vector2i
 - gazebo::math::Vector2i, 831
- Vector2i.hh, 1098
- Vector3
 - gazebo::math::Vector3, 841
- Vector3.hh, 1099
- Vector4
 - gazebo::math::Vector4, 854
- Vector4.hh, 1100
- velocityLimit
 - gazebo::physics::Joint, 393
- version
 - sdf::SDF, 681
- VertAlign
 - gazebo::rendering::MovableText, 508
- vertElem
 - gazebo::physics::MultiRayShape, 526
 - gazebo::sensors::GpuRaySensor, 336
- vertHalfAngle
 - gazebo::sensors::GpuRaySensor, 336
- vertRangeCount
 - gazebo::sensors::GpuRaySensor, 336
- vertRayCount
 - gazebo::sensors::GpuRaySensor, 336
- vertSize
 - gazebo::physics::HeightmapShape, 352
- vertexBufferCapacity
 - gazebo::rendering::DynamicRenderable, 269
- vertexIndex
 - gazebo::common::NodeAssignment, 537
- vfov
 - gazebo::sensors::GpuRaySensor, 336
- Video
 - gazebo::common::Video, 862
- Video.hh, 1102

- VideoVisual
 - gazebo::rendering::VideoVisual, 864
- VideoVisual.hh, 1103
- ViewController
 - gazebo::rendering::ViewController, 865
- ViewController.hh, 1104
- viewport
 - gazebo::rendering::Camera, 184
- visPub
 - gazebo::physics::Entity, 288
- visitRenderables
 - gazebo::rendering::MovableText, 512
- Visual
 - gazebo::rendering::Visual, 872
- Visual.hh, 1105
- VisualFromSDF
 - Messages, 63
- visualMsg
 - gazebo::physics::Entity, 288
- visualName
 - gazebo::physics::Actor, 116
- VisualPlugin
 - gazebo::VisualPlugin, 888
- VisualPluginPtr
 - gazebo, 83
- VisualPtr
 - gazebo::rendering, 99
- visuals
 - gazebo::physics::Link, 430
- w
 - gazebo::math::Quaternion, 623
 - gazebo::math::Vector4, 860
- WORLD_PLUGIN
 - Common, 32
- WaitForConnection
 - gazebo::transport::Publisher, 609
- wallTime
 - gazebo::physics::State, 747
- weight
 - gazebo::common::NodeAssignment, 537
- White
 - gazebo::common::Color, 216
- windowId
 - gazebo::rendering::Camera, 184
- WindowManager.hh, 1106
- WireBox
 - gazebo::rendering::WireBox, 892
- WireBox.hh, 1106
- World
 - gazebo::physics::World, 895
- world
 - gazebo::physics::Base, 146
 - gazebo::physics::PhysicsEngine, 569
 - gazebo::sensors::Sensor, 693
- World.hh, 1107
- worldCreated
 - gazebo::event::Events, 305
- worldName
 - gazebo::common::UpdateInfo, 811
- WorldPlugin
 - gazebo::WorldPlugin, 904
- WorldPluginPtr
 - gazebo, 83
- WorldPtr
 - gazebo::physics, 96
- WorldState
 - gazebo::physics::WorldState, 907
- WorldState.hh, 1109
- worldUpdateBegin
 - gazebo::event::Events, 305
- worldUpdateEnd
 - gazebo::event::Events, 305
- worldUpdateStart
 - gazebo::event::Events, 305
- wrench
 - gazebo::physics::Contact, 235
- Write
 - sdf::SDF, 681
- x
 - gazebo::math::Quaternion, 623
 - gazebo::math::Vector2d, 829
 - gazebo::math::Vector2i, 838
 - gazebo::math::Vector3, 851
 - gazebo::math::Vector4, 861
- X_POSITION
 - BVHLoader.hh, 924
- X_ROTATION
 - BVHLoader.hh, 924
- y
 - gazebo::math::Quaternion, 623
 - gazebo::math::Vector2d, 829
 - gazebo::math::Vector2i, 838
 - gazebo::math::Vector3, 851
 - gazebo::math::Vector4, 861
- Y_POSITION
 - BVHLoader.hh, 924
- Y_ROTATION
 - BVHLoader.hh, 924
- Yellow
 - gazebo::common::Color, 216
- z
 - gazebo::math::Quaternion, 623
 - gazebo::math::Vector3, 851
 - gazebo::math::Vector4, 861
- Z_POSITION

- BVHLoader.hh, 924
- Z_ROTATION
 - BVHLoader.hh, 924
- ZERO
 - gazebo::math::Matrix4, 466
- Zero
 - gazebo::common::Time, 796
 - gazebo::math::Pose, 593
 - gazebo::math::Vector3, 851