

Gazebo  
1.4.0

Generated by Doxygen 1.8.2

Fri Feb 1 2013 11:32:40



# Contents

<b>1</b>	<b>Gazebo API Reference</b>	<b>1</b>
<b>2</b>	<b>Todo List</b>	<b>3</b>
<b>3</b>	<b>Module Index</b>	<b>5</b>
3.1	Modules . . . . .	5
<b>4</b>	<b>Namespace Index</b>	<b>7</b>
4.1	Namespace List . . . . .	7
<b>5</b>	<b>Hierarchical Index</b>	<b>9</b>
5.1	Class Hierarchy . . . . .	9
<b>6</b>	<b>Class Index</b>	<b>15</b>
6.1	Class List . . . . .	15
<b>7</b>	<b>File Index</b>	<b>23</b>
7.1	File List . . . . .	23
<b>8</b>	<b>Module Documentation</b>	<b>27</b>
8.1	Common . . . . .	27
8.1.1	Detailed Description . . . . .	31
8.1.2	Macro Definition Documentation . . . . .	31
8.1.2.1	DIAG_TIMER . . . . .	31
8.1.2.2	gzclr_end . . . . .	31
8.1.2.3	gzclr_start . . . . .	31
8.1.2.4	gzdbg . . . . .	31
8.1.2.5	gzerr . . . . .	31
8.1.2.6	gzlog . . . . .	31
8.1.2.7	gzmsg . . . . .	32
8.1.2.8	gzthrow . . . . .	32

8.1.2.9	gzwarn . . . . .	32
8.1.3	Typedef Documentation . . . . .	32
8.1.3.1	DiagnosticTimerPtr . . . . .	32
8.1.4	Enumeration Type Documentation . . . . .	32
8.1.4.1	PluginType . . . . .	32
8.1.5	Function Documentation . . . . .	32
8.1.5.1	NullStream . . . . .	32
8.1.5.2	add_search_path_suffix . . . . .	33
8.1.5.3	ColorErr . . . . .	33
8.1.5.4	ColorMsg . . . . .	33
8.1.5.5	DownloadDependencies . . . . .	33
8.1.5.6	find_file . . . . .	33
8.1.5.7	find_file_path . . . . .	34
8.1.5.8	GetDBConfig . . . . .	34
8.1.5.9	GetManifest . . . . .	34
8.1.5.10	GetModelConfig . . . . .	34
8.1.5.11	GetModelFile . . . . .	34
8.1.5.12	GetModelName . . . . .	35
8.1.5.13	GetModelPath . . . . .	35
8.1.5.14	GetModels . . . . .	35
8.1.5.15	GetModels . . . . .	35
8.1.5.16	GetURI . . . . .	36
8.1.5.17	HasModel . . . . .	36
8.1.5.18	Init . . . . .	36
8.1.5.19	Log . . . . .	36
8.1.5.20	SetQuiet . . . . .	36
8.1.6	Variable Documentation . . . . .	37
8.1.6.1	PixelFormatNames . . . . .	37
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	stop	72
8.9	Transport	73
8.9.1	Detailed Description	74

8.9.2	Typedef Documentation . . . . .	74
8.9.2.1	CallbackHelperPtr . . . . .	74
8.9.3	Function Documentation . . . . .	75
8.9.3.1	clear_buffers . . . . .	75
8.9.3.2	fini . . . . .	75
8.9.3.3	get_master_uri . . . . .	75
8.9.3.4	get_topic_namespaces . . . . .	75
8.9.3.5	getAdvertisedTopics . . . . .	75
8.9.3.6	getAdvertisedTopics . . . . .	75
8.9.3.7	getTopicMsgType . . . . .	76
8.9.3.8	init . . . . .	76
8.9.3.9	is_stopped . . . . .	76
8.9.3.10	pause_incoming . . . . .	76
8.9.3.11	request . . . . .	77
8.9.3.12	requestNoReply . . . . .	77
8.9.3.13	requestNoReply . . . . .	77
8.9.3.14	run . . . . .	77
8.9.3.15	stop . . . . .	77
<b>9</b>	<b>Namespace Documentation</b>	<b>79</b>
9.1	boost Namespace Reference . . . . .	79
9.2	gazebo Namespace Reference . . . . .	79
9.2.1	Detailed Description . . . . .	80
9.2.2	Typedef Documentation . . . . .	81
9.2.2.1	GUIPluginPtr . . . . .	81
9.2.2.2	ModelPluginPtr . . . . .	81
9.2.2.3	SensorPluginPtr . . . . .	81
9.2.2.4	SystemPluginPtr . . . . .	81
9.2.2.5	VisualPluginPtr . . . . .	81
9.2.2.6	WorldPluginPtr . . . . .	81
9.2.3	Function Documentation . . . . .	81
9.2.3.1	add_plugin . . . . .	81
9.2.3.2	find_file . . . . .	81
9.2.3.3	fini . . . . .	81
9.2.3.4	init . . . . .	81
9.2.3.5	load . . . . .	81
9.2.3.6	print_version . . . . .	81



9.2.3.7	run	81
9.2.3.8	stop	81
9.3	gazebo::common Namespace Reference	81
9.3.1	Detailed Description	84
9.3.2	Typedef Documentation	84
9.3.2.1	AnimationPtr	84
9.3.2.2	NodeMap	84
9.3.2.3	NodeMapIter	84
9.3.2.4	NumericAnimationPtr	84
9.3.2.5	Param_V	84
9.3.2.6	PoseAnimationPtr	84
9.3.2.7	RawNodeAnim	84
9.3.2.8	RawNodeWeights	84
9.3.2.9	RawSkeletonAnim	84
9.3.2.10	StrStr_M	84
9.4	gazebo::event Namespace Reference	84
9.4.1	Detailed Description	85
9.4.2	Typedef Documentation	85
9.4.2.1	Connection_V	85
9.4.2.2	ConnectionPtr	85
9.5	gazebo::math Namespace Reference	85
9.5.1	Detailed Description	87
9.5.2	Typedef Documentation	87
9.5.2.1	GeneratorType	87
9.5.2.2	NormalRealDist	87
9.5.2.3	NRealGen	87
9.5.2.4	UIntGen	87
9.5.2.5	UniformIntDist	87
9.5.2.6	UniformRealDist	87
9.5.2.7	URealGen	87
9.6	gazebo::msgs Namespace Reference	87
9.6.1	Detailed Description	89
9.6.2	Typedef Documentation	89
9.6.2.1	MsgFactoryFn	89
9.7	gazebo::physics Namespace Reference	89
9.7.1	Detailed Description	93
9.7.2	Typedef Documentation	93

9.7.2.1	Actor_V	93
9.7.2.2	ActorPtr	93
9.7.2.3	Base_V	93
9.7.2.4	BasePtr	93
9.7.2.5	BoxShapePtr	93
9.7.2.6	Collision_V	93
9.7.2.7	CollisionPtr	93
9.7.2.8	ContactPtr	93
9.7.2.9	CylinderShapePtr	93
9.7.2.10	EntityPtr	93
9.7.2.11	HeightmapShapePtr	93
9.7.2.12	InertialPtr	93
9.7.2.13	Joint_V	93
9.7.2.14	JointController_V	93
9.7.2.15	JointControllerPtr	93
9.7.2.16	JointPtr	93
9.7.2.17	Link_V	93
9.7.2.18	LinkPtr	93
9.7.2.19	MeshShapePtr	93
9.7.2.20	Model_V	94
9.7.2.21	ModelPtr	94
9.7.2.22	MultiRayShapePtr	94
9.7.2.23	PhysicsEnginePtr	94
9.7.2.24	RayShapePtr	94
9.7.2.25	RoadPtr	94
9.7.2.26	ShapePtr	94
9.7.2.27	SphereShapePtr	94
9.7.2.28	SurfaceParamsPtr	94
9.7.2.29	WorldPtr	94
9.8	gazebo::rendering Namespace Reference	94
9.8.1	Detailed Description	96
9.8.2	Typedef Documentation	97
9.8.2.1	ArrowVisualPtr	97
9.8.2.2	AxisVisualPtr	97
9.8.2.3	CameraPtr	97
9.8.2.4	CameraVisualPtr	97
9.8.2.5	COMVisualPtr	97

9.8.2.6	ContactVisualPtr . . . . .	97
9.8.2.7	DepthCameraPtr . . . . .	97
9.8.2.8	DynamicLinesPtr . . . . .	97
9.8.2.9	GpuLaserPtr . . . . .	97
9.8.2.10	JointVisualPtr . . . . .	97
9.8.2.11	LaserVisualPtr . . . . .	97
9.8.2.12	LightPtr . . . . .	97
9.8.2.13	RFIDTagVisualPtr . . . . .	97
9.8.2.14	RFIDVisualPtr . . . . .	97
9.8.2.15	ScenePtr . . . . .	97
9.8.2.16	UserCameraPtr . . . . .	97
9.8.2.17	VisualPtr . . . . .	97
9.8.3	Enumeration Type Documentation . . . . .	97
9.8.3.1	RenderOpType . . . . .	97
9.9	gazebo::sensors Namespace Reference . . . . .	98
9.9.1	Detailed Description . . . . .	99
9.9.2	Typedef Documentation . . . . .	99
9.9.2.1	CameraSensor_V . . . . .	99
9.9.2.2	CameraSensorPtr . . . . .	99
9.9.2.3	ContactSensor_V . . . . .	99
9.9.2.4	ContactSensorPtr . . . . .	99
9.9.2.5	DepthCameraSensor_V . . . . .	99
9.9.2.6	DepthCameraSensorPtr . . . . .	100
9.9.2.7	GpuRaySensor_V . . . . .	100
9.9.2.8	GpuRaySensorPtr . . . . .	100
9.9.2.9	RaySensor_V . . . . .	100
9.9.2.10	RaySensorPtr . . . . .	100
9.9.2.11	RFIDSensor_V . . . . .	100
9.9.2.12	RFIDSensorPtr . . . . .	100
9.9.2.13	RFIDTag_V . . . . .	100
9.9.2.14	RFIDTagPtr . . . . .	100
9.9.2.15	Sensor_V . . . . .	100
9.9.2.16	SensorFactoryFn . . . . .	100
9.9.2.17	SensorPtr . . . . .	100
9.10	gazebo::transport Namespace Reference . . . . .	100
9.10.1	Typedef Documentation . . . . .	102
9.10.1.1	ConnectionPtr . . . . .	102

---

9.10.1.2	NodePtr	102
9.10.1.3	PublicationPtr	102
9.10.1.4	PublicationTransportPtr	102
9.10.1.5	PublisherPtr	102
9.10.1.6	SubscriberPtr	102
9.10.1.7	SubscriptionTransportPtr	102
9.11	google Namespace Reference	102
9.12	google::protobuf Namespace Reference	102
9.13	google::protobuf::compiler Namespace Reference	102
9.14	google::protobuf::compiler::cpp Namespace Reference	102
9.15	Ogre Namespace Reference	103
9.16	ogre Namespace Reference	103
9.17	sdf Namespace Reference	103
9.17.1	Detailed Description	104
9.17.2	Typedef Documentation	104
9.17.2.1	ElementPtr	104
9.17.2.2	ElementPtr_V	104
9.17.2.3	Param_V	104
9.17.2.4	ParamPtr	104
9.17.2.5	SDFPtr	104
9.17.3	Function Documentation	104
9.17.3.1	addNestedModel	104
9.17.3.2	copyChildren	104
9.17.3.3	init	104
9.17.3.4	initDoc	104
9.17.3.5	initDoc	104
9.17.3.6	initFile	104
9.17.3.7	initFile	104
9.17.3.8	initString	104
9.17.3.9	initXml	104
9.17.3.10	readDoc	105
9.17.3.11	readDoc	105
9.17.3.12	readFile	105
9.17.3.13	readString	105
9.17.3.14	readString	105
9.17.3.15	readXml	105
9.18	SkyX Namespace Reference	105

9.19 urdf2gazebo Namespace Reference . . . . .	105
9.19.1 Detailed Description . . . . .	105
<b>10 Class Documentation</b>	<b>107</b>
10.1 A Class Reference . . . . .	107
10.1.1 Detailed Description . . . . .	107
10.2 gazebo::physics::Actor Class Reference . . . . .	107
10.2.1 Detailed Description . . . . .	110
10.2.2 Constructor & Destructor Documentation . . . . .	110
10.2.2.1 Actor . . . . .	110
10.2.2.2 ~Actor . . . . .	110
10.2.3 Member Function Documentation . . . . .	110
10.2.3.1 Fini . . . . .	110
10.2.3.2 GetSDF . . . . .	110
10.2.3.3 Init . . . . .	111
10.2.3.4 IsActive . . . . .	111
10.2.3.5 Load . . . . .	111
10.2.3.6 Play . . . . .	111
10.2.3.7 Stop . . . . .	111
10.2.3.8 Update . . . . .	111
10.2.3.9 UpdateParameters . . . . .	111
10.2.4 Member Data Documentation . . . . .	112
10.2.4.1 active . . . . .	112
10.2.4.2 autoStart . . . . .	112
10.2.4.3 bonePosePub . . . . .	112
10.2.4.4 interpolateX . . . . .	112
10.2.4.5 lastPos . . . . .	112
10.2.4.6 lastScriptTime . . . . .	112
10.2.4.7 lastTraj . . . . .	112
10.2.4.8 loop . . . . .	112
10.2.4.9 mainLink . . . . .	112
10.2.4.10 mesh . . . . .	112
10.2.4.11 oldAction . . . . .	112
10.2.4.12 pathLength . . . . .	113
10.2.4.13 playStartTime . . . . .	113
10.2.4.14 prevFrameTime . . . . .	113
10.2.4.15 scriptLength . . . . .	113

10.2.4.16	skelAnimation	113
10.2.4.17	skeleton	113
10.2.4.18	skelNodesMap	113
10.2.4.19	skinFile	113
10.2.4.20	skinScale	113
10.2.4.21	startDelay	113
10.2.4.22	trajectories	113
10.2.4.23	trajInfo	114
10.2.4.24	visualName	114
10.3	gazebo::math::Angle Class Reference	114
10.3.1	Detailed Description	115
10.3.2	Constructor & Destructor Documentation	115
10.3.2.1	Angle	115
10.3.2.2	Angle	115
10.3.2.3	Angle	116
10.3.2.4	~Angle	116
10.3.3	Member Function Documentation	116
10.3.3.1	Degree	116
10.3.3.2	Normalize	116
10.3.3.3	operator!=	116
10.3.3.4	operator*	116
10.3.3.5	operator*	116
10.3.3.6	operator*= 117	
10.3.3.7	operator+ 117	
10.3.3.8	operator+= 117	
10.3.3.9	operator- 117	
10.3.3.10	operator-= 118	
10.3.3.11	operator/ 118	
10.3.3.12	operator/= 118	
10.3.3.13	operator< 118	
10.3.3.14	operator<= 119	
10.3.3.15	operator== 119	
10.3.3.16	operator> 119	
10.3.3.17	operator>= 119	
10.3.3.18	Radian	120
10.3.3.19	SetFromDegree	120
10.3.3.20	SetFromRadian	120

10.3.4	Friends And Related Function Documentation . . . . .	120
10.3.4.1	operator<< . . . . .	120
10.3.4.2	operator>> . . . . .	121
10.4	gazebo::common::Animation Class Reference . . . . .	121
10.4.1	Detailed Description . . . . .	122
10.4.2	Member Typedef Documentation . . . . .	122
10.4.2.1	KeyFrame_V . . . . .	122
10.4.3	Constructor & Destructor Documentation . . . . .	123
10.4.3.1	Animation . . . . .	123
10.4.3.2	~Animation . . . . .	123
10.4.4	Member Function Documentation . . . . .	123
10.4.4.1	AddTime . . . . .	123
10.4.4.2	GetKeyFrame . . . . .	123
10.4.4.3	GetKeyFrameCount . . . . .	123
10.4.4.4	GetKeyFramesAtTime . . . . .	123
10.4.4.5	GetLength . . . . .	124
10.4.4.6	GetTime . . . . .	124
10.4.4.7	SetLength . . . . .	124
10.4.4.8	SetTime . . . . .	124
10.4.5	Member Data Documentation . . . . .	124
10.4.5.1	build . . . . .	124
10.4.5.2	keyFrames . . . . .	125
10.4.5.3	length . . . . .	125
10.4.5.4	loop . . . . .	125
10.4.5.5	name . . . . .	125
10.4.5.6	timePos . . . . .	125
10.5	gazebo::rendering::ArrowVisual Class Reference . . . . .	125
10.5.1	Detailed Description . . . . .	126
10.5.2	Constructor & Destructor Documentation . . . . .	126
10.5.2.1	ArrowVisual . . . . .	126
10.5.2.2	~ArrowVisual . . . . .	127
10.5.3	Member Function Documentation . . . . .	127
10.5.3.1	Load . . . . .	127
10.5.3.2	ShowRotation . . . . .	127
10.6	gazebo::common::AssertionInternalError Class Reference . . . . .	127
10.6.1	Detailed Description . . . . .	128
10.6.2	Constructor & Destructor Documentation . . . . .	128

10.6.2.1	AssertionInternalError . . . . .	128
10.6.2.2	~AssertionInternalError . . . . .	129
10.7	gazebo::rendering::AxisVisual Class Reference . . . . .	129
10.7.1	Detailed Description . . . . .	130
10.7.2	Constructor & Destructor Documentation . . . . .	130
10.7.2.1	AxisVisual . . . . .	130
10.7.2.2	~AxisVisual . . . . .	130
10.7.3	Member Function Documentation . . . . .	130
10.7.3.1	Load . . . . .	130
10.7.3.2	ScaleXAxis . . . . .	130
10.7.3.3	ScaleYAxis . . . . .	131
10.7.3.4	ScaleZAxis . . . . .	131
10.7.3.5	SetAxisMaterial . . . . .	131
10.7.3.6	ShowRotation . . . . .	131
10.8	gazebo::physics::BallJoint< T > Class Template Reference . . . . .	131
10.8.1	Detailed Description . . . . .	132
10.8.2	Constructor & Destructor Documentation . . . . .	132
10.8.2.1	BallJoint . . . . .	132
10.8.2.2	~BallJoint . . . . .	133
10.8.3	Member Function Documentation . . . . .	133
10.8.3.1	GetAngleCount . . . . .	133
10.8.3.2	GetHighStop . . . . .	133
10.8.3.3	GetLowStop . . . . .	133
10.8.3.4	Load . . . . .	133
10.8.3.5	SetAxis . . . . .	133
10.8.3.6	SetHighStop . . . . .	133
10.8.3.7	SetLowStop . . . . .	133
10.9	gazebo::physics::Base Class Reference . . . . .	133
10.9.1	Detailed Description . . . . .	136
10.9.2	Member Enumeration Documentation . . . . .	136
10.9.2.1	EntityType . . . . .	136
10.9.3	Constructor & Destructor Documentation . . . . .	137
10.9.3.1	Base . . . . .	137
10.9.3.2	~Base . . . . .	137
10.9.4	Member Function Documentation . . . . .	137
10.9.4.1	AddChild . . . . .	137
10.9.4.2	AddType . . . . .	138



---

10.9.4.3	Fini . . . . .	138
10.9.4.4	GetById . . . . .	138
10.9.4.5	GetByName . . . . .	138
10.9.4.6	GetChild . . . . .	138
10.9.4.7	GetChild . . . . .	139
10.9.4.8	GetChildCount . . . . .	139
10.9.4.9	GetId . . . . .	139
10.9.4.10	GetName . . . . .	139
10.9.4.11	GetParent . . . . .	139
10.9.4.12	GetParentId . . . . .	139
10.9.4.13	GetSaveable . . . . .	140
10.9.4.14	GetScopedName . . . . .	140
10.9.4.15	GetSDF . . . . .	140
10.9.4.16	GetType . . . . .	140
10.9.4.17	GetWorld . . . . .	140
10.9.4.18	HasType . . . . .	140
10.9.4.19	Init . . . . .	141
10.9.4.20	IsSelected . . . . .	141
10.9.4.21	Load . . . . .	141
10.9.4.22	operator== . . . . .	141
10.9.4.23	Print . . . . .	142
10.9.4.24	RemoveChild . . . . .	142
10.9.4.25	RemoveChild . . . . .	142
10.9.4.26	RemoveChildren . . . . .	142
10.9.4.27	Reset . . . . .	142
10.9.4.28	Reset . . . . .	142
10.9.4.29	SetName . . . . .	143
10.9.4.30	SetParent . . . . .	143
10.9.4.31	SetSaveable . . . . .	143
10.9.4.32	SetSelected . . . . .	143
10.9.4.33	SetWorld . . . . .	143
10.9.4.34	Update . . . . .	143
10.9.4.35	UpdateParameters . . . . .	144
10.9.5	Member Data Documentation . . . . .	144
10.9.5.1	children . . . . .	144
10.9.5.2	childrenEnd . . . . .	144
10.9.5.3	parent . . . . .	144

10.9.5.4	sdf	144
10.9.5.5	world	144
10.10	gazebo::math::Box Class Reference	144
10.10.1	Detailed Description	146
10.10.2	Constructor & Destructor Documentation	146
10.10.2.1	Box	146
10.10.2.2	Box	146
10.10.2.3	Box	146
10.10.2.4	~Box	146
10.10.3	Member Function Documentation	146
10.10.3.1	GetCenter	146
10.10.3.2	GetSize	146
10.10.3.3	GetXLength	147
10.10.3.4	GetYLength	147
10.10.3.5	GetZLength	147
10.10.3.6	Merge	147
10.10.3.7	operator+	147
10.10.3.8	operator+=	147
10.10.3.9	operator-	148
10.10.3.10	operator=	148
10.10.3.11	operator==	148
10.10.4	Friends And Related Function Documentation	148
10.10.4.1	operator<<	149
10.10.5	Member Data Documentation	149
10.10.5.1	max	149
10.10.5.2	min	149
10.11	gazebo::physics::BoxShape Class Reference	149
10.11.1	Detailed Description	150
10.11.2	Constructor & Destructor Documentation	151
10.11.2.1	BoxShape	151
10.11.2.2	~BoxShape	151
10.11.3	Member Function Documentation	151
10.11.3.1	FillMsg	151
10.11.3.2	GetSize	151
10.11.3.3	Init	151
10.11.3.4	ProcessMsg	151
10.11.3.5	SetSize	152

10.12	gazebo::common::BVHLoader Class Reference	152
10.12.1	Detailed Description	152
10.12.2	Constructor & Destructor Documentation	152
10.12.2.1	BVHLoader	152
10.12.2.2	~BVHLoader	152
10.12.3	Member Function Documentation	152
10.12.3.1	Load	152
10.13	gazebo::transport::CallbackHelper Class Reference	153
10.13.1	Detailed Description	154
10.13.2	Constructor & Destructor Documentation	154
10.13.2.1	CallbackHelper	154
10.13.2.2	~CallbackHelper	154
10.13.3	Member Function Documentation	154
10.13.3.1	GetId	154
10.13.3.2	GetLatching	154
10.13.3.3	GetMsgType	155
10.13.3.4	HandleData	155
10.13.3.5	IsLocal	155
10.13.4	Member Data Documentation	155
10.13.4.1	latching	155
10.14	gazebo::transport::CallbackHelperT< M > Class Template Reference	155
10.14.1	Detailed Description	156
10.14.2	Constructor & Destructor Documentation	156
10.14.2.1	CallbackHelperT	156
10.14.3	Member Function Documentation	157
10.14.3.1	GetMsgType	157
10.14.3.2	HandleData	157
10.14.3.3	IsLocal	157
10.15	gazebo::rendering::Camera Class Reference	157
10.15.1	Detailed Description	163
10.15.2	Constructor & Destructor Documentation	163
10.15.2.1	Camera	163
10.15.2.2	~Camera	164
10.15.3	Member Function Documentation	164
10.15.3.1	AnimationComplete	164
10.15.3.2	AttachToVisual	164
10.15.3.3	AttachToVisualImpl	164

---

10.15.3.4 AttachToVisualImpl . . . . .	165
10.15.3.5 ConnectNewImageFrame . . . . .	165
10.15.3.6 CreateRenderTexture . . . . .	165
10.15.3.7 DisconnectNewImageFrame . . . . .	165
10.15.3.8 EnableSaveFrame . . . . .	166
10.15.3.9 Fini . . . . .	166
10.15.3.10GetAspectRatio . . . . .	166
10.15.3.11GetAvgFPS . . . . .	166
10.15.3.12GetCameraToViewportRay . . . . .	166
10.15.3.13GetDirection . . . . .	167
10.15.3.14GetFarClip . . . . .	167
10.15.3.15GetFrameFilename . . . . .	167
10.15.3.16GetHFOV . . . . .	167
10.15.3.17GetImageByteSize . . . . .	167
10.15.3.18GetImageByteSize . . . . .	167
10.15.3.19GetImageData . . . . .	168
10.15.3.20GetImageDepth . . . . .	168
10.15.3.21GetImageFormat . . . . .	168
10.15.3.22GetImageHeight . . . . .	168
10.15.3.23GetImageWidth . . . . .	168
10.15.3.24GetInitialized . . . . .	169
10.15.3.25GetLastRenderWallTime . . . . .	169
10.15.3.26GetName . . . . .	169
10.15.3.27GetNearClip . . . . .	169
10.15.3.28GetOgreCamera . . . . .	169
10.15.3.29GetPitchNode . . . . .	169
10.15.3.30GetRenderRate . . . . .	170
10.15.3.31GetRenderTexture . . . . .	170
10.15.3.32GetRight . . . . .	170
10.15.3.33GetScene . . . . .	170
10.15.3.34GetSceneNode . . . . .	170
10.15.3.35GetTextureHeight . . . . .	170
10.15.3.36GetTextureWidth . . . . .	171
10.15.3.37GetTriangleCount . . . . .	171
10.15.3.38GetUp . . . . .	171
10.15.3.39GetVFOV . . . . .	171
10.15.3.40GetViewport . . . . .	171

10.15.3.41GetViewportHeight . . . . .	171
10.15.3.42GetViewportWidth . . . . .	172
10.15.3.43GetWindowId . . . . .	172
10.15.3.44GetWorldPointOnPlane . . . . .	172
10.15.3.45GetWorldPose . . . . .	172
10.15.3.46GetWorldPosition . . . . .	172
10.15.3.47GetWorldRotation . . . . .	173
10.15.3.48GetZValue . . . . .	173
10.15.3.49Init . . . . .	173
10.15.3.50IsAnimating . . . . .	173
10.15.3.51IsInitialized . . . . .	173
10.15.3.52IsVisible . . . . .	173
10.15.3.53IsVisible . . . . .	174
10.15.3.54Load . . . . .	174
10.15.3.55Load . . . . .	174
10.15.3.56MoveToPosition . . . . .	174
10.15.3.57MoveToPositions . . . . .	175
10.15.3.58PostRender . . . . .	175
10.15.3.59Render . . . . .	175
10.15.3.60RenderImpl . . . . .	175
10.15.3.61RotatePitch . . . . .	175
10.15.3.62RotateYaw . . . . .	175
10.15.3.63SaveFrame . . . . .	176
10.15.3.64SaveFrame . . . . .	176
10.15.3.65SetAspectRatio . . . . .	176
10.15.3.66SetCaptureData . . . . .	176
10.15.3.67SetClipDist . . . . .	176
10.15.3.68SetHFOV . . . . .	177
10.15.3.69SetImageHeight . . . . .	177
10.15.3.70SetImageSize . . . . .	177
10.15.3.71SetImageWidth . . . . .	177
10.15.3.72SetName . . . . .	177
10.15.3.73SetRenderRate . . . . .	178
10.15.3.74SetRenderTarget . . . . .	178
10.15.3.75SetSaveFramePathname . . . . .	178
10.15.3.76SetScene . . . . .	178
10.15.3.77SetSceneNode . . . . .	178

10.15.3.78	SetWindowId	178
10.15.3.79	SetWorldPose	178
10.15.3.80	SetWorldPosition	179
10.15.3.81	SetWorldRotation	179
10.15.3.82	ShowWireframe	179
10.15.3.83	ToggleShowWireframe	179
10.15.3.84	TrackVisual	179
10.15.3.85	TrackVisualImpl	179
10.15.3.86	TrackVisualImpl	180
10.15.3.87	Translate	180
10.15.3.88	Update	180
10.15.4	Member Data Documentation	180
10.15.4.1	animState	180
10.15.4.2	bayerFrameBuffer	180
10.15.4.3	camera	180
10.15.4.4	captureData	180
10.15.4.5	connections	181
10.15.4.6	imageFormat	181
10.15.4.7	imageHeight	181
10.15.4.8	imageWidth	181
10.15.4.9	initialized	181
10.15.4.10	lastRenderWallTime	181
10.15.4.11	name	181
10.15.4.12	newData	181
10.15.4.13	newImageFrame	181
10.15.4.14	onAnimationComplete	181
10.15.4.15	pitchNode	181
10.15.4.16	prevAnimTime	182
10.15.4.17	renderTarget	182
10.15.4.18	renderTexture	182
10.15.4.19	requests	182
10.15.4.20	saveCount	182
10.15.4.21	saveFrameBuffer	182
10.15.4.22	scene	182
10.15.4.23	sceneNode	182
10.15.4.24	sdf	182
10.15.4.25	textureHeight	182

---

10.15.4.26	textureWidth . . . . .	182
10.15.4.27	viewport . . . . .	182
10.15.4.28	windowId . . . . .	183
10.16	gazebo::sensors::CameraSensor Class Reference . . . . .	183
10.16.1	Detailed Description . . . . .	184
10.16.2	Constructor & Destructor Documentation . . . . .	184
10.16.2.1	CameraSensor . . . . .	184
10.16.2.2	~CameraSensor . . . . .	184
10.16.3	Member Function Documentation . . . . .	184
10.16.3.1	Fini . . . . .	184
10.16.3.2	GetCamera . . . . .	185
10.16.3.3	GetImageData . . . . .	185
10.16.3.4	GetImageHeight . . . . .	185
10.16.3.5	GetImageWidth . . . . .	185
10.16.3.6	GetTopic . . . . .	185
10.16.3.7	Init . . . . .	185
10.16.3.8	IsActive . . . . .	186
10.16.3.9	Load . . . . .	186
10.16.3.10	Load . . . . .	186
10.16.3.11	SaveFrame . . . . .	186
10.16.3.12	SetParent . . . . .	186
10.16.3.13	UpdateImpl . . . . .	187
10.17	gazebo::rendering::CameraVisual Class Reference . . . . .	187
10.17.1	Detailed Description . . . . .	188
10.17.2	Constructor & Destructor Documentation . . . . .	188
10.17.2.1	CameraVisual . . . . .	188
10.17.2.2	~CameraVisual . . . . .	188
10.17.3	Member Function Documentation . . . . .	188
10.17.3.1	Load . . . . .	188
10.18	gazebo::common::ColladaLoader Class Reference . . . . .	188
10.18.1	Detailed Description . . . . .	189
10.18.2	Constructor & Destructor Documentation . . . . .	189
10.18.2.1	ColladaLoader . . . . .	189
10.18.2.2	~ColladaLoader . . . . .	189
10.18.3	Member Function Documentation . . . . .	189
10.18.3.1	Load . . . . .	189
10.19	gazebo::physics::Collision Class Reference . . . . .	190

10.19.1 Detailed Description . . . . .	192
10.19.2 Constructor & Destructor Documentation . . . . .	192
10.19.2.1 Collision . . . . .	192
10.19.2.2 ~Collision . . . . .	193
10.19.3 Member Function Documentation . . . . .	193
10.19.3.1 AddContact . . . . .	193
10.19.3.2 ConnectContact . . . . .	193
10.19.3.3 DisconnectContact . . . . .	193
10.19.3.4 FillMsg . . . . .	193
10.19.3.5 Fini . . . . .	193
10.19.3.6 GetBoundingBox . . . . .	193
10.19.3.7 GetContactsEnabled . . . . .	194
10.19.3.8 GetLaserRetro . . . . .	194
10.19.3.9 GetLink . . . . .	194
10.19.3.10 GetModel . . . . .	194
10.19.3.11 GetRelativeAngularAccel . . . . .	194
10.19.3.12 GetRelativeAngularVel . . . . .	194
10.19.3.13 GetRelativeLinearAccel . . . . .	195
10.19.3.14 GetRelativeLinearVel . . . . .	195
10.19.3.15 GetShape . . . . .	195
10.19.3.16 GetShapeType . . . . .	195
10.19.3.17 GetState . . . . .	195
10.19.3.18 GetSurface . . . . .	196
10.19.3.19 GetWorldAngularAccel . . . . .	196
10.19.3.20 GetWorldAngularVel . . . . .	196
10.19.3.21 GetWorldLinearAccel . . . . .	196
10.19.3.22 GetWorldLinearVel . . . . .	196
10.19.3.23 Init . . . . .	196
10.19.3.24 IsPlaceable . . . . .	197
10.19.3.25 Load . . . . .	197
10.19.3.26 ProcessMsg . . . . .	197
10.19.3.27 SetCategoryBits . . . . .	197
10.19.3.28 SetCollideBits . . . . .	197
10.19.3.29 SetCollision . . . . .	197
10.19.3.30 SetContactsEnabled . . . . .	198
10.19.3.31 SetLaserRetro . . . . .	198
10.19.3.32 SetShape . . . . .	198



10.19.3.33	SetState	198
10.19.3.34	UpdateParameters	198
10.19.4	Member Data Documentation	198
10.19.4.1	link	199
10.19.4.2	placeable	199
10.19.4.3	shape	199
10.20	gazebo::physics::CollisionState Class Reference	199
10.20.1	Detailed Description	200
10.20.2	Constructor & Destructor Documentation	200
10.20.2.1	CollisionState	200
10.20.2.2	CollisionState	200
10.20.2.3	CollisionState	200
10.20.2.4	~CollisionState	201
10.20.3	Member Function Documentation	201
10.20.3.1	FillSDF	201
10.20.3.2	GetPose	201
10.20.3.3	IsZero	201
10.20.3.4	Load	201
10.20.3.5	operator+	202
10.20.3.6	operator-	202
10.20.3.7	operator=	202
10.20.4	Friends And Related Function Documentation	202
10.20.4.1	operator<<	202
10.21	gazebo::common::Color Class Reference	203
10.21.1	Detailed Description	205
10.21.2	Member Typedef Documentation	205
10.21.2.1	ABGR	205
10.21.2.2	ARGB	205
10.21.2.3	BGRA	205
10.21.2.4	RGBA	205
10.21.3	Constructor & Destructor Documentation	205
10.21.3.1	Color	206
10.21.3.2	Color	206
10.21.3.3	Color	206
10.21.3.4	~Color	206
10.21.4	Member Function Documentation	206
10.21.4.1	GetAsABGR	206

10.21.4.2	GetAsARGB	206
10.21.4.3	GetAsBGRA	206
10.21.4.4	GetAsHSV	207
10.21.4.5	GetAsRGBA	207
10.21.4.6	GetAsYUV	207
10.21.4.7	operator!=	207
10.21.4.8	operator*	207
10.21.4.9	operator*	208
10.21.4.10	operator*= operator+	208
10.21.4.11	operator+	208
10.21.4.12	operator+	208
10.21.4.13	operator+=	209
10.21.4.14	operator-	209
10.21.4.15	operator-	209
10.21.4.16	operator-=	209
10.21.4.17	operator/	210
10.21.4.18	operator/	210
10.21.4.19	operator/=	210
10.21.4.20	operator=	210
10.21.4.21	operator==	211
10.21.4.22	operator[]	211
10.21.4.23	Reset	211
10.21.4.24	Set	211
10.21.4.25	SetFromABGR	211
10.21.4.26	SetFromARGB	212
10.21.4.27	SetFromBGRA	212
10.21.4.28	SetFromHSV	212
10.21.4.29	SetFromRGBA	212
10.21.4.30	SetFromYUV	212
10.21.5	Friends And Related Function Documentation	213
10.21.5.1	operator<<	213
10.21.5.2	operator>>	213
10.21.6	Member Data Documentation	213
10.21.6.1	a	213
10.21.6.2	b	213
10.21.6.3	Black	213
10.21.6.4	Blue	213

---

10.21.6.5 g . . . . .	213
10.21.6.6 Green . . . . .	213
10.21.6.7 Purple . . . . .	213
10.21.6.8 r . . . . .	214
10.21.6.9 Red . . . . .	214
10.21.6.10White . . . . .	214
10.21.6.11Yellow . . . . .	214
10.22gazebo::rendering::COMVisual Class Reference . . . . .	214
10.22.1 Detailed Description . . . . .	215
10.22.2 Constructor & Destructor Documentation . . . . .	215
10.22.2.1 COMVisual . . . . .	215
10.22.2.2 ~COMVisual . . . . .	215
10.22.3 Member Function Documentation . . . . .	215
10.22.3.1 Load . . . . .	215
10.22.3.2 Load . . . . .	215
10.23gazebo::event::Connection Class Reference . . . . .	216
10.23.1 Detailed Description . . . . .	216
10.23.2 Constructor & Destructor Documentation . . . . .	216
10.23.2.1 Connection . . . . .	216
10.23.2.2 Connection . . . . .	216
10.23.2.3 ~Connection . . . . .	216
10.23.3 Member Function Documentation . . . . .	217
10.23.3.1 GetId . . . . .	217
10.24gazebo::transport::Connection Class Reference . . . . .	217
10.24.1 Detailed Description . . . . .	219
10.24.2 Member Typedef Documentation . . . . .	219
10.24.2.1 AcceptCallback . . . . .	219
10.24.2.2 ReadCallback . . . . .	219
10.24.3 Constructor & Destructor Documentation . . . . .	219
10.24.3.1 Connection . . . . .	219
10.24.3.2 ~Connection . . . . .	219
10.24.4 Member Function Documentation . . . . .	219
10.24.4.1 AsyncRead . . . . .	219
10.24.4.2 Cancel . . . . .	219
10.24.4.3 Connect . . . . .	220
10.24.4.4 ConnectToShutdown . . . . .	220
10.24.4.5 DisconnectShutdown . . . . .	220

10.24.4.6 EnqueueMsg . . . . .	220
10.24.4.7 GetId . . . . .	221
10.24.4.8 GetLocalAddress . . . . .	221
10.24.4.9 GetLocalHostname . . . . .	221
10.24.4.10GetLocalPort . . . . .	221
10.24.4.11GetLocalURI . . . . .	221
10.24.4.12GetRemoteAddress . . . . .	221
10.24.4.13GetRemoteHostname . . . . .	222
10.24.4.14GetRemotePort . . . . .	222
10.24.4.15GetRemoteURI . . . . .	222
10.24.4.16IsOpen . . . . .	222
10.24.4.17Listen . . . . .	222
10.24.4.18ProcessWriteQueue . . . . .	222
10.24.4.19Read . . . . .	223
10.24.4.20Shutdown . . . . .	223
10.24.4.21StartRead . . . . .	223
10.24.4.22StopRead . . . . .	223
10.24.4.23ValidateIP . . . . .	223
10.25gazebo::transport::ConnectionManager Class Reference . . . . .	223
10.25.1 Detailed Description . . . . .	225
10.25.2 Member Function Documentation . . . . .	225
10.25.2.1 Advertise . . . . .	225
10.25.2.2 ConnectToRemoteHost . . . . .	225
10.25.2.3 Fini . . . . .	225
10.25.2.4 GetAllPublishers . . . . .	225
10.25.2.5 GetTopicNamespaces . . . . .	226
10.25.2.6 Init . . . . .	226
10.25.2.7 IsRunning . . . . .	226
10.25.2.8 RegisterTopicNamespace . . . . .	226
10.25.2.9 RemoveConnection . . . . .	226
10.25.2.10Run . . . . .	227
10.25.2.11RunUpdate . . . . .	227
10.25.2.12Stop . . . . .	227
10.25.2.13Subscribe . . . . .	227
10.25.2.14Unadvertise . . . . .	227
10.25.2.15Unsubscribe . . . . .	227
10.25.2.16Unsubscribe . . . . .	227

10.25.3 Member Data Documentation . . . . .	228
10.25.3.1 eventConnections . . . . .	228
10.26gazebo::common::Console Class Reference . . . . .	228
10.26.1 Detailed Description . . . . .	228
10.27gazebo::physics::Contact Class Reference . . . . .	229
10.27.1 Detailed Description . . . . .	230
10.27.2 Constructor & Destructor Documentation . . . . .	230
10.27.2.1 Contact . . . . .	230
10.27.2.2 Contact . . . . .	230
10.27.2.3 ~Contact . . . . .	230
10.27.3 Member Function Documentation . . . . .	230
10.27.3.1 DebugString . . . . .	230
10.27.3.2 FillMsg . . . . .	230
10.27.3.3 operator= . . . . .	230
10.27.3.4 operator= . . . . .	231
10.27.3.5 Reset . . . . .	231
10.27.4 Member Data Documentation . . . . .	231
10.27.4.1 collision1 . . . . .	231
10.27.4.2 collision2 . . . . .	231
10.27.4.3 count . . . . .	231
10.27.4.4 depths . . . . .	231
10.27.4.5 normals . . . . .	231
10.27.4.6 positions . . . . .	232
10.27.4.7 time . . . . .	232
10.27.4.8 wrench . . . . .	232
10.28gazebo::physics::ContactManager Class Reference . . . . .	232
10.28.1 Detailed Description . . . . .	233
10.28.2 Constructor & Destructor Documentation . . . . .	233
10.28.2.1 ContactManager . . . . .	233
10.28.2.2 ~ContactManager . . . . .	233
10.28.3 Member Function Documentation . . . . .	233
10.28.3.1 Clear . . . . .	233
10.28.3.2 GetContact . . . . .	233
10.28.3.3 GetContactCount . . . . .	233
10.28.3.4 GetContacts . . . . .	233
10.28.3.5 Init . . . . .	234
10.28.3.6 NewContact . . . . .	234

10.28.3.7 PublishContacts . . . . .	234
10.28.3.8 ResetCount . . . . .	234
10.29gazebo::sensors::ContactSensor Class Reference . . . . .	234
10.29.1 Detailed Description . . . . .	236
10.29.2 Constructor & Destructor Documentation . . . . .	236
10.29.2.1 ContactSensor . . . . .	236
10.29.2.2 ~ContactSensor . . . . .	236
10.29.3 Member Function Documentation . . . . .	236
10.29.3.1 Fini . . . . .	236
10.29.3.2 GetCollisionContactCount . . . . .	236
10.29.3.3 GetCollisionCount . . . . .	236
10.29.3.4 GetCollisionName . . . . .	237
10.29.3.5 GetContacts . . . . .	237
10.29.3.6 GetContacts . . . . .	237
10.29.3.7 Init . . . . .	237
10.29.3.8 IsActive . . . . .	237
10.29.3.9 Load . . . . .	238
10.29.3.10Load . . . . .	238
10.29.3.11UpdateImpl . . . . .	238
10.30gazebo::rendering::ContactVisual Class Reference . . . . .	238
10.30.1 Detailed Description . . . . .	239
10.30.2 Constructor & Destructor Documentation . . . . .	239
10.30.2.1 ContactVisual . . . . .	239
10.30.2.2 ~ContactVisual . . . . .	240
10.30.3 Member Function Documentation . . . . .	240
10.30.3.1 SetEnabled . . . . .	240
10.31gazebo::rendering::Conversions Class Reference . . . . .	240
10.31.1 Detailed Description . . . . .	240
10.31.2 Member Function Documentation . . . . .	241
10.31.2.1 Convert . . . . .	241
10.31.2.2 Convert . . . . .	241
10.31.2.3 Convert . . . . .	241
10.31.2.4 Convert . . . . .	241
10.31.2.5 Convert . . . . .	242
10.31.2.6 Convert . . . . .	242
10.32sdf::Converter Class Reference . . . . .	242
10.32.1 Detailed Description . . . . .	242

---

10.32.2 Member Function Documentation . . . . .	242
10.32.2.1 Convert . . . . .	242
10.33gazebo::physics::CylinderShape Class Reference . . . . .	243
10.33.1 Detailed Description . . . . .	244
10.33.2 Constructor & Destructor Documentation . . . . .	244
10.33.2.1 CylinderShape . . . . .	244
10.33.2.2 ~CylinderShape . . . . .	244
10.33.3 Member Function Documentation . . . . .	244
10.33.3.1 FillMsg . . . . .	244
10.33.3.2 GetLength . . . . .	244
10.33.3.3 GetRadius . . . . .	245
10.33.3.4 Init . . . . .	245
10.33.3.5 ProcessMsg . . . . .	245
10.33.3.6 SetLength . . . . .	245
10.33.3.7 SetRadius . . . . .	245
10.33.3.8 SetSize . . . . .	245
10.34gazebo::rendering::DepthCamera Class Reference . . . . .	246
10.34.1 Detailed Description . . . . .	247
10.34.2 Constructor & Destructor Documentation . . . . .	247
10.34.2.1 DepthCamera . . . . .	247
10.34.2.2 ~DepthCamera . . . . .	247
10.34.3 Member Function Documentation . . . . .	248
10.34.3.1 ConnectNewDepthFrame . . . . .	248
10.34.3.2 ConnectNewRGBPointCloud . . . . .	248
10.34.3.3 CreateDepthTexture . . . . .	248
10.34.3.4 DisconnectNewDepthFrame . . . . .	248
10.34.3.5 DisconnectNewRGBPointCloud . . . . .	249
10.34.3.6 Fini . . . . .	249
10.34.3.7 GetDepthData . . . . .	249
10.34.3.8 Init . . . . .	249
10.34.3.9 Load . . . . .	249
10.34.3.10Load . . . . .	249
10.34.3.11PostRender . . . . .	249
10.34.3.12SetDepthTarget . . . . .	250
10.34.4 Member Data Documentation . . . . .	250
10.34.4.1 depthTarget . . . . .	250
10.34.4.2 depthTexture . . . . .	250

---

10.34.4.3 depthViewport . . . . .	250
10.35gazebo::sensors::DepthCameraSensor Class Reference . . . . .	250
10.35.1 Constructor & Destructor Documentation . . . . .	252
10.35.1.1 DepthCameraSensor . . . . .	252
10.35.1.2 ~DepthCameraSensor . . . . .	252
10.35.2 Member Function Documentation . . . . .	252
10.35.2.1 Fini . . . . .	252
10.35.2.2 GetDepthCamera . . . . .	252
10.35.2.3 Init . . . . .	252
10.35.2.4 Load . . . . .	252
10.35.2.5 Load . . . . .	253
10.35.2.6 SaveFrame . . . . .	253
10.35.2.7 SetActive . . . . .	253
10.35.2.8 SetParent . . . . .	253
10.35.2.9 UpdateImpl . . . . .	253
10.36gazebo::common::DiagnosticManager Class Reference . . . . .	254
10.36.1 Detailed Description . . . . .	255
10.36.2 Member Function Documentation . . . . .	255
10.36.2.1 CreateTimer . . . . .	255
10.36.2.2 GetEnabled . . . . .	255
10.36.2.3 GetLabel . . . . .	255
10.36.2.4 GetTime . . . . .	255
10.36.2.5 GetTime . . . . .	256
10.36.2.6 GetTimerCount . . . . .	256
10.36.2.7 SetEnabled . . . . .	256
10.36.2.8 TimerStart . . . . .	256
10.36.2.9 TimerStop . . . . .	256
10.37gazebo::common::DiagnosticTimer Class Reference . . . . .	257
10.37.1 Detailed Description . . . . .	257
10.37.2 Constructor & Destructor Documentation . . . . .	257
10.37.2.1 DiagnosticTimer . . . . .	257
10.37.2.2 ~DiagnosticTimer . . . . .	258
10.37.3 Member Function Documentation . . . . .	258
10.37.3.1 GetName . . . . .	258
10.38gazebo::rendering::DynamicLines Class Reference . . . . .	258
10.38.1 Detailed Description . . . . .	260
10.38.2 Constructor & Destructor Documentation . . . . .	260



10.38.2.1 DynamicLines . . . . .	260
10.38.2.2 ~DynamicLines . . . . .	260
10.38.3 Member Function Documentation . . . . .	260
10.38.3.1 AddPoint . . . . .	260
10.38.3.2 AddPoint . . . . .	260
10.38.3.3 Clear . . . . .	261
10.38.3.4 CreateVertexDeclaration . . . . .	261
10.38.3.5 FillHardwareBuffers . . . . .	261
10.38.3.6 GetMovableType . . . . .	261
10.38.3.7 getMovableType . . . . .	261
10.38.3.8 GetPoint . . . . .	261
10.38.3.9 GetPointCount . . . . .	262
10.38.3.10SetPoint . . . . .	262
10.38.3.11Update . . . . .	262
10.39gazebo::rendering::DynamicRenderable Class Reference . . . . .	262
10.39.1 Detailed Description . . . . .	264
10.39.2 Constructor & Destructor Documentation . . . . .	264
10.39.2.1 DynamicRenderable . . . . .	264
10.39.2.2 ~DynamicRenderable . . . . .	264
10.39.3 Member Function Documentation . . . . .	264
10.39.3.1 CreateVertexDeclaration . . . . .	264
10.39.3.2 FillHardwareBuffers . . . . .	264
10.39.3.3 getBoundingRadius . . . . .	265
10.39.3.4 GetOperationType . . . . .	265
10.39.3.5 getSquaredViewDepth . . . . .	265
10.39.3.6 Init . . . . .	265
10.39.3.7 PrepareHardwareBuffers . . . . .	265
10.39.3.8 SetOperationType . . . . .	266
10.39.4 Member Data Documentation . . . . .	266
10.39.4.1 indexBufferCapacity . . . . .	266
10.39.4.2 vertexBufferCapacity . . . . .	266
10.40sdf::Element Class Reference . . . . .	266
10.40.1 Detailed Description . . . . .	269
10.40.2 Constructor & Destructor Documentation . . . . .	269
10.40.2.1 Element . . . . .	269
10.40.2.2 ~Element . . . . .	269
10.40.3 Member Function Documentation . . . . .	269

10.40.3.1 AddAttribute . . . . .	269
10.40.3.2 AddElement . . . . .	269
10.40.3.3 AddElementDescription . . . . .	269
10.40.3.4 AddValue . . . . .	269
10.40.3.5 ClearElements . . . . .	269
10.40.3.6 Clone . . . . .	270
10.40.3.7 Copy . . . . .	270
10.40.3.8 GetAttribute . . . . .	270
10.40.3.9 GetAttribute . . . . .	270
10.40.3.10GetAttributeCount . . . . .	270
10.40.3.11GetAttributeSet . . . . .	270
10.40.3.12GetCopyChildren . . . . .	270
10.40.3.13GetDescription . . . . .	270
10.40.3.14GetElement . . . . .	270
10.40.3.15GetElement . . . . .	270
10.40.3.16GetElementDescription . . . . .	270
10.40.3.17GetElementDescription . . . . .	270
10.40.3.18GetElementDescriptionCount . . . . .	271
10.40.3.19GetElementImpl . . . . .	271
10.40.3.20GetFirstElement . . . . .	271
10.40.3.21GetInclude . . . . .	271
10.40.3.22GetName . . . . .	271
10.40.3.23GetNextElement . . . . .	271
10.40.3.24GetParent . . . . .	271
10.40.3.25GetRequired . . . . .	271
10.40.3.26GetValue . . . . .	271
10.40.3.27GetValueBool . . . . .	271
10.40.3.28GetValueChar . . . . .	271
10.40.3.29GetValueColor . . . . .	271
10.40.3.30GetValueDouble . . . . .	271
10.40.3.31GetValueFloat . . . . .	271
10.40.3.32GetValueInt . . . . .	271
10.40.3.33GetValuePose . . . . .	271
10.40.3.34GetValueQuaternion . . . . .	271
10.40.3.35GetValueString . . . . .	271
10.40.3.36GetValueTime . . . . .	271
10.40.3.37GetValueUInt . . . . .	271

10.40.3.38GetValueVector2d . . . . .	271
10.40.3.39GetValueVector3 . . . . .	271
10.40.3.40HasAttribute . . . . .	272
10.40.3.41HasElement . . . . .	272
10.40.3.42HasElementDescription . . . . .	272
10.40.3.43InsertElement . . . . .	272
10.40.3.44PrintDescription . . . . .	272
10.40.3.45PrintDocLeftPane . . . . .	272
10.40.3.46PrintDocRightPane . . . . .	272
10.40.3.47PrintValues . . . . .	272
10.40.3.48PrintWiki . . . . .	272
10.40.3.49RemoveChild . . . . .	272
10.40.3.50RemoveFromParent . . . . .	273
10.40.3.51Reset . . . . .	273
10.40.3.52Set . . . . .	273
10.40.3.53Set . . . . .	273
10.40.3.54Set . . . . .	273
10.40.3.55Set . . . . .	273
10.40.3.56Set . . . . .	273
10.40.3.57Set . . . . .	273
10.40.3.58Set . . . . .	273
10.40.3.59Set . . . . .	273
10.40.3.60Set . . . . .	273
10.40.3.61Set . . . . .	273
10.40.3.62Set . . . . .	273
10.40.3.63Set . . . . .	273
10.40.3.64Set . . . . .	273
10.40.3.65Set . . . . .	273
10.40.3.66Set . . . . .	273
10.40.3.67SetCopyChildren . . . . .	273
10.40.3.68SetDescription . . . . .	273
10.40.3.69SetInclude . . . . .	273
10.40.3.70SetName . . . . .	273
10.40.3.71SetParent . . . . .	273
10.40.3.72SetRequired . . . . .	273
10.40.3.73ToString . . . . .	274
10.40.3.74Update . . . . .	274

10.41 gazebo::physics::Entity Class Reference . . . . .	274
10.41.1 Detailed Description . . . . .	277
10.41.2 Constructor & Destructor Documentation . . . . .	277
10.41.2.1 Entity . . . . .	277
10.41.2.2 ~Entity . . . . .	277
10.41.3 Member Function Documentation . . . . .	277
10.41.3.1 Fini . . . . .	277
10.41.3.2 GetBoundingBox . . . . .	277
10.41.3.3 GetChildCollision . . . . .	277
10.41.3.4 GetChildLink . . . . .	278
10.41.3.5 GetCollisionBoundingBox . . . . .	278
10.41.3.6 GetDirtyPose . . . . .	278
10.41.3.7 GetInitialRelativePose . . . . .	278
10.41.3.8 GetNearestEntityBelow . . . . .	279
10.41.3.9 GetParentModel . . . . .	279
10.41.3.10 GetRelativeAngularAccel . . . . .	279
10.41.3.11 GetRelativeAngularVel . . . . .	279
10.41.3.12 GetRelativeLinearAccel . . . . .	279
10.41.3.13 GetRelativeLinearVel . . . . .	280
10.41.3.14 GetRelativePose . . . . .	280
10.41.3.15 GetWorldAngularAccel . . . . .	280
10.41.3.16 GetWorldAngularVel . . . . .	280
10.41.3.17 GetWorldLinearAccel . . . . .	280
10.41.3.18 GetWorldLinearVel . . . . .	281
10.41.3.19 GetWorldPose . . . . .	281
10.41.3.20 IsCanonicalLink . . . . .	281
10.41.3.21 IsStatic . . . . .	281
10.41.3.22 Load . . . . .	281
10.41.3.23 OnPoseChange . . . . .	282
10.41.3.24 PlaceOnEntity . . . . .	282
10.41.3.25 PlaceOnNearestEntityBelow . . . . .	282
10.41.3.26 Reset . . . . .	282
10.41.3.27 SetAnimation . . . . .	282
10.41.3.28 SetAnimation . . . . .	282
10.41.3.29 SetCanonicalLink . . . . .	282
10.41.3.30 SetInitialRelativePose . . . . .	283
10.41.3.31 SetName . . . . .	283

10.41.3.32	SetRelativePose	283
10.41.3.33	SetStatic	283
10.41.3.34	SetWorldPose	283
10.41.3.35	SetWorldTwist	284
10.41.3.36	StopAnimation	284
10.41.3.37	UpdateParameters	284
10.41.4	Member Data Documentation	284
10.41.4.1	animation	284
10.41.4.2	animationConnection	284
10.41.4.3	animationStartPose	284
10.41.4.4	connections	285
10.41.4.5	dirtyPose	285
10.41.4.6	node	285
10.41.4.7	parentEntity	285
10.41.4.8	poseMsg	285
10.41.4.9	prevAnimationTime	285
10.41.4.10	requestPub	285
10.41.4.11	visPub	285
10.41.4.12	visualMsg	285
10.42	gazebo::event::Event Class Reference	285
10.42.1	Detailed Description	287
10.42.2	Constructor & Destructor Documentation	287
10.42.2.1	~Event	287
10.42.3	Member Function Documentation	287
10.42.3.1	Disconnect	287
10.42.3.2	Disconnect	287
10.43	gazebo::rendering::Events Class Reference	288
10.43.1	Detailed Description	288
10.43.2	Member Function Documentation	288
10.43.2.1	ConnectCreateScene	288
10.43.2.2	ConnectRemoveScene	289
10.43.2.3	DisconnectCreateScene	289
10.43.2.4	DisconnectRemoveScene	289
10.43.3	Member Data Documentation	289
10.43.3.1	createScene	289
10.43.3.2	removeScene	289
10.44	gazebo::event::Events Class Reference	290

10.44.1 Detailed Description . . . . .	292
10.44.2 Member Function Documentation . . . . .	292
10.44.2.1 ConnectAddEntity . . . . .	292
10.44.2.2 ConnectCreateEntity . . . . .	293
10.44.2.3 ConnectDeleteEntity . . . . .	293
10.44.2.4 ConnectDiagTimerStart . . . . .	293
10.44.2.5 ConnectDiagTimerStop . . . . .	293
10.44.2.6 ConnectPause . . . . .	294
10.44.2.7 ConnectPostRender . . . . .	294
10.44.2.8 ConnectPreRender . . . . .	294
10.44.2.9 ConnectRender . . . . .	295
10.44.2.10 ConnectSetSelectedEntity . . . . .	295
10.44.2.11 ConnectStep . . . . .	295
10.44.2.12 ConnectStop . . . . .	295
10.44.2.13 ConnectWorldCreated . . . . .	296
10.44.2.14 ConnectWorldUpdateEnd . . . . .	296
10.44.2.15 ConnectWorldUpdateStart . . . . .	296
10.44.2.16 DisconnectAddEntity . . . . .	297
10.44.2.17 DisconnectCreateEntity . . . . .	297
10.44.2.18 DisconnectDeleteEntity . . . . .	297
10.44.2.19 DisconnectDiagTimerStart . . . . .	297
10.44.2.20 DisconnectDiagTimerStop . . . . .	297
10.44.2.21 DisconnectPause . . . . .	298
10.44.2.22 DisconnectPostRender . . . . .	298
10.44.2.23 DisconnectPreRender . . . . .	298
10.44.2.24 DisconnectRender . . . . .	298
10.44.2.25 DisconnectSetSelectedEntity . . . . .	298
10.44.2.26 DisconnectStep . . . . .	299
10.44.2.27 DisconnectStop . . . . .	299
10.44.2.28 DisconnectWorldCreated . . . . .	299
10.44.2.29 DisconnectWorldUpdateEnd . . . . .	299
10.44.2.30 DisconnectWorldUpdateStart . . . . .	299
10.44.3 Member Data Documentation . . . . .	300
10.44.3.1 addEntity . . . . .	300
10.44.3.2 deleteEntity . . . . .	300
10.44.3.3 diagTimerStart . . . . .	300
10.44.3.4 diagTimerStop . . . . .	300

10.44.3.5 entityCreated . . . . .	300
10.44.3.6 pause . . . . .	300
10.44.3.7 postRender . . . . .	300
10.44.3.8 preRender . . . . .	301
10.44.3.9 render . . . . .	301
10.44.3.10setSelectedEntity . . . . .	301
10.44.3.11step . . . . .	301
10.44.3.12stop . . . . .	301
10.44.3.13worldCreated . . . . .	301
10.44.3.14worldUpdateEnd . . . . .	301
10.44.3.15worldUpdateStart . . . . .	301
10.45gazebo::event::EventT< T > Class Template Reference . . . . .	302
10.45.1 Detailed Description . . . . .	304
10.45.2 Member Function Documentation . . . . .	304
10.45.2.1 operator() . . . . .	304
10.45.2.2 operator() . . . . .	304
10.45.2.3 operator() . . . . .	304
10.45.2.4 operator() . . . . .	304
10.45.2.5 operator() . . . . .	305
10.45.2.6 operator() . . . . .	305
10.45.2.7 operator() . . . . .	305
10.45.2.8 operator() . . . . .	306
10.45.2.9 operator() . . . . .	306
10.45.2.10operator() . . . . .	306
10.45.2.11operator() . . . . .	307
10.45.2.12Signal . . . . .	307
10.45.2.13Signal . . . . .	307
10.45.2.14Signal . . . . .	307
10.45.2.15Signal . . . . .	307
10.45.2.16Signal . . . . .	308
10.45.2.17Signal . . . . .	308
10.45.2.18Signal . . . . .	308
10.45.2.19Signal . . . . .	309
10.45.2.20Signal . . . . .	309
10.45.2.21Signal . . . . .	309
10.45.2.22Signal . . . . .	310
10.46gazebo::common::Exception Class Reference . . . . .	310

10.46.1 Detailed Description . . . . .	311
10.46.2 Constructor & Destructor Documentation . . . . .	311
10.46.2.1 Exception . . . . .	311
10.46.2.2 Exception . . . . .	311
10.46.2.3 ~Exception . . . . .	311
10.46.3 Member Function Documentation . . . . .	311
10.46.3.1 GetErrorFile . . . . .	312
10.46.3.2 GetErrorStr . . . . .	312
10.46.3.3 Print . . . . .	312
10.46.4 Friends And Related Function Documentation . . . . .	312
10.46.4.1 operator<< . . . . .	312
10.47gazebo::rendering::FPSViewController Class Reference . . . . .	312
10.47.1 Detailed Description . . . . .	313
10.47.2 Constructor & Destructor Documentation . . . . .	314
10.47.2.1 FPSViewController . . . . .	314
10.47.2.2 ~FPSViewController . . . . .	314
10.47.3 Member Function Documentation . . . . .	314
10.47.3.1 GetTypeString . . . . .	314
10.47.3.2 HandleKeyPressEvent . . . . .	314
10.47.3.3 HandleKeyReleaseEvent . . . . .	314
10.47.3.4 HandleMouseEvent . . . . .	314
10.47.3.5 Init . . . . .	315
10.47.3.6 Update . . . . .	315
10.48urdf2gazebo::GazeboExtension Class Reference . . . . .	315
10.49google::protobuf::compiler::cpp::GazeboGenerator Class Reference . . . . .	315
10.49.1 Detailed Description . . . . .	316
10.49.2 Constructor & Destructor Documentation . . . . .	316
10.49.2.1 GazeboGenerator . . . . .	316
10.49.2.2 ~GazeboGenerator . . . . .	316
10.49.3 Member Function Documentation . . . . .	316
10.49.3.1 Generate . . . . .	316
10.50gazebo::rendering::GpuLaser Class Reference . . . . .	316
10.50.1 Detailed Description . . . . .	318
10.50.2 Constructor & Destructor Documentation . . . . .	318
10.50.2.1 GpuLaser . . . . .	318
10.50.2.2 ~GpuLaser . . . . .	318
10.50.3 Member Function Documentation . . . . .	318



10.50.3.1 ConnectNewLaserFrame . . . . .	318
10.50.3.2 CreateLaserTexture . . . . .	318
10.50.3.3 DisconnectNewLaserFrame . . . . .	319
10.50.3.4 Fini . . . . .	319
10.50.3.5 GetLaserData . . . . .	319
10.50.3.6 Init . . . . .	319
10.50.3.7 Load . . . . .	319
10.50.3.8 Load . . . . .	319
10.50.3.9 notifyRenderSingleObject . . . . .	319
10.50.3.10PostRender . . . . .	319
10.50.3.11SetParentSensor . . . . .	320
10.50.3.12SetRangeCount . . . . .	320
10.51 gazebo::sensors::GpuRaySensor Class Reference . . . . .	320
10.51.1 Constructor & Destructor Documentation . . . . .	323
10.51.1.1 GpuRaySensor . . . . .	323
10.51.1.2 ~GpuRaySensor . . . . .	323
10.51.2 Member Function Documentation . . . . .	324
10.51.2.1 ConnectNewLaserFrame . . . . .	324
10.51.2.2 DisconnectNewLaserFrame . . . . .	324
10.51.2.3 Fini . . . . .	324
10.51.2.4 GetAngleMax . . . . .	324
10.51.2.5 GetAngleMin . . . . .	324
10.51.2.6 GetAngleResolution . . . . .	324
10.51.2.7 GetCameraCount . . . . .	324
10.51.2.8 GetCosHorzFOV . . . . .	325
10.51.2.9 GetCosVertFOV . . . . .	325
10.51.2.10GetFiducial . . . . .	325
10.51.2.11GetHorzFOV . . . . .	325
10.51.2.12GetHorzHalfAngle . . . . .	325
10.51.2.13GetLaserCamera . . . . .	326
10.51.2.14GetRange . . . . .	326
10.51.2.15GetRangeCount . . . . .	326
10.51.2.16GetRangeCountRatio . . . . .	326
10.51.2.17GetRangeMax . . . . .	326
10.51.2.18GetRangeMin . . . . .	327
10.51.2.19GetRangeResolution . . . . .	327
10.51.2.20GetRanges . . . . .	327

10.51.2.21	GetRayCount	327
10.51.2.22	GetRayCountRatio	327
10.51.2.23	GetRetro	327
10.51.2.24	GetVertFOV	328
10.51.2.25	GetVertHalfAngle	328
10.51.2.26	GetVerticalAngleMax	328
10.51.2.27	GetVerticalAngleMin	328
10.51.2.28	GetVerticalRangeCount	328
10.51.2.29	GetVerticalRayCount	329
10.51.2.30	Init	329
10.51.2.31	IsHorizontal	329
10.51.2.32	Load	329
10.51.2.33	Load	329
10.51.2.34	SetAngleMax	329
10.51.2.35	SetAngleMin	330
10.51.2.36	SetVerticalAngleMax	330
10.51.2.37	SetVerticalAngleMin	330
10.51.2.38	UpdateImpl	330
10.51.3	Member Data Documentation	330
10.51.3.1	cameraCount	330
10.51.3.2	cameraElem	330
10.51.3.3	chfov	330
10.51.3.4	cvfov	331
10.51.3.5	far	331
10.51.3.6	hfov	331
10.51.3.7	horzElem	331
10.51.3.8	horzHalfAngle	331
10.51.3.9	horzRangeCount	331
10.51.3.10	horzRayCount	331
10.51.3.11	isHorizontal	331
10.51.3.12	near	331
10.51.3.13	rangeCountRatio	331
10.51.3.14	rangeElem	331
10.51.3.15	rayCountRatio	332
10.51.3.16	scanElem	332
10.51.3.17	vertElem	332
10.51.3.18	vertHalfAngle	332

---

10.51.3.19	vertRangeCount . . . . .	332
10.51.3.20	vertRayCount . . . . .	332
10.51.3.21	vfov . . . . .	332
10.52	gazebo::rendering::Grid Class Reference . . . . .	332
10.52.1	Detailed Description . . . . .	333
10.52.2	Constructor & Destructor Documentation . . . . .	333
10.52.2.1	Grid . . . . .	333
10.52.2.2	~Grid . . . . .	334
10.52.3	Member Function Documentation . . . . .	334
10.52.3.1	Enable . . . . .	334
10.52.3.2	GetCellCount . . . . .	334
10.52.3.3	GetCellLength . . . . .	334
10.52.3.4	GetColor . . . . .	334
10.52.3.5	GetHeight . . . . .	334
10.52.3.6	GetLineWidth . . . . .	334
10.52.3.7	GetSceneNode . . . . .	335
10.52.3.8	Init . . . . .	335
10.52.3.9	SetCellCount . . . . .	335
10.52.3.10	SetCellLength . . . . .	335
10.52.3.11	SetColor . . . . .	335
10.52.3.12	SetHeight . . . . .	335
10.52.3.13	SetLineWidth . . . . .	335
10.52.3.14	SetUserData . . . . .	336
10.53	gazebo::physics::Gripper Class Reference . . . . .	336
10.53.1	Detailed Description . . . . .	336
10.53.2	Constructor & Destructor Documentation . . . . .	336
10.53.2.1	Gripper . . . . .	336
10.53.2.2	~Gripper . . . . .	337
10.53.3	Member Function Documentation . . . . .	337
10.53.3.1	Init . . . . .	337
10.53.3.2	Load . . . . .	337
10.54	gazebo::rendering::GUIOverlay Class Reference . . . . .	337
10.54.1	Detailed Description . . . . .	338
10.54.2	Constructor & Destructor Documentation . . . . .	338
10.54.2.1	GUIOverlay . . . . .	338
10.54.2.2	~GUIOverlay . . . . .	338
10.54.3	Member Function Documentation . . . . .	338

---

10.54.3.1 AttachCameraToImage . . . . .	338
10.54.3.2 AttachCameraToImage . . . . .	339
10.54.3.3 ButtonCallback . . . . .	339
10.54.3.4 CreateWindow . . . . .	339
10.54.3.5 HandleKeyPressEvent . . . . .	339
10.54.3.6 HandleKeyReleaseEvent . . . . .	340
10.54.3.7 HandleMouseEvent . . . . .	340
10.54.3.8 Hide . . . . .	340
10.54.3.9 Init . . . . .	340
10.54.3.10 IsInitialized . . . . .	340
10.54.3.11 LoadLayout . . . . .	341
10.54.3.12 Resize . . . . .	341
10.54.3.13 Show . . . . .	341
10.54.3.14 Update . . . . .	341
10.55 gazebo::rendering::GzTerrainMatGen Class Reference . . . . .	341
10.55.1 Constructor & Destructor Documentation . . . . .	342
10.55.1.1 GzTerrainMatGen . . . . .	342
10.55.1.2 ~GzTerrainMatGen . . . . .	342
10.56 gazebo::rendering::Heightmap Class Reference . . . . .	342
10.56.1 Detailed Description . . . . .	342
10.56.2 Constructor & Destructor Documentation . . . . .	343
10.56.2.1 Heightmap . . . . .	343
10.56.2.2 ~Heightmap . . . . .	343
10.56.3 Member Function Documentation . . . . .	343
10.56.3.1 GetHeight . . . . .	343
10.56.3.2 GetOgreTerrain . . . . .	343
10.56.3.3 Load . . . . .	343
10.56.3.4 LoadFromMsg . . . . .	343
10.57 gazebo::physics::HeightmapShape Class Reference . . . . .	344
10.57.1 Detailed Description . . . . .	345
10.57.2 Constructor & Destructor Documentation . . . . .	345
10.57.2.1 HeightmapShape . . . . .	345
10.57.2.2 ~HeightmapShape . . . . .	346
10.57.3 Member Function Documentation . . . . .	346
10.57.3.1 FillMsg . . . . .	346
10.57.3.2 GetHeight . . . . .	346
10.57.3.3 GetMaxHeight . . . . .	346

10.57.3.4	GetMinHeight	346
10.57.3.5	GetPos	346
10.57.3.6	GetSize	347
10.57.3.7	GetSubSampling	347
10.57.3.8	GetURI	347
10.57.3.9	GetVertexCount	347
10.57.3.10	Init	347
10.57.3.11	Load	347
10.57.3.12	ProcessMsg	348
10.57.4	Member Data Documentation	348
10.57.4.1	heights	348
10.57.4.2	img	348
10.57.4.3	scale	348
10.57.4.4	subSampling	348
10.57.4.5	vertSize	348
10.58	gazebo::physics::Hinge2Joint< T > Class Template Reference	348
10.58.1	Detailed Description	349
10.58.2	Constructor & Destructor Documentation	349
10.58.2.1	Hinge2Joint	349
10.58.2.2	~Hinge2Joint	350
10.58.3	Member Function Documentation	350
10.58.3.1	GetAngleCount	350
10.58.3.2	Load	350
10.59	gazebo::physics::HingeJoint< T > Class Template Reference	350
10.59.1	Detailed Description	351
10.59.2	Constructor & Destructor Documentation	351
10.59.2.1	HingeJoint	351
10.59.2.2	~HingeJoint	351
10.59.3	Member Function Documentation	351
10.59.3.1	GetAngleCount	351
10.59.3.2	Init	351
10.59.3.3	Load	352
10.60	gazebo::common::Image Class Reference	352
10.60.1	Detailed Description	353
10.60.2	Member Enumeration Documentation	353
10.60.2.1	PixelFormat	353
10.60.3	Constructor & Destructor Documentation	354

10.60.3.1 Image	354
10.60.3.2 ~Image	354
10.60.4 Member Function Documentation	354
10.60.4.1 ConvertPixelFormat	354
10.60.4.2 GetAvgColor	354
10.60.4.3 GetBPP	355
10.60.4.4 GetData	355
10.60.4.5 GetFilename	355
10.60.4.6 GetHeight	355
10.60.4.7 GetMaxColor	355
10.60.4.8 GetPitch	355
10.60.4.9 GetPixel	356
10.60.4.10GetPixelFormat	356
10.60.4.11GetRGBData	356
10.60.4.12GetWidth	356
10.60.4.13Load	356
10.60.4.14Rescale	356
10.60.4.15SavePNG	357
10.60.4.16SetFromData	357
10.60.4.17Valid	357
10.61 gazebo::sensors::ImuSensor Class Reference	357
10.61.1 Detailed Description	359
10.61.2 Constructor & Destructor Documentation	359
10.61.2.1 ImuSensor	359
10.61.2.2 ~ImuSensor	359
10.61.3 Member Function Documentation	359
10.61.3.1 Fini	359
10.61.3.2 GetAngularVelocity	359
10.61.3.3 GetLinearAcceleration	359
10.61.3.4 Init	359
10.61.3.5 Load	360
10.61.3.6 Load	360
10.61.3.7 UpdateImpl	360
10.62 gazebo::physics::Inertial Class Reference	360
10.62.1 Detailed Description	362
10.62.2 Constructor & Destructor Documentation	362
10.62.2.1 Inertial	362

10.62.2.2 Inertial	362
10.62.2.3 Inertial	362
10.62.2.4 ~Inertial	363
10.62.3 Member Function Documentation	363
10.62.3.1 GetCoG	363
10.62.3.2 GetlXX	363
10.62.3.3 GetlXY	363
10.62.3.4 GetlXZ	363
10.62.3.5 GetlYY	363
10.62.3.6 GetlYZ	364
10.62.3.7 GetlZZ	364
10.62.3.8 GetMass	364
10.62.3.9 GetPose	364
10.62.3.10 GetPrincipalMoments	364
10.62.3.11 GetProductsofInertia	364
10.62.3.12 Load	364
10.62.3.13 operator+	365
10.62.3.14 operator+=	365
10.62.3.15 operator=	365
10.62.3.16 ProcessMsg	365
10.62.3.17 Reset	366
10.62.3.18 Rotate	366
10.62.3.19 SetCoG	366
10.62.3.20 SetCoG	366
10.62.3.21 SetInertiaMatrix	366
10.62.3.22 SetlXX	366
10.62.3.23 SetlXY	367
10.62.3.24 SetlXZ	367
10.62.3.25 SetlYY	367
10.62.3.26 SetlYZ	367
10.62.3.27 SetlZZ	367
10.62.3.28 SetMass	367
10.62.3.29 UpdateParameters	368
10.62.4 Friends And Related Function Documentation	368
10.62.4.1 operator<<	368
10.63 gazebo::common::InternalError Class Reference	368
10.63.1 Detailed Description	369

10.63.2 Constructor & Destructor Documentation . . . . .	369
10.63.2.1 InternalError . . . . .	369
10.63.2.2 InternalError . . . . .	369
10.63.2.3 ~InternalError . . . . .	369
10.64 gazebo::transport::IOManager Class Reference . . . . .	369
10.64.1 Detailed Description . . . . .	370
10.64.2 Constructor & Destructor Documentation . . . . .	370
10.64.2.1 IOManager . . . . .	370
10.64.2.2 ~IOManager . . . . .	370
10.64.3 Member Function Documentation . . . . .	370
10.64.3.1 DecCount . . . . .	370
10.64.3.2 GetCount . . . . .	370
10.64.3.3 GetIO . . . . .	370
10.64.3.4 IncCount . . . . .	371
10.64.3.5 Stop . . . . .	371
10.65 gazebo::physics::Joint Class Reference . . . . .	371
10.65.1 Detailed Description . . . . .	374
10.65.2 Member Enumeration Documentation . . . . .	374
10.65.2.1 Attribute . . . . .	374
10.65.3 Constructor & Destructor Documentation . . . . .	375
10.65.3.1 Joint . . . . .	375
10.65.3.2 ~Joint . . . . .	375
10.65.4 Member Function Documentation . . . . .	375
10.65.4.1 ApplyDamping . . . . .	375
10.65.4.2 AreConnected . . . . .	375
10.65.4.3 Attach . . . . .	375
10.65.4.4 ConnectJointUpdate . . . . .	375
10.65.4.5 Detach . . . . .	376
10.65.4.6 DisconnectJointUpdate . . . . .	376
10.65.4.7 FillMsg . . . . .	376
10.65.4.8 GetAnchor . . . . .	376
10.65.4.9 GetAngle . . . . .	376
10.65.4.10 GetAngleCount . . . . .	377
10.65.4.11 GetAngleImpl . . . . .	377
10.65.4.12 GetChild . . . . .	377
10.65.4.13 GetForce . . . . .	377
10.65.4.14 GetForceTorque . . . . .	378



---

10.65.4.15	GetGlobalAxis	378
10.65.4.16	GetHighStop	378
10.65.4.17	GetJointLink	378
10.65.4.18	GetLinkForce	379
10.65.4.19	GetLinkTorque	379
10.65.4.20	GetLocalAxis	379
10.65.4.21	GetLowStop	379
10.65.4.22	GetMaxForce	380
10.65.4.23	GetParent	380
10.65.4.24	GetVelocity	380
10.65.4.25	init	380
10.65.4.26	Load	381
10.65.4.27	Load	381
10.65.4.28	Load	381
10.65.4.29	Reset	381
10.65.4.30	SetAnchor	381
10.65.4.31	SetAngle	382
10.65.4.32	SetAttribute	382
10.65.4.33	SetAxis	382
10.65.4.34	SetDamping	382
10.65.4.35	SetForce	382
10.65.4.36	SetHighStop	383
10.65.4.37	SetLowStop	383
10.65.4.38	SetMaxForce	383
10.65.4.39	SetModel	383
10.65.4.40	SetState	383
10.65.4.41	SetVelocity	384
10.65.4.42	Update	384
10.65.4.43	UpdateParameters	384
10.65.5	Member Data Documentation	384
10.65.5.1	anchorLink	384
10.65.5.2	anchorPos	384
10.65.5.3	applyDamping	384
10.65.5.4	childLink	384
10.65.5.5	dampingCoefficient	385
10.65.5.6	forceApplied	385
10.65.5.7	model	385

10.65.5.8 parentLink . . . . .	385
10.66gazebo::physics::JointController Class Reference . . . . .	385
10.66.1 Detailed Description . . . . .	385
10.66.2 Constructor & Destructor Documentation . . . . .	386
10.66.2.1 JointController . . . . .	386
10.66.3 Member Function Documentation . . . . .	386
10.66.3.1 AddJoint . . . . .	386
10.66.3.2 Reset . . . . .	386
10.66.3.3 SetJointPosition . . . . .	386
10.66.3.4 SetJointPosition . . . . .	386
10.66.3.5 SetJointPositions . . . . .	386
10.66.3.6 Update . . . . .	387
10.67gazebo::physics::JointState Class Reference . . . . .	387
10.67.1 Detailed Description . . . . .	388
10.67.2 Constructor & Destructor Documentation . . . . .	388
10.67.2.1 JointState . . . . .	388
10.67.2.2 JointState . . . . .	388
10.67.2.3 JointState . . . . .	388
10.67.2.4 ~JointState . . . . .	389
10.67.3 Member Function Documentation . . . . .	389
10.67.3.1 FillSDF . . . . .	389
10.67.3.2 GetAngle . . . . .	389
10.67.3.3 GetAngleCount . . . . .	389
10.67.3.4 GetAngles . . . . .	389
10.67.3.5 IsZero . . . . .	390
10.67.3.6 Load . . . . .	390
10.67.3.7 operator+ . . . . .	390
10.67.3.8 operator- . . . . .	390
10.67.3.9 operator= . . . . .	390
10.67.4 Friends And Related Function Documentation . . . . .	391
10.67.4.1 operator<< . . . . .	391
10.68gazebo::rendering::JointVisual Class Reference . . . . .	391
10.68.1 Detailed Description . . . . .	392
10.68.2 Constructor & Destructor Documentation . . . . .	392
10.68.2.1 JointVisual . . . . .	392
10.68.2.2 ~JointVisual . . . . .	393
10.68.3 Member Function Documentation . . . . .	393

10.68.3.1 Load . . . . .	393
10.69gazebo::physics::JointWrench Class Reference . . . . .	393
10.69.1 Detailed Description . . . . .	393
10.69.2 Member Function Documentation . . . . .	394
10.69.2.1 operator= . . . . .	394
10.69.3 Member Data Documentation . . . . .	394
10.69.3.1 body1Force . . . . .	394
10.69.3.2 body1Torque . . . . .	394
10.69.3.3 body2Force . . . . .	394
10.69.3.4 body2Torque . . . . .	394
10.70gazebo::common::KeyFrame Class Reference . . . . .	394
10.70.1 Detailed Description . . . . .	395
10.70.2 Constructor & Destructor Documentation . . . . .	395
10.70.2.1 KeyFrame . . . . .	395
10.70.2.2 ~KeyFrame . . . . .	395
10.70.3 Member Function Documentation . . . . .	396
10.70.3.1 GetTime . . . . .	396
10.70.4 Member Data Documentation . . . . .	396
10.70.4.1 time . . . . .	396
10.71gazebo::rendering::LaserVisual Class Reference . . . . .	396
10.71.1 Detailed Description . . . . .	397
10.71.2 Constructor & Destructor Documentation . . . . .	397
10.71.2.1 LaserVisual . . . . .	397
10.71.2.2 ~LaserVisual . . . . .	397
10.71.3 Member Function Documentation . . . . .	397
10.71.3.1 SetEmissive . . . . .	397
10.72gazebo::rendering::Light Class Reference . . . . .	397
10.72.1 Detailed Description . . . . .	399
10.72.2 Constructor & Destructor Documentation . . . . .	399
10.72.2.1 Light . . . . .	399
10.72.2.2 ~Light . . . . .	399
10.72.3 Member Function Documentation . . . . .	399
10.72.3.1 FillMsg . . . . .	399
10.72.3.2 GetDiffuseColor . . . . .	400
10.72.3.3 GetDirection . . . . .	400
10.72.3.4 GetName . . . . .	400
10.72.3.5 GetPosition . . . . .	400

10.72.3.6	GetSpecularColor	400
10.72.3.7	GetType	400
10.72.3.8	Load	401
10.72.3.9	Load	401
10.72.3.10	LoadFromMsg	401
10.72.3.11	OnPoseChange	401
10.72.3.12	SetAttenuation	401
10.72.3.13	SetCastShadows	401
10.72.3.14	SetDiffuseColor	401
10.72.3.15	SetDirection	402
10.72.3.16	SetLightType	402
10.72.3.17	SetName	402
10.72.3.18	SetPosition	402
10.72.3.19	SetRange	402
10.72.3.20	SetSelected	402
10.72.3.21	SetSpecularColor	403
10.72.3.22	SetSpotFalloff	403
10.72.3.23	SetSpotInnerAngle	403
10.72.3.24	SetSpotOuterAngle	403
10.72.3.25	ShowVisual	403
10.72.3.26	ToggleShowVisual	403
10.72.3.27	UpdateFromMsg	404
10.73	gazebo::physics::Link Class Reference	404
10.73.1	Detailed Description	408
10.73.2	Constructor & Destructor Documentation	408
10.73.2.1	Link	408
10.73.2.2	~Link	408
10.73.3	Member Function Documentation	408
10.73.3.1	AddChildJoint	408
10.73.3.2	AddForce	409
10.73.3.3	AddForceAtRelativePosition	409
10.73.3.4	AddForceAtWorldPosition	409
10.73.3.5	AddParentJoint	409
10.73.3.6	AddRelativeForce	409
10.73.3.7	AddRelativeTorque	409
10.73.3.8	AddTorque	410
10.73.3.9	AttachStaticModel	410

10.73.3.10ConnectEnabled . . . . .	410
10.73.3.11DetachAllStaticModels . . . . .	410
10.73.3.12DetachStaticModel . . . . .	410
10.73.3.13DisconnectEnabled . . . . .	411
10.73.3.14FillMsg . . . . .	411
10.73.3.15Fini . . . . .	411
10.73.3.16GetAngularDamping . . . . .	411
10.73.3.17GetBoundingBox . . . . .	411
10.73.3.18GetChildJointsLinks . . . . .	411
10.73.3.19GetCollision . . . . .	412
10.73.3.20GetCollision . . . . .	412
10.73.3.21GetCollisionByld . . . . .	412
10.73.3.22GetCollisions . . . . .	412
10.73.3.23GetEnabled . . . . .	412
10.73.3.24GetGravityMode . . . . .	413
10.73.3.25GetInertial . . . . .	413
10.73.3.26GetKinematic . . . . .	413
10.73.3.27GetLinearDamping . . . . .	413
10.73.3.28GetModel . . . . .	413
10.73.3.29GetParentJointsLinks . . . . .	413
10.73.3.30GetRelativeAngularAccel . . . . .	414
10.73.3.31GetRelativeAngularVel . . . . .	414
10.73.3.32GetRelativeForce . . . . .	414
10.73.3.33GetRelativeLinearAccel . . . . .	414
10.73.3.34GetRelativeLinearVel . . . . .	414
10.73.3.35GetRelativeTorque . . . . .	415
10.73.3.36GetSelfCollide . . . . .	415
10.73.3.37GetSensorCount . . . . .	415
10.73.3.38GetSensorName . . . . .	415
10.73.3.39GetWorldAngularAccel . . . . .	415
10.73.3.40GetWorldForce . . . . .	416
10.73.3.41GetWorldLinearAccel . . . . .	416
10.73.3.42GetWorldTorque . . . . .	416
10.73.3.43Init . . . . .	416
10.73.3.44Load . . . . .	416
10.73.3.45OnPoseChange . . . . .	417
10.73.3.46ProcessMsg . . . . .	417

10.73.3.47RemoveChildJoint . . . . .	417
10.73.3.48RemoveParentJoint . . . . .	417
10.73.3.49Reset . . . . .	417
10.73.3.50SetAngularAccel . . . . .	417
10.73.3.51SetAngularDamping . . . . .	417
10.73.3.52SetAngularVel . . . . .	418
10.73.3.53SetAutoDisable . . . . .	418
10.73.3.54SetCollideMode . . . . .	418
10.73.3.55SetEnabled . . . . .	418
10.73.3.56SetForce . . . . .	418
10.73.3.57SetGravityMode . . . . .	419
10.73.3.58SetInertial . . . . .	419
10.73.3.59SetKinematic . . . . .	419
10.73.3.60SetLaserRetro . . . . .	419
10.73.3.61SetLinearAccel . . . . .	419
10.73.3.62SetLinearDamping . . . . .	419
10.73.3.63SetLinearVel . . . . .	420
10.73.3.64SetSelected . . . . .	420
10.73.3.65SetSelfCollide . . . . .	420
10.73.3.66SetState . . . . .	420
10.73.3.67SetTorque . . . . .	420
10.73.3.68Update . . . . .	420
10.73.3.69UpdateMass . . . . .	421
10.73.3.70UpdateParameters . . . . .	421
10.73.3.71UpdateSurface . . . . .	421
10.73.4 Member Data Documentation . . . . .	421
10.73.4.1 angularAccel . . . . .	421
10.73.4.2 attachedModelsOffset . . . . .	421
10.73.4.3 cgVisuals . . . . .	421
10.73.4.4 inertial . . . . .	421
10.73.4.5 linearAccel . . . . .	421
10.73.4.6 visuals . . . . .	421
10.74gazebo::physics::LinkState Class Reference . . . . .	422
10.74.1 Detailed Description . . . . .	423
10.74.2 Constructor & Destructor Documentation . . . . .	423
10.74.2.1 LinkState . . . . .	423
10.74.2.2 LinkState . . . . .	423

---

10.74.2.3 LinkState . . . . .	423
10.74.2.4 ~LinkState . . . . .	424
10.74.3 Member Function Documentation . . . . .	424
10.74.3.1 FillSDF . . . . .	424
10.74.3.2 GetAcceleration . . . . .	424
10.74.3.3 GetCollisionState . . . . .	424
10.74.3.4 GetCollisionState . . . . .	424
10.74.3.5 GetCollisionStateCount . . . . .	425
10.74.3.6 GetCollisionStates . . . . .	425
10.74.3.7 GetPose . . . . .	425
10.74.3.8 GetVelocity . . . . .	425
10.74.3.9 GetWrench . . . . .	426
10.74.3.10 IsZero . . . . .	426
10.74.3.11 Load . . . . .	426
10.74.3.12 operator+ . . . . .	426
10.74.3.13 operator- . . . . .	426
10.74.3.14 operator= . . . . .	427
10.74.4 Friends And Related Function Documentation . . . . .	427
10.74.4.1 operator<< . . . . .	427
10.75 gazebo::common::LogPlay Class Reference . . . . .	427
10.75.1 Member Function Documentation . . . . .	428
10.75.1.1 GetChunk . . . . .	428
10.75.1.2 GetChunkCount . . . . .	429
10.75.1.3 GetEncoding . . . . .	429
10.75.1.4 GetGazeboVersion . . . . .	429
10.75.1.5 GetLogVersion . . . . .	429
10.75.1.6 GetRandSeed . . . . .	429
10.75.1.7 IsOpen . . . . .	430
10.75.1.8 Open . . . . .	430
10.75.1.9 Step . . . . .	430
10.76 Logplay Class Reference . . . . .	430
10.76.1 Detailed Description . . . . .	430
10.77 gazebo::common::LogRecord Class Reference . . . . .	431
10.77.1 Detailed Description . . . . .	431
10.77.2 Member Function Documentation . . . . .	432
10.77.2.1 Add . . . . .	432
10.77.2.2 GetEncoding . . . . .	432

10.77.2.3	GetRunning	432
10.77.2.4	Init	433
10.77.2.5	Remove	433
10.77.2.6	Start	433
10.77.2.7	Stop	433
10.78	gazebo::Master Class Reference	433
10.78.1	Detailed Description	434
10.78.2	Constructor & Destructor Documentation	434
10.78.2.1	Master	434
10.78.2.2	~Master	434
10.78.3	Member Function Documentation	434
10.78.3.1	Fini	434
10.78.3.2	Init	434
10.78.3.3	Run	435
10.78.3.4	RunOnce	435
10.78.3.5	RunThread	435
10.78.3.6	Stop	435
10.79	gazebo::common::Material Class Reference	435
10.79.1	Detailed Description	437
10.79.2	Member Enumeration Documentation	438
10.79.2.1	BlendMode	438
10.79.2.2	ShadeMode	438
10.79.3	Constructor & Destructor Documentation	438
10.79.3.1	Material	438
10.79.3.2	~Material	438
10.79.3.3	Material	438
10.79.4	Member Function Documentation	438
10.79.4.1	GetAmbient	438
10.79.4.2	GetBlendFactors	439
10.79.4.3	GetBlendMode	439
10.79.4.4	GetDepthWrite	439
10.79.4.5	GetDiffuse	439
10.79.4.6	GetEmissive	439
10.79.4.7	GetLighting	439
10.79.4.8	GetName	440
10.79.4.9	GetPointSize	440
10.79.4.10	GetShadeMode	440



10.79.4.11	GetShininess . . . . .	440
10.79.4.12	GetSpecular . . . . .	440
10.79.4.13	GetTextureImage . . . . .	440
10.79.4.14	GetTransparency . . . . .	441
10.79.4.15	SetAmbient . . . . .	441
10.79.4.16	SetBlendFactors . . . . .	441
10.79.4.17	SetBlendMode . . . . .	441
10.79.4.18	SetDepthWrite . . . . .	441
10.79.4.19	SetDiffuse . . . . .	441
10.79.4.20	SetEmissive . . . . .	442
10.79.4.21	SetLighting . . . . .	442
10.79.4.22	SetPointSize . . . . .	442
10.79.4.23	SetShadeMode . . . . .	442
10.79.4.24	SetShininess . . . . .	442
10.79.4.25	SetSpecular . . . . .	442
10.79.4.26	SetTextureImage . . . . .	443
10.79.4.27	SetTextureImage . . . . .	443
10.79.4.28	SetTransparency . . . . .	443
10.79.5	Friends And Related Function Documentation . . . . .	443
10.79.5.1	operator<< . . . . .	443
10.79.6	Member Data Documentation . . . . .	443
10.79.6.1	ambient . . . . .	443
10.79.6.2	blendMode . . . . .	443
10.79.6.3	BlendModeStr . . . . .	443
10.79.6.4	diffuse . . . . .	443
10.79.6.5	emissive . . . . .	444
10.79.6.6	name . . . . .	444
10.79.6.7	pointSize . . . . .	444
10.79.6.8	shadeMode . . . . .	444
10.79.6.9	ShadeModeStr . . . . .	444
10.79.6.10	shininess . . . . .	444
10.79.6.11	specular . . . . .	444
10.79.6.12	texImage . . . . .	444
10.79.6.13	transparency . . . . .	444
10.80	gazebo::math::Matrix3 Class Reference . . . . .	444
10.80.1	Detailed Description . . . . .	445
10.80.2	Constructor & Destructor Documentation . . . . .	445

10.80.2.1 Matrix3 . . . . .	445
10.80.2.2 Matrix3 . . . . .	446
10.80.2.3 Matrix3 . . . . .	446
10.80.2.4 ~Matrix3 . . . . .	446
10.80.3 Member Function Documentation . . . . .	446
10.80.3.1 operator== . . . . .	446
10.80.3.2 operator[] . . . . .	446
10.80.3.3 operator[] . . . . .	447
10.80.3.4 SetCol . . . . .	447
10.80.3.5 SetFromAxes . . . . .	447
10.80.3.6 SetFromAxis . . . . .	447
10.80.4 Friends And Related Function Documentation . . . . .	447
10.80.4.1 operator<< . . . . .	448
10.80.5 Member Data Documentation . . . . .	448
10.80.5.1 m . . . . .	448
10.81 gazebo::math::Matrix4 Class Reference . . . . .	448
10.81.1 Detailed Description . . . . .	450
10.81.2 Constructor & Destructor Documentation . . . . .	450
10.81.2.1 Matrix4 . . . . .	450
10.81.2.2 Matrix4 . . . . .	450
10.81.2.3 Matrix4 . . . . .	450
10.81.2.4 ~Matrix4 . . . . .	450
10.81.3 Member Function Documentation . . . . .	450
10.81.3.1 GetAsPose . . . . .	451
10.81.3.2 GetEulerRotation . . . . .	451
10.81.3.3 GetRotation . . . . .	451
10.81.3.4 GetTranslation . . . . .	451
10.81.3.5 Inverse . . . . .	451
10.81.3.6 IsAffine . . . . .	451
10.81.3.7 operator* . . . . .	451
10.81.3.8 operator* . . . . .	452
10.81.3.9 operator* . . . . .	452
10.81.3.10 operator= . . . . .	452
10.81.3.11 operator= . . . . .	452
10.81.3.12 operator== . . . . .	453
10.81.3.13 operator[] . . . . .	453
10.81.3.14 operator[] . . . . .	453

10.81.3.15Set . . . . .	453
10.81.3.16SetScale . . . . .	454
10.81.3.17SetTranslate . . . . .	454
10.81.3.18TransformAffine . . . . .	454
10.81.4 Friends And Related Function Documentation . . . . .	454
10.81.4.1 operator<< . . . . .	455
10.81.5 Member Data Documentation . . . . .	455
10.81.5.1 IDENTITY . . . . .	455
10.81.5.2 m . . . . .	455
10.81.5.3 ZERO . . . . .	455
10.82gazebo::common::Mesh Class Reference . . . . .	455
10.82.1 Detailed Description . . . . .	457
10.82.2 Constructor & Destructor Documentation . . . . .	457
10.82.2.1 Mesh . . . . .	457
10.82.2.2 ~Mesh . . . . .	457
10.82.3 Member Function Documentation . . . . .	457
10.82.3.1 AddMaterial . . . . .	457
10.82.3.2 AddSubMesh . . . . .	457
10.82.3.3 FillArrays . . . . .	457
10.82.3.4 GenSphericalTexCoord . . . . .	458
10.82.3.5 GetAABB . . . . .	458
10.82.3.6 GetIndexCount . . . . .	458
10.82.3.7 GetMaterial . . . . .	458
10.82.3.8 GetMaterialCount . . . . .	458
10.82.3.9 GetMax . . . . .	458
10.82.3.10GetMin . . . . .	459
10.82.3.11GetName . . . . .	459
10.82.3.12GetNormalCount . . . . .	459
10.82.3.13GetPath . . . . .	459
10.82.3.14GetSkeleton . . . . .	459
10.82.3.15GetSubMesh . . . . .	459
10.82.3.16GetSubMeshCount . . . . .	460
10.82.3.17GetTexCoordCount . . . . .	460
10.82.3.18GetVertexCount . . . . .	460
10.82.3.19HasSkeleton . . . . .	460
10.82.3.20RecalculateNormals . . . . .	460
10.82.3.21Scale . . . . .	460

10.82.3.22	SetName	461
10.82.3.23	SetPath	461
10.82.3.24	SetScale	461
10.82.3.25	SetSkeleton	461
10.83	gazebo::common::MeshCSG Class Reference	461
10.83.1	Detailed Description	462
10.83.2	Member Enumeration Documentation	462
10.83.2.1	BooleanOperation	462
10.83.3	Constructor & Destructor Documentation	462
10.83.3.1	MeshCSG	462
10.83.3.2	~MeshCSG	462
10.83.4	Member Function Documentation	462
10.83.4.1	CreateBoolean	462
10.84	gazebo::common::MeshLoader Class Reference	463
10.84.1	Detailed Description	463
10.84.2	Constructor & Destructor Documentation	463
10.84.2.1	MeshLoader	463
10.84.2.2	~MeshLoader	463
10.84.3	Member Function Documentation	464
10.84.3.1	Load	464
10.85	gazebo::common::MeshManager Class Reference	464
10.85.1	Detailed Description	465
10.85.2	Member Function Documentation	465
10.85.2.1	AddMesh	465
10.85.2.2	CreateBox	466
10.85.2.3	CreateCamera	466
10.85.2.4	CreateCone	466
10.85.2.5	CreateCylinder	466
10.85.2.6	CreatePlane	466
10.85.2.7	CreatePlane	467
10.85.2.8	CreateSphere	467
10.85.2.9	CreateTube	467
10.85.2.10	GenSphericalTexCoord	468
10.85.2.11	GetMesh	468
10.85.2.12	GetMeshAABB	468
10.85.2.13	HasMesh	468
10.85.2.14	IsValidFilename	468

---

10.85.2.15	Load	468
10.86	gazebo::physics::Model Class Reference	469
10.86.1	Detailed Description	472
10.86.2	Constructor & Destructor Documentation	472
10.86.2.1	Model	472
10.86.2.2	~Model	472
10.86.3	Member Function Documentation	472
10.86.3.1	AttachStaticModel	473
10.86.3.2	DetachStaticModel	473
10.86.3.3	FillMsg	473
10.86.3.4	Fini	473
10.86.3.5	GetAutoDisable	473
10.86.3.6	GetBoundingBox	474
10.86.3.7	GetJoint	474
10.86.3.8	GetJointController	474
10.86.3.9	GetJointCount	474
10.86.3.10	GetJoints	474
10.86.3.11	GetLink	475
10.86.3.12	GetLinkById	475
10.86.3.13	GetLinks	475
10.86.3.14	GetPluginCount	475
10.86.3.15	GetRelativeAngularAccel	475
10.86.3.16	GetRelativeAngularVel	476
10.86.3.17	GetRelativeLinearAccel	476
10.86.3.18	GetRelativeLinearVel	476
10.86.3.19	GetSDF	476
10.86.3.20	GetSensorCount	476
10.86.3.21	GetWorldAngularAccel	477
10.86.3.22	GetWorldAngularVel	477
10.86.3.23	GetWorldLinearAccel	477
10.86.3.24	GetWorldLinearVel	477
10.86.3.25	Init	477
10.86.3.26	Load	477
10.86.3.27	LoadJoints	478
10.86.3.28	LoadPlugins	478
10.86.3.29	OnPoseChange	478
10.86.3.30	ProcessMsg	478

10.86.3.31RemoveChild . . . . .	478
10.86.3.32Reset . . . . .	478
10.86.3.33SetAngularAccel . . . . .	478
10.86.3.34SetAngularVel . . . . .	479
10.86.3.35SetAutoDisable . . . . .	479
10.86.3.36SetCollideMode . . . . .	479
10.86.3.37SetEnabled . . . . .	479
10.86.3.38SetGravityMode . . . . .	479
10.86.3.39SetJointAnimation . . . . .	480
10.86.3.40SetJointPosition . . . . .	480
10.86.3.41SetJointPositions . . . . .	480
10.86.3.42SetLaserRetro . . . . .	480
10.86.3.43SetLinearAccel . . . . .	480
10.86.3.44SetLinearVel . . . . .	481
10.86.3.45SetLinkWorldPose . . . . .	481
10.86.3.46SetLinkWorldPose . . . . .	481
10.86.3.47SetState . . . . .	481
10.86.3.48StopAnimation . . . . .	481
10.86.3.49Update . . . . .	482
10.86.3.50UpdateParameters . . . . .	482
10.86.4 Member Data Documentation . . . . .	482
10.86.4.1 attachedModels . . . . .	482
10.86.4.2 attachedModelsOffset . . . . .	482
10.87gazebo::common::ModelDatabase Class Reference . . . . .	482
10.87.1 Detailed Description . . . . .	483
10.88gazebo::ModelPlugin Class Reference . . . . .	484
10.88.1 Detailed Description . . . . .	484
10.88.2 Constructor & Destructor Documentation . . . . .	484
10.88.2.1 ModelPlugin . . . . .	485
10.88.2.2 ~ModelPlugin . . . . .	485
10.88.3 Member Function Documentation . . . . .	485
10.88.3.1 Init . . . . .	485
10.88.3.2 Load . . . . .	485
10.88.3.3 Reset . . . . .	485
10.89gazebo::physics::ModelState Class Reference . . . . .	485
10.89.1 Detailed Description . . . . .	487
10.89.2 Constructor & Destructor Documentation . . . . .	487

10.89.2.1 ModelState . . . . .	487
10.89.2.2 ModelState . . . . .	487
10.89.2.3 ModelState . . . . .	487
10.89.2.4 ~ModelState . . . . .	488
10.89.3 Member Function Documentation . . . . .	488
10.89.3.1 FillSDF . . . . .	488
10.89.3.2 GetJointState . . . . .	488
10.89.3.3 GetJointState . . . . .	488
10.89.3.4 GetJointStateCount . . . . .	489
10.89.3.5 GetJointStates . . . . .	489
10.89.3.6 GetLinkState . . . . .	489
10.89.3.7 GetLinkState . . . . .	489
10.89.3.8 GetLinkStateCount . . . . .	490
10.89.3.9 GetLinkStates . . . . .	490
10.89.3.10 GetPose . . . . .	490
10.89.3.11 HasJointState . . . . .	490
10.89.3.12 HasLinkState . . . . .	491
10.89.3.13 IsZero . . . . .	491
10.89.3.14 Load . . . . .	491
10.89.3.15 operator+ . . . . .	491
10.89.3.16 operator- . . . . .	492
10.89.3.17 operator= . . . . .	492
10.89.4 Friends And Related Function Documentation . . . . .	492
10.89.4.1 operator<< . . . . .	492
10.90 gazebo::common::MouseEvent Class Reference . . . . .	492
10.90.1 Detailed Description . . . . .	493
10.90.2 Member Enumeration Documentation . . . . .	494
10.90.2.1 Buttons . . . . .	494
10.90.2.2 EventType . . . . .	494
10.90.3 Constructor & Destructor Documentation . . . . .	494
10.90.3.1 MouseEvent . . . . .	494
10.90.4 Member Data Documentation . . . . .	494
10.90.4.1 alt . . . . .	494
10.90.4.2 button . . . . .	494
10.90.4.3 buttons . . . . .	494
10.90.4.4 control . . . . .	494
10.90.4.5 dragging . . . . .	495

10.90.4.6	moveScale	495
10.90.4.7	pos	495
10.90.4.8	pressPos	495
10.90.4.9	prevPos	495
10.90.4.10	scroll	495
10.90.4.11	shift	495
10.90.4.12	type	495
10.91	gazebo::rendering::MovableText Class Reference	495
10.91.1	Detailed Description	497
10.91.2	Member Enumeration Documentation	497
10.91.2.1	HorizAlign	497
10.91.2.2	VertAlign	498
10.91.3	Constructor & Destructor Documentation	498
10.91.3.1	MovableText	498
10.91.3.2	~MovableText	498
10.91.4	Member Function Documentation	498
10.91.4.1	_setupGeometry	498
10.91.4.2	_updateColors	498
10.91.4.3	GetAABB	498
10.91.4.4	GetBaseline	498
10.91.4.5	getBoundingRadius	498
10.91.4.6	GetCharHeight	498
10.91.4.7	GetColor	499
10.91.4.8	GetFont	499
10.91.4.9	getLights	499
10.91.4.10	getMaterial	499
10.91.4.11	getRenderOperation	499
10.91.4.12	GetShowOnTop	499
10.91.4.13	GetSpaceWidth	499
10.91.4.14	getSquaredViewDepth	499
10.91.4.15	GetText	499
10.91.4.16	getWorldTransforms	500
10.91.4.17	Load	500
10.91.4.18	SetBaseline	500
10.91.4.19	SetCharHeight	500
10.91.4.20	SetColor	500
10.91.4.21	SetFontName	500



---

10.91.4.22	SetShowOnTop	501
10.91.4.23	SetSpaceWidth	501
10.91.4.24	SetText	501
10.91.4.25	SetTextAlignment	501
10.91.4.26	Update	501
10.91.4.27	visitRenderables	501
10.92	gazebo::msgs::MsgFactory Class Reference	501
10.92.1	Detailed Description	502
10.92.2	Member Function Documentation	502
10.92.2.1	GetMsgTypes	502
10.92.2.2	NewMsg	502
10.92.2.3	RegisterMsg	502
10.93	gazebo::sensors::MultiCameraSensor Class Reference	503
10.93.1	Detailed Description	504
10.93.2	Constructor & Destructor Documentation	504
10.93.2.1	MultiCameraSensor	504
10.93.2.2	~MultiCameraSensor	504
10.93.3	Member Function Documentation	504
10.93.3.1	Fini	504
10.93.3.2	GetCamera	504
10.93.3.3	GetCameraCount	505
10.93.3.4	GetImageData	505
10.93.3.5	GetImageHeight	505
10.93.3.6	GetImageWidth	506
10.93.3.7	GetTopic	506
10.93.3.8	Init	506
10.93.3.9	Load	506
10.93.3.10	SaveFrame	506
10.93.3.11	UpdateImpl	507
10.94	gazebo::physics::MultiRayShape Class Reference	507
10.94.1	Detailed Description	510
10.94.2	Constructor & Destructor Documentation	510
10.94.2.1	MultiRayShape	510
10.94.2.2	~MultiRayShape	510
10.94.3	Member Function Documentation	510
10.94.3.1	AddRay	510
10.94.3.2	ConnectNewLaserScans	510

10.94.3.3 DisconnectNewLaserScans . . . . .	511
10.94.3.4 FillMsg . . . . .	511
10.94.3.5 GetFiducial . . . . .	511
10.94.3.6 GetMaxAngle . . . . .	511
10.94.3.7 GetMaxRange . . . . .	511
10.94.3.8 GetMinAngle . . . . .	512
10.94.3.9 GetMinRange . . . . .	512
10.94.3.10GetRange . . . . .	512
10.94.3.11GetResRange . . . . .	512
10.94.3.12GetRetro . . . . .	512
10.94.3.13GetSampleCount . . . . .	513
10.94.3.14GetScanResolution . . . . .	513
10.94.3.15GetVerticalMaxAngle . . . . .	513
10.94.3.16GetVerticalMinAngle . . . . .	513
10.94.3.17GetVerticalSampleCount . . . . .	513
10.94.3.18GetVerticalScanResolution . . . . .	513
10.94.3.19Init . . . . .	514
10.94.3.20ProcessMsg . . . . .	514
10.94.3.21Update . . . . .	514
10.94.3.22UpdateRays . . . . .	514
10.94.4 Member Data Documentation . . . . .	514
10.94.4.1 horzElem . . . . .	514
10.94.4.2 newLaserScans . . . . .	514
10.94.4.3 offset . . . . .	514
10.94.4.4 rangeElem . . . . .	514
10.94.4.5 rayElem . . . . .	515
10.94.4.6 rays . . . . .	515
10.94.4.7 scanElem . . . . .	515
10.94.4.8 vertElem . . . . .	515
10.95gazebo::transport::Node Class Reference . . . . .	515
10.95.1 Detailed Description . . . . .	516
10.95.2 Constructor & Destructor Documentation . . . . .	517
10.95.2.1 Node . . . . .	517
10.95.2.2 ~Node . . . . .	517
10.95.3 Member Function Documentation . . . . .	517
10.95.3.1 Advertise . . . . .	517
10.95.3.2 DecodeTopicName . . . . .	517

10.95.3.3 EncodeTopicName . . . . .	517
10.95.3.4 Fini . . . . .	518
10.95.3.5 GetId . . . . .	518
10.95.3.6 GetMsgType . . . . .	518
10.95.3.7 GetTopicNamespace . . . . .	518
10.95.3.8 HandleData . . . . .	518
10.95.3.9 HasLatchedSubscriber . . . . .	519
10.95.3.10Init . . . . .	519
10.95.3.11InsertLatchedMsg . . . . .	519
10.95.3.12ProcessIncoming . . . . .	519
10.95.3.13ProcessPublishers . . . . .	519
10.95.3.14RemoveCallback . . . . .	519
10.95.3.15Subscribe . . . . .	520
10.95.3.16Subscribe . . . . .	520
10.95.3.17Subscribe . . . . .	520
10.95.3.18Subscribe . . . . .	521
10.96gazebo::common::NodeAnimation Class Reference . . . . .	521
10.96.1 Detailed Description . . . . .	522
10.96.2 Constructor & Destructor Documentation . . . . .	522
10.96.2.1 NodeAnimation . . . . .	522
10.96.2.2 ~NodeAnimation . . . . .	522
10.96.3 Member Function Documentation . . . . .	522
10.96.3.1 AddKeyFrame . . . . .	523
10.96.3.2 AddKeyFrame . . . . .	523
10.96.3.3 GetFrameAt . . . . .	523
10.96.3.4 GetFrameCount . . . . .	523
10.96.3.5 GetKeyFrame . . . . .	523
10.96.3.6 GetKeyFrame . . . . .	524
10.96.3.7 GetLength . . . . .	524
10.96.3.8 GetName . . . . .	524
10.96.3.9 GetTimeAtX . . . . .	524
10.96.3.10Scale . . . . .	524
10.96.3.11SetName . . . . .	525
10.96.4 Member Data Documentation . . . . .	525
10.96.4.1 keyFrames . . . . .	525
10.96.4.2 length . . . . .	525
10.96.4.3 name . . . . .	525

10.97gazebo::common::NodeAssignment Struct Reference . . . . .	525
10.97.1 Detailed Description . . . . .	525
10.97.2 Member Data Documentation . . . . .	526
10.97.2.1 nodeIndex . . . . .	526
10.97.2.2 vertexIndex . . . . .	526
10.97.2.3 weight . . . . .	526
10.98gazebo::common::NodeTransform Class Reference . . . . .	526
10.98.1 Detailed Description . . . . .	527
10.98.2 Member Enumeration Documentation . . . . .	527
10.98.2.1 TransformType . . . . .	527
10.98.3 Constructor & Destructor Documentation . . . . .	528
10.98.3.1 NodeTransform . . . . .	528
10.98.3.2 NodeTransform . . . . .	528
10.98.3.3 ~NodeTransform . . . . .	528
10.98.4 Member Function Documentation . . . . .	528
10.98.4.1 Get . . . . .	528
10.98.4.2 GetSID . . . . .	528
10.98.4.3 GetType . . . . .	528
10.98.4.4 operator() . . . . .	529
10.98.4.5 operator* . . . . .	529
10.98.4.6 operator* . . . . .	529
10.98.4.7 PrintSource . . . . .	529
10.98.4.8 RecalculateMatrix . . . . .	529
10.98.4.9 Set . . . . .	529
10.98.4.10SetComponent . . . . .	530
10.98.4.11SetSID . . . . .	530
10.98.4.12SetSourceValues . . . . .	530
10.98.4.13SetSourceValues . . . . .	530
10.98.4.14SetSourceValues . . . . .	530
10.98.4.15SetType . . . . .	530
10.98.5 Member Data Documentation . . . . .	530
10.98.5.1 sid . . . . .	530
10.98.5.2 source . . . . .	531
10.98.5.3 transform . . . . .	531
10.98.5.4 type . . . . .	531
10.99gazebo::common::NumericAnimation Class Reference . . . . .	531
10.99.1 Detailed Description . . . . .	532

10.99.2	Constructor & Destructor Documentation . . . . .	532
10.99.2.1	NumericAnimation . . . . .	532
10.99.2.2	~NumericAnimation . . . . .	532
10.99.3	Member Function Documentation . . . . .	532
10.99.3.1	CreateKeyFrame . . . . .	532
10.99.3.2	GetInterpolatedKeyFrame . . . . .	532
10.100	gazebo::common::NumericKeyFrame Class Reference . . . . .	533
10.100.1	Detailed Description . . . . .	533
10.100.2	Constructor & Destructor Documentation . . . . .	533
10.100.2.1	NumericKeyFrame . . . . .	533
10.100.2.2	~NumericKeyFrame . . . . .	534
10.100.3	Member Function Documentation . . . . .	534
10.100.3.1	GetValue . . . . .	534
10.100.3.2	SetValue . . . . .	534
10.100.4	Member Data Documentation . . . . .	534
10.100.4.1	value . . . . .	534
10.101	gazebo::rendering::OrbitViewController Class Reference . . . . .	534
10.101.1	Detailed Description . . . . .	536
10.101.2	Constructor & Destructor Documentation . . . . .	536
10.101.2.1	OrbitViewController . . . . .	536
10.101.2.2	~OrbitViewController . . . . .	536
10.101.3	Member Function Documentation . . . . .	536
10.101.3.1	GetFocalPoint . . . . .	536
10.101.3.2	GetTypeString . . . . .	536
10.101.3.3	HandleKeyPressEvent . . . . .	536
10.101.3.4	HandleKeyReleaseEvent . . . . .	537
10.101.3.5	HandleMouseEvent . . . . .	537
10.101.3.6	Init . . . . .	537
10.101.3.7	Init . . . . .	537
10.101.3.8	SetDistance . . . . .	537
10.101.3.9	SetFocalPoint . . . . .	537
10.101.3.10	Update . . . . .	538
10.102	df::Param Class Reference . . . . .	538
10.102.1	Detailed Description . . . . .	540
10.102.2	Constructor & Destructor Documentation . . . . .	540
10.102.2.1	Param . . . . .	540
10.102.2.2	~Param . . . . .	540

10.102.3	Member Function Documentation . . . . .	540
10.102.3.1	Clone . . . . .	540
10.102.3.2	Get . . . . .	540
10.102.3.3	Get . . . . .	540
10.102.3.4	Get . . . . .	540
10.102.3.5	Get . . . . .	540
10.102.3.6	Get . . . . .	540
10.102.3.7	Get . . . . .	540
10.102.3.8	Get . . . . .	541
10.102.3.9	Get . . . . .	541
10.102.3.10	Get . . . . .	541
10.102.3.11	Get . . . . .	541
10.102.3.12	Get . . . . .	541
10.102.3.13	Get . . . . .	541
10.102.3.14	Get . . . . .	541
10.102.3.15	Get . . . . .	541
10.102.3.16	GetAsString . . . . .	541
10.102.3.17	GetDefaultAsString . . . . .	541
10.102.3.18	GetDescription . . . . .	541
10.102.3.19	GetKey . . . . .	541
10.102.3.20	GetRequired . . . . .	541
10.102.3.21	GetSet . . . . .	541
10.102.3.22	GetTypeName . . . . .	541
10.102.3.23	Bool . . . . .	541
10.102.3.24	Char . . . . .	541
10.102.3.25	Color . . . . .	542
10.102.3.26	Double . . . . .	542
10.102.3.27	Float . . . . .	542
10.102.3.28	Int . . . . .	542
10.102.3.29	Pose . . . . .	542
10.102.3.30	Quaternion . . . . .	542
10.102.3.31	Str . . . . .	542
10.102.3.32	Time . . . . .	542
10.102.3.33	UInt . . . . .	542
10.102.3.34	Vector2d . . . . .	542
10.102.3.35	Vector2i . . . . .	542
10.102.3.36	Vector3 . . . . .	542

10.102.3.37	reset . . . . .	542
10.102.3.38	set . . . . .	542
10.102.3.39	set . . . . .	542
10.102.3.40	set . . . . .	542
10.102.3.41	set . . . . .	542
10.102.3.42	set . . . . .	542
10.102.3.43	set . . . . .	542
10.102.3.44	set . . . . .	542
10.102.3.45	set . . . . .	542
10.102.3.46	set . . . . .	542
10.102.3.47	set . . . . .	542
10.102.3.48	set . . . . .	543
10.102.3.49	set . . . . .	543
10.102.3.50	set . . . . .	543
10.102.3.51	set . . . . .	543
10.102.3.52	set . . . . .	543
10.102.3.53	setDescription . . . . .	543
10.102.3.54	setFromString . . . . .	543
10.102.3.55	setUpdateFunc . . . . .	543
10.102.3.56	update . . . . .	543
10.102.4	Member Data Documentation . . . . .	543
10.102.4.1	description . . . . .	543
10.102.4.2	key . . . . .	543
10.102.4.3	required . . . . .	543
10.102.4.4	set . . . . .	543
10.102.4.5	typeName . . . . .	544
10.102.4.6	updateFunc . . . . .	544
10.104	ParamT< T > Class Template Reference . . . . .	544
10.104.1	df::ParamT< T > Class Template Reference . . . . .	544
10.104.1	Detailed Description . . . . .	545
10.104.2	Constructor & Destructor Documentation . . . . .	545
10.104.2.1	ParamT . . . . .	545
10.104.2.2	~ParamT . . . . .	545
10.104.3	Member Function Documentation . . . . .	546
10.104.3.1	Clone . . . . .	546
10.104.3.2	getAsString . . . . .	546
10.104.3.3	getDefaultAsString . . . . .	546

10.104.3.4	GetDefaultValue	546
10.104.3.5	GetValue	546
10.104.3.6	operator*	546
10.104.3.7	Reset	546
10.104.3.8	Set	546
10.104.3.9	SetFromString	546
10.104.3.10	SetValue	547
10.104.3.11	Update	547
10.104.4	Friends And Related Function Documentation	547
10.104.4.1	operator<<	547
10.104.5	Member Data Documentation	547
10.104.5.1	defaultValue	547
10.104.5.2	value	547
10.105	Gazebo::physics::PhysicsEngine Class Reference	547
10.105.1	Detailed Description	550
10.105.2	Constructor & Destructor Documentation	550
10.105.2.1	PhysicsEngine	550
10.105.2.2	~PhysicsEngine	550
10.105.3	Member Function Documentation	550
10.105.3.1	CreateCollision	550
10.105.3.2	CreateCollision	550
10.105.3.3	CreateJoint	551
10.105.3.4	CreateLink	551
10.105.3.5	CreateShape	551
10.105.3.6	DebugPrint	551
10.105.3.7	Finis	551
10.105.3.8	GetAutoDisableFlag	551
10.105.3.9	GetContactManager	551
10.105.3.10	GetContactMaxCorrectingVel	552
10.105.3.11	GetContactSurfaceLayer	552
10.105.3.12	GetGravity	552
10.105.3.13	GetMaxContacts	552
10.105.3.14	GetPhysicsUpdateMutex	552
10.105.3.15	GetSORPGSIters	553
10.105.3.16	GetSORPGSPreconIters	553
10.105.3.17	GetSORPGSW	553
10.105.3.18	GetStepTime	553



10.105.3.10	GetUpdatePeriod	553
10.105.3.20	GetUpdateRate	553
10.105.3.23	GetWorldCFM	554
10.105.3.24	GetWorldERP	554
10.105.3.26	Hit	554
10.105.3.24	WaitForThread	554
10.105.3.25	Load	554
10.105.3.26	OnPhysicsMsg	554
10.105.3.27	OnRequest	554
10.105.3.28	Reset	555
10.105.3.29	SetAutoDisableFlag	555
10.105.3.30	SetContactMaxCorrectingVel	555
10.105.3.33	SetContactSurfaceLayer	555
10.105.3.32	SetGravity	555
10.105.3.33	SetMaxContacts	555
10.105.3.33	SetSeed	556
10.105.3.35	SetSORPGSIlters	556
10.105.3.36	SetSORPGSPreconlters	556
10.105.3.37	SetSORPGSW	556
10.105.3.38	SetStepTime	556
10.105.3.39	SetUpdateRate	557
10.105.3.40	SetWorldCFM	557
10.105.3.41	SetWorldERP	557
10.105.3.42	UpdateCollision	557
10.105.3.43	UpdatePhysics	557
10.105.4	Member Data Documentation	557
10.105.4.1	contactManager	557
10.105.4.2	node	557
10.105.4.3	physicsSub	558
10.105.4.4	physicsUpdateMutex	558
10.105.4.5	requestSub	558
10.105.4.6	responsePub	558
10.105.4.7	sdf	558
10.105.4.8	world	558
10.106	gazebo::physics::PhysicsFactory Class Reference	558
10.106.1	Detailed Description	558
10.106.2	Member Function Documentation	559

10.106.2.1	NewPhysicsEngine	559
10.106.2.2	RegisterAll	559
10.106.2.3	RegisterPhysicsEngine	559
10.107	gazebo::common::PID Class Reference	559
10.107.1	Detailed Description	560
10.107.2	Constructor & Destructor Documentation	560
10.107.2.1	PID	560
10.107.2.2	~PID	561
10.107.3	Member Function Documentation	561
10.107.3.1	GetCmd	561
10.107.3.2	GetErrors	561
10.107.3.3	Init	561
10.107.3.4	operator=	561
10.107.3.5	Reset	562
10.107.3.6	SetCmd	562
10.107.3.7	SetCmdMax	562
10.107.3.8	SetCmdMin	562
10.107.3.9	SetDGain	562
10.107.3.10	SetIGain	562
10.107.3.11	SetIMax	562
10.107.3.12	SetIMin	563
10.107.3.13	SetPGain	563
10.107.3.14	Update	563
10.108	gazebo::math::Plane Class Reference	563
10.108.1	Detailed Description	564
10.108.2	Constructor & Destructor Documentation	564
10.108.2.1	Plane	564
10.108.2.2	Plane	564
10.108.2.3	Plane	564
10.108.2.4	~Plane	565
10.108.3	Member Function Documentation	565
10.108.3.1	Distance	565
10.108.3.2	operator=	565
10.108.3.3	Set	565
10.108.4	Member Data Documentation	565
10.108.4.1	id	566
10.108.4.2	normal	566

10.108.4.3	size . . . . .	566
10.109	gazebo::physics::PlaneShape Class Reference . . . . .	566
10.109.1	Detailed Description . . . . .	567
10.109.2	Constructor & Destructor Documentation . . . . .	567
10.109.2.1	PlaneShape . . . . .	567
10.109.2.2	~PlaneShape . . . . .	567
10.109.3	Member Function Documentation . . . . .	568
10.109.3.1	CreatePlane . . . . .	568
10.109.3.2	FillMsg . . . . .	568
10.109.3.3	GetNormal . . . . .	568
10.109.3.4	GetSize . . . . .	568
10.109.3.5	Init . . . . .	568
10.109.3.6	ProcessMsg . . . . .	568
10.109.3.7	SetAltitude . . . . .	569
10.109.3.8	SetNormal . . . . .	569
10.109.3.9	SetSize . . . . .	569
10.110	df::Plugin Class Reference . . . . .	569
10.110.1	Constructor & Destructor Documentation . . . . .	570
10.110.1.1	Plugin . . . . .	570
10.110.2	Member Function Documentation . . . . .	570
10.110.2.1	Clear . . . . .	570
10.110.2.2	Print . . . . .	570
10.110.3	Member Data Documentation . . . . .	570
10.110.3.1	data . . . . .	570
10.110.3.2	filename . . . . .	570
10.110.3.3	name . . . . .	570
10.111	gazebo::PluginT< T > Class Template Reference . . . . .	571
10.111.1	Detailed Description . . . . .	571
10.111.2	Member Typedef Documentation . . . . .	571
10.111.2.1	TPtr . . . . .	571
10.111.3	Member Function Documentation . . . . .	572
10.111.3.1	Create . . . . .	572
10.111.3.2	GetFilename . . . . .	572
10.111.3.3	GetHandle . . . . .	572
10.111.3.4	GetType . . . . .	572
10.111.4	Member Data Documentation . . . . .	572
10.111.4.1	filename . . . . .	572

10.111.4.2	handle . . . . .	572
10.111.4.3	type . . . . .	573
10.112	gazebo::math::Pose Class Reference . . . . .	573
10.112.1	Detailed Description . . . . .	574
10.112.2	Constructor & Destructor Documentation . . . . .	575
10.112.2.1	Pose . . . . .	575
10.112.2.2	Pose . . . . .	575
10.112.2.3	Pose . . . . .	575
10.112.2.4	Pose . . . . .	575
10.112.2.5	~Pose . . . . .	575
10.112.3	Member Function Documentation . . . . .	575
10.112.3.1	CoordPoseSolve . . . . .	575
10.112.3.2	CoordPositionAdd . . . . .	576
10.112.3.3	CoordPositionAdd . . . . .	576
10.112.3.4	CoordPositionSub . . . . .	576
10.112.3.5	CoordRotationAdd . . . . .	576
10.112.3.6	CoordRotationSub . . . . .	577
10.112.3.7	Correct . . . . .	577
10.112.3.8	GetInverse . . . . .	577
10.112.3.9	IsFinite . . . . .	577
10.112.3.10	operator!= . . . . .	577
10.112.3.11	operator* . . . . .	578
10.112.3.12	operator+ . . . . .	578
10.112.3.13	operator+= . . . . .	578
10.112.3.14	operator- . . . . .	578
10.112.3.15	operator- . . . . .	579
10.112.3.16	operator-= . . . . .	579
10.112.3.17	operator== . . . . .	579
10.112.3.18	Reset . . . . .	579
10.112.3.19	RotatePositionAboutOrigin . . . . .	579
10.112.3.20	Round . . . . .	580
10.112.3.21	Set . . . . .	580
10.112.3.22	Set . . . . .	580
10.112.4	Friends And Related Function Documentation . . . . .	580
10.112.4.1	operator<< . . . . .	580
10.112.4.2	operator>> . . . . .	581
10.112.5	Member Data Documentation . . . . .	581

10.112.5.1	pos . . . . .	581
10.112.5.2	rot . . . . .	581
10.112.5.3	Zero . . . . .	581
10.113	gazebo::common::PoseAnimation Class Reference . . . . .	581
10.113.1	Detailed Description . . . . .	582
10.113.2	Constructor & Destructor Documentation . . . . .	582
10.113.2.1	PoseAnimation . . . . .	582
10.113.2.2	~PoseAnimation . . . . .	583
10.113.3	Member Function Documentation . . . . .	583
10.113.3.1	BuildInterpolationSplines . . . . .	583
10.113.3.2	CreateKeyFrame . . . . .	583
10.113.3.3	GetInterpolatedKeyFrame . . . . .	583
10.113.3.4	GetInterpolatedKeyFrame . . . . .	583
10.114	gazebo::common::PoseKeyFrame Class Reference . . . . .	584
10.114.1	Detailed Description . . . . .	584
10.114.2	Constructor & Destructor Documentation . . . . .	585
10.114.2.1	PoseKeyFrame . . . . .	585
10.114.2.2	~PoseKeyFrame . . . . .	585
10.114.3	Member Function Documentation . . . . .	585
10.114.3.1	GetRotation . . . . .	585
10.114.3.2	GetTranslation . . . . .	585
10.114.3.3	SetRotation . . . . .	585
10.114.3.4	SetTranslation . . . . .	585
10.114.4	Member Data Documentation . . . . .	586
10.114.4.1	rotate . . . . .	586
10.114.4.2	translate . . . . .	586
10.115	gazebo::rendering::Projector Class Reference . . . . .	586
10.115.1	Detailed Description . . . . .	586
10.115.2	Constructor & Destructor Documentation . . . . .	587
10.115.2.1	Projector . . . . .	587
10.115.2.2	~Projector . . . . .	587
10.115.3	Member Function Documentation . . . . .	587
10.115.3.1	GetParent . . . . .	587
10.115.3.2	Load . . . . .	587
10.115.3.3	Load . . . . .	587
10.115.3.4	Load . . . . .	587
10.115.3.5	SetEnabled . . . . .	588

10.115.3.6	SetTexture	588
10.115.3.7	Toggle	588
10.116	Gazebo::transport::Publication Class Reference	588
10.116.1	Detailed Description	589
10.116.2	Constructor & Destructor Documentation	589
10.116.2.1	Publication	589
10.116.2.2	~Publication	590
10.116.3	Member Function Documentation	590
10.116.3.1	AddPublisher	590
10.116.3.2	AddSubscription	590
10.116.3.3	AddSubscription	590
10.116.3.4	AddTransport	590
10.116.3.5	GetCallbackCount	590
10.116.3.6	GetLocallyAdvertised	591
10.116.3.7	GetMsgType	591
10.116.3.8	GetNodeCount	591
10.116.3.9	GetRemoteSubscriptionCount	591
10.116.3.10	GetTransportCount	591
10.116.3.11	HasTransport	591
10.116.3.12	LocalPublish	592
10.116.3.13	Publish	592
10.116.3.14	RemoveSubscription	592
10.116.3.15	RemoveSubscription	592
10.116.3.16	RemoveTransport	592
10.116.3.17	SetLocallyAdvertised	593
10.117	Gazebo::transport::PublicationTransport Class Reference	593
10.117.1	Detailed Description	593
10.117.2	Constructor & Destructor Documentation	594
10.117.2.1	PublicationTransport	594
10.117.2.2	~PublicationTransport	594
10.117.3	Member Function Documentation	594
10.117.3.1	AddCallback	594
10.117.3.2	Finis	594
10.117.3.3	GetConnection	594
10.117.3.4	GetMsgType	594
10.117.3.5	GetTopic	595
10.117.3.6	Init	595

10.118.1	gazebo::transport::Publisher Class Reference . . . . .	595
10.118.1	Detailed Description . . . . .	596
10.118.2	Constructor & Destructor Documentation . . . . .	596
10.118.2.1	Publisher . . . . .	596
10.118.2.2	Publisher . . . . .	596
10.118.2.3	~Publisher . . . . .	596
10.118.3	Member Function Documentation . . . . .	596
10.118.3.1	GetLatching . . . . .	596
10.118.3.2	GetMsgType . . . . .	596
10.118.3.3	GetOutgoingCount . . . . .	597
10.118.3.4	GetPrevMsg . . . . .	597
10.118.3.5	GetTopic . . . . .	597
10.118.3.6	HasConnections . . . . .	597
10.118.3.7	Publish . . . . .	597
10.118.3.8	Publish . . . . .	598
10.118.3.9	SendMessage . . . . .	598
10.118.3.10	SetPublication . . . . .	598
10.118.3.11	WaitForConnection . . . . .	598
10.119.1	gazebo::math::Quaternion Class Reference . . . . .	598
10.119.1	Detailed Description . . . . .	601
10.119.2	Constructor & Destructor Documentation . . . . .	601
10.119.2.1	Quaternion . . . . .	601
10.119.2.2	Quaternion . . . . .	601
10.119.2.3	Quaternion . . . . .	602
10.119.2.4	Quaternion . . . . .	602
10.119.2.5	Quaternion . . . . .	602
10.119.2.6	Quaternion . . . . .	602
10.119.2.7	~Quaternion . . . . .	602
10.119.3	Member Function Documentation . . . . .	602
10.119.3.1	Correct . . . . .	602
10.119.3.2	Dot . . . . .	603
10.119.3.3	EulerToQuaternion . . . . .	603
10.119.3.4	EulerToQuaternion . . . . .	603
10.119.3.5	GetAsAxis . . . . .	603
10.119.3.6	GetAsEuler . . . . .	603
10.119.3.7	GetAsMatrix3 . . . . .	604
10.119.3.8	GetAsMatrix4 . . . . .	604

10.119.3.9	GetExp	604
10.119.3.10	GetInverse	604
10.119.3.11	GetLog	604
10.119.3.12	GetPitch	604
10.119.3.13	GetRoll	605
10.119.3.14	GetXAxis	605
10.119.3.15	GetYaw	605
10.119.3.16	GetYAxis	605
10.119.3.17	GetZAxis	605
10.119.3.18	Invert	605
10.119.3.19	IsFinite	605
10.119.3.20	Normalize	606
10.119.3.21	operator!=	606
10.119.3.22	operator*	606
10.119.3.23	operator*	606
10.119.3.24	operator*	606
10.119.3.25	operator*= operator+	607
10.119.3.26	operator+ operator+=	607
10.119.3.27	operator+= operator-	607
10.119.3.28	operator- operator-	608
10.119.3.29	operator- operator-=	608
10.119.3.30	operator-= operator=	608
10.119.3.31	operator= operator==	608
10.119.3.32	operator== RotateVector	608
10.119.3.33	RotateVector RotateVectorReverse	609
10.119.3.34	Round	609
10.119.3.35	Scale	609
10.119.3.36	Set	609
10.119.3.37	SetFromAxis	609
10.119.3.38	SetFromAxis	610
10.119.3.39	SetFromEuler	610
10.119.3.40	SetFromEuler	610
10.119.3.41	SetToIdentity	610
10.119.3.42	Slerp	610
10.119.3.43	Squad	611
10.119.4	Friends And Related Function Documentation	611



10.119.4.1operator<<	611
10.119.4.2operator>>	611
10.119.5Member Data Documentation	611
10.119.5.1w	611
10.119.5.2x	612
10.119.5.3y	612
10.119.5.4z	612
10.120gazebo::math::Rand Class Reference	612
10.120.1Detailed Description	612
10.120.2Member Function Documentation	613
10.120.2.1GetDbfNormal	613
10.120.2.2GetDbfUniform	613
10.120.2.3GetIntNormal	613
10.120.2.4GetIntUniform	613
10.120.2.5GetSeed	613
10.120.2.6SetSeed	613
10.121gazebo::transport::RawCallbackHelper Class Reference	614
10.121.1Detailed Description	614
10.121.2Constructor & Destructor Documentation	615
10.121.2.1RawCallbackHelper	615
10.121.3Member Function Documentation	615
10.121.3.1GetMsgType	615
10.121.3.2HandleData	615
10.121.3.3IsLocal	615
10.122gazebo::sensors::RaySensor Class Reference	616
10.122.1Detailed Description	617
10.122.2Constructor & Destructor Documentation	617
10.122.2.1RaySensor	617
10.122.2.2~RaySensor	618
10.122.3Member Function Documentation	618
10.122.3.1Fini	618
10.122.3.2GetAngleMax	618
10.122.3.3GetAngleMin	618
10.122.3.4GetAngleResolution	618
10.122.3.5GetFiducial	618
10.122.3.6GetLaserShape	619
10.122.3.7GetRange	619

10.122.3.8	GetRangeCount	619
10.122.3.9	GetRangeMax	619
10.122.3.10	GetRangeMin	620
10.122.3.11	GetRangeResolution	620
10.122.3.12	GetRanges	620
10.122.3.13	GetRayCount	620
10.122.3.14	GetRetro	620
10.122.3.15	GetTopic	621
10.122.3.16	GetVerticalAngleMax	621
10.122.3.17	GetVerticalAngleMin	621
10.122.3.18	GetVerticalRangeCount	621
10.122.3.19	GetVerticalRayCount	621
10.122.3.20	Init	621
10.122.3.21	IsActive	622
10.122.3.22	Load	622
10.122.3.23	UpdateImpl	622
10.123	<b>Gazebo::physics::RayShape Class Reference</b>	622
10.123.1	Detailed Description	624
10.123.2	Constructor & Destructor Documentation	624
10.123.2.1	RayShape	624
10.123.2.2	RayShape	625
10.123.2.3	~RayShape	625
10.123.3	Member Function Documentation	625
10.123.3.1	FillMsg	625
10.123.3.2	GetFiducial	625
10.123.3.3	GetGlobalPoints	625
10.123.3.4	GetIntersection	625
10.123.3.5	GetLength	626
10.123.3.6	GetRelativePoints	626
10.123.3.7	GetRetro	626
10.123.3.8	Init	626
10.123.3.9	ProcessMsg	626
10.123.3.10	SetFiducial	627
10.123.3.11	SetLength	627
10.123.3.12	SetPoints	627
10.123.3.13	SetRetro	627
10.123.3.14	Update	627

10.123.4	Member Data Documentation . . . . .	627
10.123.4.1	contactFiducial . . . . .	627
10.123.4.2	contactLen . . . . .	628
10.123.4.3	contactRetro . . . . .	628
10.123.4.4	globalEndPos . . . . .	628
10.123.4.5	globalStartPos . . . . .	628
10.123.4.6	relativeEndPos . . . . .	628
10.123.4.7	relativeStartPos . . . . .	628
10.124	gazebo::rendering::RenderEngine Class Reference . . . . .	628
10.124.1	Detailed Description . . . . .	630
10.124.2	Member Enumeration Documentation . . . . .	630
10.124.2.1	RenderPathType . . . . .	630
10.124.3	Member Function Documentation . . . . .	630
10.124.3.1	AddResourcePath . . . . .	630
10.124.3.2	CreateScene . . . . .	631
10.124.3.3	Finis . . . . .	631
10.124.3.4	GetRenderPathType . . . . .	631
10.124.3.5	GetScene . . . . .	631
10.124.3.6	GetScene . . . . .	631
10.124.3.7	GetSceneCount . . . . .	632
10.124.3.8	Init . . . . .	632
10.124.3.9	Load . . . . .	632
10.124.3.10	RemoveScene . . . . .	632
10.124.4	Member Data Documentation . . . . .	632
10.124.4.1	dummyContext . . . . .	632
10.124.4.2	dummyDisplay . . . . .	632
10.124.4.3	dummyWindowId . . . . .	632
10.124.4.4	root . . . . .	632
10.125	gazebo::sensors::RFIDSensor Class Reference . . . . .	633
10.125.1	Detailed Description . . . . .	634
10.125.2	Constructor & Destructor Documentation . . . . .	634
10.125.2.1	RFIDSensor . . . . .	634
10.125.2.2	~RFIDSensor . . . . .	634
10.125.3	Member Function Documentation . . . . .	634
10.125.3.1	AddTag . . . . .	634
10.125.3.2	Finis . . . . .	634
10.125.3.3	Init . . . . .	634

10.125.3.4	Load	634
10.125.3.5	Load	634
10.125.3.6	UpdateImpl	635
10.126	Gazebo::sensors::RFIDTag Class Reference	635
10.126.1	Detailed Description	636
10.126.2	Constructor & Destructor Documentation	636
10.126.2.1	RFIDTag	636
10.126.2.2	~RFIDTag	636
10.126.3	Member Function Documentation	636
10.126.3.1	Finis	636
10.126.3.2	GetTagPose	637
10.126.3.3	Init	637
10.126.3.4	Load	637
10.126.3.5	Load	637
10.126.3.6	UpdateImpl	637
10.127	Gazebo::rendering::RFIDTagVisual Class Reference	637
10.127.1	Detailed Description	638
10.127.2	Constructor & Destructor Documentation	638
10.127.2.1	RFIDTagVisual	638
10.127.2.2	~RFIDTagVisual	639
10.128	Gazebo::rendering::RFIDVisual Class Reference	639
10.128.1	Detailed Description	640
10.128.2	Constructor & Destructor Documentation	640
10.128.2.1	RFIDVisual	640
10.128.2.2	~RFIDVisual	640
10.129	Road Class Reference	640
10.129.1	Detailed Description	640
10.130	Gazebo::physics::Road Class Reference	640
10.130.1	Detailed Description	641
10.130.2	Constructor & Destructor Documentation	641
10.130.2.1	Road	641
10.130.2.2	~Road	642
10.130.3	Member Function Documentation	642
10.130.3.1	Init	642
10.130.3.2	Load	642
10.131	Gazebo::rendering::Road2d Class Reference	642
10.131.1	Constructor & Destructor Documentation	642

10.131.1.1	Road2d . . . . .	642
10.131.1.2	~Road2d . . . . .	642
10.131.2	Member Function Documentation . . . . .	643
10.131.2.1	Load . . . . .	643
10.132	gazebo::math::RotationSpline Class Reference . . . . .	643
10.132.1	Detailed Description . . . . .	644
10.132.2	Constructor & Destructor Documentation . . . . .	644
10.132.2.1	RotationSpline . . . . .	644
10.132.2.2	~RotationSpline . . . . .	644
10.132.3	Member Function Documentation . . . . .	644
10.132.3.1	AddPoint . . . . .	644
10.132.3.2	Clear . . . . .	644
10.132.3.3	GetNumPoints . . . . .	644
10.132.3.4	GetPoint . . . . .	644
10.132.3.5	Interpolate . . . . .	645
10.132.3.6	Interpolate . . . . .	645
10.132.3.7	RecalcTangents . . . . .	645
10.132.3.8	SetAutoCalculate . . . . .	646
10.132.3.9	UpdatePoint . . . . .	646
10.132.4	Member Data Documentation . . . . .	646
10.132.4.1	autoCalc . . . . .	646
10.132.4.2	points . . . . .	646
10.132.4.3	tangents . . . . .	646
10.133	gazebo::rendering::RTShaderSystem Class Reference . . . . .	647
10.133.1	Detailed Description . . . . .	648
10.133.2	Member Enumeration Documentation . . . . .	648
10.133.2.1	LightingModel . . . . .	648
10.133.3	Member Function Documentation . . . . .	648
10.133.3.1	AddScene . . . . .	648
10.133.3.2	ApplyShadows . . . . .	649
10.133.3.3	AttachEntity . . . . .	649
10.133.3.4	AttachViewport . . . . .	649
10.133.3.5	Clear . . . . .	649
10.133.3.6	DetachEntity . . . . .	649
10.133.3.7	DetachViewport . . . . .	649
10.133.3.8	Finis . . . . .	650
10.133.3.9	GenerateShaders . . . . .	650

10.133.3.10	GetPSSMShadowCameraSetup . . . . .	650
10.133.3.11	hit . . . . .	650
10.133.3.12	RemoveScene . . . . .	650
10.133.3.13	RemoveShadows . . . . .	650
10.133.3.13	SetPerPixelLighting . . . . .	651
10.133.3.15	UpdateShaders . . . . .	651
10.134	Gazebo::rendering::Scene Class Reference . . . . .	651
10.134.1	Detailed Description . . . . .	654
10.134.2	Constructor & Destructor Documentation . . . . .	654
10.134.2.1	Scene . . . . .	654
10.134.2.2	~Scene . . . . .	654
10.134.3	Member Function Documentation . . . . .	654
10.134.3.1	AddVisual . . . . .	655
10.134.3.2	Clear . . . . .	655
10.134.3.3	CloneVisual . . . . .	655
10.134.3.4	CreateCamera . . . . .	655
10.134.3.5	CreateDepthCamera . . . . .	655
10.134.3.6	CreateGrid . . . . .	656
10.134.3.7	CreateUserCamera . . . . .	656
10.134.3.8	DrawLine . . . . .	656
10.134.3.9	GetAmbientColor . . . . .	656
10.134.3.10	GetBackgroundColor . . . . .	657
10.134.3.10	GetCamera . . . . .	657
10.134.3.10	GetCamera . . . . .	657
10.134.3.10	GetCameraCount . . . . .	657
10.134.3.10	GetFirstContact . . . . .	658
10.134.3.10	GetGrid . . . . .	658
10.134.3.10	GetGridCount . . . . .	658
10.134.3.10	GetHeightBelowPoint . . . . .	658
10.134.3.10	GetHeightmap . . . . .	658
10.134.3.10	GetId . . . . .	659
10.134.3.20	GetIdString . . . . .	659
10.134.3.20	GetLight . . . . .	659
10.134.3.20	GetLight . . . . .	659
10.134.3.23	GetLightCount . . . . .	659
10.134.3.24	GetManager . . . . .	660
10.134.3.25	SetModelVisualAt . . . . .	660

10.134.3.26	GetName . . . . .	660
10.134.3.27	GetSelectedVisual . . . . .	660
10.134.3.28	GetShadowsEnabled . . . . .	660
10.134.3.29	GetUserCamera . . . . .	661
10.134.3.30	GetUserCameraCount . . . . .	661
10.134.3.31	GetVisual . . . . .	661
10.134.3.32	GetVisualAt . . . . .	661
10.134.3.33	GetVisualAt . . . . .	661
10.134.3.34	GetVisualBelow . . . . .	662
10.134.3.35	GetVisualsBelowPoint . . . . .	662
10.134.3.36	GetWorldVisual . . . . .	662
10.134.3.37	Init . . . . .	662
10.134.3.38	Load . . . . .	662
10.134.3.39	Load . . . . .	663
10.134.3.40	PreRender . . . . .	663
10.134.3.41	PrintSceneGraph . . . . .	663
10.134.3.42	RemoveVisual . . . . .	663
10.134.3.43	SelectVisual . . . . .	663
10.134.3.44	SetAmbientColor . . . . .	663
10.134.3.45	SetBackgroundColor . . . . .	663
10.134.3.46	SetFog . . . . .	664
10.134.3.47	SetGrid . . . . .	664
10.134.3.48	SetShadowsEnabled . . . . .	664
10.134.3.49	SetTransparent . . . . .	664
10.134.3.50	SetVisible . . . . .	664
10.134.3.51	ShowCollisions . . . . .	665
10.134.3.52	ShowCOMs . . . . .	665
10.134.3.53	ShowContacts . . . . .	665
10.134.3.54	ShowJoints . . . . .	665
10.134.3.55	SnapVisualToNearestBelow . . . . .	665
10.134.3.56	StripSceneName . . . . .	665
10.134.4	Member Data Documentation . . . . .	666
10.134.4.1	skyx . . . . .	666
10.135	gazebo::physics::ScrewJoint< T > Class Template Reference . . . . .	666
10.135.1	Detailed Description . . . . .	667
10.135.2	Constructor & Destructor Documentation . . . . .	667
10.135.2.1	ScrewJoint . . . . .	667

10.135.2.2~ScrewJoint . . . . .	667
10.135.3 Member Function Documentation . . . . .	667
10.135.3.1GetAnchor . . . . .	667
10.135.3.2GetAngleCount . . . . .	668
10.135.3.3Load . . . . .	668
10.135.3.4SetAnchor . . . . .	668
10.135.3.5SetThreadPitch . . . . .	668
10.135.4 Member Data Documentation . . . . .	668
10.135.4.1fakeAnchor . . . . .	668
10.135.4.2threadPitch . . . . .	668
10.136df::SDF Class Reference . . . . .	669
10.136.1Detailed Description . . . . .	669
10.136.2Constructor & Destructor Documentation . . . . .	669
10.136.2.1SDF . . . . .	669
10.136.3Member Function Documentation . . . . .	669
10.136.3.1PrintDescription . . . . .	669
10.136.3.2PrintDoc . . . . .	669
10.136.3.3PrintValues . . . . .	669
10.136.3.4PrintWiki . . . . .	669
10.136.3.5SetFromString . . . . .	669
10.136.3.6ToString . . . . .	670
10.136.3.7Write . . . . .	670
10.136.4Member Data Documentation . . . . .	670
10.136.4.1root . . . . .	670
10.136.4.2version . . . . .	670
10.137gazebo::rendering::SelectionObj Class Reference . . . . .	670
10.137.1Detailed Description . . . . .	670
10.137.2Constructor & Destructor Documentation . . . . .	671
10.137.2.1SelectionObj . . . . .	671
10.137.2.2~SelectionObj . . . . .	671
10.137.3Member Function Documentation . . . . .	671
10.137.3.1Attach . . . . .	671
10.137.3.2Clear . . . . .	671
10.137.3.3GetVisualName . . . . .	671
10.137.3.4Init . . . . .	671
10.137.3.5IsActive . . . . .	671
10.137.3.6SetActive . . . . .	672



10.137.3.7	SetHighlight	672
10.138	gazebo::sensors::Sensor Class Reference	672
10.138.1	Detailed Description	674
10.138.2	Constructor & Destructor Documentation	675
10.138.2.1	Sensor	675
10.138.2.2	~Sensor	675
10.138.3	Member Function Documentation	675
10.138.3.1	ConnectUpdated	675
10.138.3.2	DisconnectUpdated	675
10.138.3.3	FillMsg	675
10.138.3.4	Fini	676
10.138.3.5	GetLastMeasurementTime	676
10.138.3.6	GetLastUpdateTime	676
10.138.3.7	GetName	676
10.138.3.8	GetParentName	676
10.138.3.9	GetPose	677
10.138.3.10	GetScopedName	677
10.138.3.11	GetTopic	677
10.138.3.12	GetType	677
10.138.3.13	GetUpdateRate	677
10.138.3.14	GetVisualize	677
10.138.3.15	GetWorldName	678
10.138.3.16	hit	678
10.138.3.17	IsActive	678
10.138.3.18	load	678
10.138.3.19	load	678
10.138.3.20	SetActive	679
10.138.3.21	SetParent	679
10.138.3.22	SetUpdateRate	679
10.138.3.23	update	679
10.138.3.24	updateImpl	679
10.138.4	Member Data Documentation	680
10.138.4.1	active	680
10.138.4.2	connections	680
10.138.4.3	lastMeasurementTime	680
10.138.4.4	lastUpdateTime	680
10.138.4.5	node	680

10.138.4.6	parentName	680
10.138.4.7	plugins	680
10.138.4.8	pose	680
10.138.4.9	poseSub	680
10.138.4.10	self	681
10.138.4.11	updatePeriod	681
10.138.4.12	world	681
10.139	SensorFactor Class Reference	681
10.139.1	Detailed Description	681
10.140	Gazebo::sensors::SensorFactory Class Reference	681
10.140.1	Member Function Documentation	682
10.140.1.1	GetSensorTypes	682
10.140.1.2	NewSensor	682
10.140.1.3	RegisterAll	682
10.140.1.4	RegisterSensor	682
10.141	Gazebo::sensors::SensorManager Class Reference	683
10.141.1	Detailed Description	684
10.141.2	Member Function Documentation	684
10.141.2.1	CreateSensor	684
10.141.2.2	Fin	684
10.141.2.3	GetSensor	684
10.141.2.4	GetSensors	685
10.141.2.5	GetSensorTypes	685
10.141.2.6	Init	685
10.141.2.7	RemoveSensor	685
10.141.2.8	RemoveSensors	685
10.141.2.9	Run	685
10.141.2.10	SensorsInitialized	685
10.141.2.11	Stop	686
10.141.2.12	Update	686
10.142	Gazebo::SensorPlugin Class Reference	686
10.142.1	Detailed Description	687
10.142.2	Constructor & Destructor Documentation	687
10.142.2.1	SensorPlugin	687
10.142.2.2	~SensorPlugin	687
10.142.3	Member Function Documentation	687
10.142.3.1	Init	687

10.142.3.2	Load . . . . .	687
10.142.3.3	Reset . . . . .	687
10.143	Gazebo::Server Class Reference . . . . .	688
10.143.1	Constructor & Destructor Documentation . . . . .	688
10.143.1.1	Server . . . . .	688
10.143.1.2	~Server . . . . .	688
10.143.2	Member Function Documentation . . . . .	688
10.143.2.1	Finis . . . . .	688
10.143.2.2	GetInitialized . . . . .	688
10.143.2.3	Init . . . . .	688
10.143.2.4	LoadFile . . . . .	688
10.143.2.5	LoadString . . . . .	688
10.143.2.6	ParseArgs . . . . .	688
10.143.2.7	PrintUsage . . . . .	688
10.143.2.8	Run . . . . .	688
10.143.2.9	SetParams . . . . .	688
10.143.2.10	Stop . . . . .	689
10.143.3	Member Data Documentation . . . . .	689
10.143.3.1	systemPluginsArgc . . . . .	689
10.143.3.2	systemPluginsArgv . . . . .	689
10.144	Gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg Class Reference . . . . .	689
10.144.1	Detailed Description . . . . .	690
10.144.2	Member Function Documentation . . . . .	690
10.144.2.1	defaultVpParams . . . . .	690
10.144.2.2	generateFragmentProgram . . . . .	690
10.144.2.3	generateVertexProgram . . . . .	690
10.144.2.4	generateVertexProgramSource . . . . .	690
10.144.2.5	generateVpDynamicShadows . . . . .	690
10.144.2.6	generateVpDynamicShadowsParams . . . . .	690
10.144.2.7	generateVpFooter . . . . .	690
10.144.2.8	generateVpHeader . . . . .	690
10.145	Gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL Class Reference . . . . .	691
10.145.1	Detailed Description . . . . .	692
10.145.2	Member Function Documentation . . . . .	692
10.145.2.1	defaultVpParams . . . . .	692
10.145.2.2	generateFpDynamicShadows . . . . .	692
10.145.2.3	generateFpDynamicShadowsHelpers . . . . .	692

10.145.2.4	generateFpDynamicShadowsParams	692
10.145.2.5	generateFpFooter	692
10.145.2.6	generateFpHeader	692
10.145.2.7	generateFpLayer	693
10.145.2.8	generateFragmentProgram	693
10.145.2.9	generateFragmentProgramSource	693
10.145.2.10	generateVertexProgram	693
10.145.2.11	generateVertexProgramSource	693
10.145.2.12	generateVpDynamicShadows	693
10.145.2.13	generateVpDynamicShadowsParams	693
10.145.2.14	generateVpFooter	693
10.145.2.15	generateVpHeader	693
10.145.2.16	updateParams	693
10.145.2.17	updateVpParams	693
10.146	gazebo::physics::Shape Class Reference	693
10.146.1	Detailed Description	694
10.146.2	Constructor & Destructor Documentation	694
10.146.2.1	Shape	695
10.146.2.2	~Shape	695
10.146.3	Member Function Documentation	695
10.146.3.1	FillMsg	695
10.146.3.2	Init	695
10.146.3.3	ProcessMsg	695
10.146.4	Member Data Documentation	696
10.146.4.1	collisionParent	696
10.147	Singleton< T > Class Template Reference	696
10.147.1	Detailed Description	698
10.147.2	Constructor & Destructor Documentation	698
10.147.2.1	SingletonT	698
10.147.2.2	~SingletonT	698
10.147.3	Member Function Documentation	698
10.147.3.1	Instance	698
10.148	gazebo::common::Skeleton Class Reference	698
10.148.1	Detailed Description	700
10.148.2	Constructor & Destructor Documentation	700
10.148.2.1	Skeleton	700
10.148.2.2	~Skeleton	700

10.148.2.3~Skeleton	700
10.148.3 Member Function Documentation	700
10.148.3.1AddAnimation	700
10.148.3.2AddVertNodeWeight	701
10.148.3.3BuildNodeMap	701
10.148.3.4GetAnimation	701
10.148.3.5GetBindShapeTransform	701
10.148.3.6GetNodeByHandle	701
10.148.3.7GetNodeById	701
10.148.3.8GetNodeByName	702
10.148.3.9GetNodes	702
10.148.3.10GetNumAnimations	702
10.148.3.11GetNumJoints	702
10.148.3.12GetNumNodes	702
10.148.3.13GetNumVertNodeWeights	703
10.148.3.14GetRootNode	703
10.148.3.15GetVertNodeWeight	703
10.148.3.16PrintTransforms	703
10.148.3.17Scale	703
10.148.3.18SetBindShapeTransform	703
10.148.3.19SetNumVertAttached	704
10.148.3.20SetRootNode	704
10.148.4 Member Data Documentation	704
10.148.4.1animations	704
10.148.4.2bindShapeTransform	704
10.148.4.3nodes	704
10.148.4.4rootNW	704
10.148.4.5root	704
10.149 gazebo::common::SkeletonAnimation Class Reference	705
10.149.1 Detailed Description	706
10.149.2 Constructor & Destructor Documentation	706
10.149.2.1SkeletonAnimation	706
10.149.2.2~SkeletonAnimation	706
10.149.3 Member Function Documentation	706
10.149.3.1AddKeyFrame	706
10.149.3.2AddKeyFrame	706
10.149.3.3GetLength	706

10.149.3.4	GetName . . . . .	707
10.149.3.5	GetNodeCount . . . . .	707
10.149.3.6	GetNodePoseAt . . . . .	707
10.149.3.7	GetPoseAt . . . . .	707
10.149.3.8	GetPoseAtX . . . . .	708
10.149.3.9	HasNode . . . . .	708
10.149.3.10	Scale . . . . .	708
10.149.3.11	SetName . . . . .	708
10.149.4	Member Data Documentation . . . . .	708
10.149.4.1	animations . . . . .	708
10.149.4.2	length . . . . .	709
10.149.4.3	name . . . . .	709
10.150	Gazebo::common::SkeletonNode Class Reference . . . . .	709
10.150.1	Detailed Description . . . . .	711
10.150.2	Member Enumeration Documentation . . . . .	711
10.150.2.1	SkeletonNodeType . . . . .	711
10.150.3	Constructor & Destructor Documentation . . . . .	711
10.150.3.1	SkeletonNode . . . . .	711
10.150.3.2	SkeletonNode . . . . .	712
10.150.3.3	~SkeletonNode . . . . .	712
10.150.4	Member Function Documentation . . . . .	712
10.150.4.1	AddChild . . . . .	712
10.150.4.2	AddRawTransform . . . . .	712
10.150.4.3	GetChild . . . . .	712
10.150.4.4	GetChildById . . . . .	712
10.150.4.5	GetChildByName . . . . .	713
10.150.4.6	GetChildCount . . . . .	713
10.150.4.7	GetHandle . . . . .	713
10.150.4.8	GetId . . . . .	713
10.150.4.9	GetInverseBindTransform . . . . .	713
10.150.4.10	GetModelTransform . . . . .	714
10.150.4.11	GetName . . . . .	714
10.150.4.12	GetNumRawTrans . . . . .	714
10.150.4.13	GetParent . . . . .	714
10.150.4.14	GetRawTransform . . . . .	714
10.150.4.15	GetRawTransforms . . . . .	714
10.150.4.16	GetTransform . . . . .	715

10.150.4.1	GetTransforms . . . . .	715
10.150.4.18	Joint . . . . .	715
10.150.4.19	RootNode . . . . .	715
10.150.4.20	Reset . . . . .	715
10.150.4.23	SetHandle . . . . .	715
10.150.4.24	SetId . . . . .	716
10.150.4.25	SetInitialTransform . . . . .	716
10.150.4.26	SetInverseBindTransform . . . . .	716
10.150.4.25	SetModelTransform . . . . .	716
10.150.4.26	SetName . . . . .	716
10.150.4.27	SetParent . . . . .	716
10.150.4.28	SetTransform . . . . .	717
10.150.4.29	SetType . . . . .	717
10.150.4.30	UpdateChildrenTransforms . . . . .	717
10.150.5	Member Data Documentation . . . . .	717
10.150.5.1	children . . . . .	717
10.150.5.2	handle . . . . .	717
10.150.5.3	id . . . . .	717
10.150.5.4	initialTransform . . . . .	717
10.150.5.5	invBindTransform . . . . .	717
10.150.5.6	modelTransform . . . . .	717
10.150.5.7	name . . . . .	718
10.150.5.8	parent . . . . .	718
10.150.5.9	rawTransforms . . . . .	718
10.150.5.10	transform . . . . .	718
10.150.5.11	type . . . . .	718
10.151	gazebo::physics::SliderJoint< T > Class Template Reference . . . . .	718
10.151.1	Detailed Description . . . . .	719
10.151.2	Constructor & Destructor Documentation . . . . .	719
10.151.2.1	SliderJoint . . . . .	719
10.151.2.2	~SliderJoint . . . . .	719
10.151.3	Member Function Documentation . . . . .	719
10.151.3.1	GetAnchor . . . . .	719
10.151.3.2	GetAngleCount . . . . .	720
10.151.3.3	Load . . . . .	720
10.151.3.4	SetAnchor . . . . .	720
10.151.4	Member Data Documentation . . . . .	720

10.151.4.1fakeAnchor . . . . .	720
10.152 gazebo::rendering::GzTerrainMatGen::SM2Profile Class Reference . . . . .	720
10.152.1 Detailed Description . . . . .	721
10.152.2 Constructor & Destructor Documentation . . . . .	722
10.152.2.1 SM2Profile . . . . .	722
10.152.2.2 ~SM2Profile . . . . .	722
10.152.3 Member Function Documentation . . . . .	722
10.152.3.1 addTechnique . . . . .	722
10.152.3.2 generate . . . . .	722
10.152.3.3 generateForCompositeMap . . . . .	722
10.152.3.4 updateParams . . . . .	722
10.152.3.5 updateParamsForCompositeMap . . . . .	722
10.153 gazebo::physics::SphereShape Class Reference . . . . .	722
10.153.1 Detailed Description . . . . .	723
10.153.2 Constructor & Destructor Documentation . . . . .	724
10.153.2.1 SphereShape . . . . .	724
10.153.2.2 ~SphereShape . . . . .	724
10.153.3 Member Function Documentation . . . . .	724
10.153.3.1 fillMsg . . . . .	724
10.153.3.2 getRadius . . . . .	724
10.153.3.3 init . . . . .	724
10.153.3.4 processMsg . . . . .	724
10.153.3.5 setRadius . . . . .	725
10.154 gazebo::math::Spline Class Reference . . . . .	725
10.154.1 Detailed Description . . . . .	726
10.154.2 Constructor & Destructor Documentation . . . . .	726
10.154.2.1 Spline . . . . .	726
10.154.2.2 ~Spline . . . . .	726
10.154.3 Member Function Documentation . . . . .	726
10.154.3.1 addPoint . . . . .	726
10.154.3.2 clear . . . . .	726
10.154.3.3 getPoint . . . . .	726
10.154.3.4 getPointCount . . . . .	727
10.154.3.5 getTangent . . . . .	727
10.154.3.6 getTension . . . . .	727
10.154.3.7 interpolate . . . . .	727
10.154.3.8 interpolate . . . . .	727



10.154.3.9	RecalcTangents . . . . .	728
10.154.3.10	SetAutoCalculate . . . . .	728
10.154.3.11	SetTension . . . . .	728
10.154.3.12	UpdatePoint . . . . .	728
10.154.4	Member Data Documentation . . . . .	729
10.154.4.1	autoCalc . . . . .	729
10.154.4.2	coeffs . . . . .	729
10.154.4.3	points . . . . .	729
10.154.4.4	tangents . . . . .	729
10.154.4.5	tension . . . . .	729
10.155	gazebo::physics::State Class Reference . . . . .	729
10.155.1	Detailed Description . . . . .	731
10.155.2	Constructor & Destructor Documentation . . . . .	731
10.155.2.1	State . . . . .	731
10.155.2.2	State . . . . .	731
10.155.2.3	~State . . . . .	731
10.155.3	Member Function Documentation . . . . .	731
10.155.3.1	GetName . . . . .	731
10.155.3.2	GetRealTime . . . . .	732
10.155.3.3	GetSimTime . . . . .	732
10.155.3.4	GetWallTime . . . . .	732
10.155.3.5	Load . . . . .	732
10.155.3.6	operator- . . . . .	732
10.155.3.7	operator= . . . . .	733
10.155.3.8	SetName . . . . .	733
10.155.4	Member Data Documentation . . . . .	733
10.155.4.1	name . . . . .	733
10.155.4.2	realTime . . . . .	733
10.155.4.3	simTime . . . . .	733
10.155.4.4	wallTime . . . . .	733
10.156	gazebo::common::STLLoader Class Reference . . . . .	733
10.156.1	Detailed Description . . . . .	734
10.156.2	Constructor & Destructor Documentation . . . . .	734
10.156.2.1	STLLoader . . . . .	734
10.156.2.2	~STLLoader . . . . .	734
10.156.3	Member Function Documentation . . . . .	734
10.156.3.1	Load . . . . .	734

10.157.0	Gazebo::common::SubMesh Class Reference . . . . .	735
10.157.1	Detailed Description . . . . .	737
10.157.2	Member Enumeration Documentation . . . . .	737
10.157.2.1	PrimitiveType . . . . .	737
10.157.3	Constructor & Destructor Documentation . . . . .	737
10.157.3.1	SubMesh . . . . .	737
10.157.3.2	~SubMesh . . . . .	737
10.157.4	Member Function Documentation . . . . .	737
10.157.4.1	AddIndex . . . . .	737
10.157.4.2	AddNodeAssignment . . . . .	738
10.157.4.3	AddNormal . . . . .	738
10.157.4.4	AddNormal . . . . .	738
10.157.4.5	AddTexCoord . . . . .	738
10.157.4.6	AddVertex . . . . .	738
10.157.4.7	AddVertex . . . . .	739
10.157.4.8	CopyNormals . . . . .	739
10.157.4.9	CopyVertices . . . . .	739
10.157.4.10	FillArrays . . . . .	739
10.157.4.11	GenSphericalTexCoord . . . . .	739
10.157.4.12	GetIndex . . . . .	739
10.157.4.13	GetIndexCount . . . . .	740
10.157.4.14	GetMaterialIndex . . . . .	740
10.157.4.15	GetMax . . . . .	740
10.157.4.16	GetMaxIndex . . . . .	740
10.157.4.17	GetMin . . . . .	740
10.157.4.18	GetNodeAssignment . . . . .	740
10.157.4.19	GetNodeAssignmentsCount . . . . .	740
10.157.4.20	GetNormal . . . . .	740
10.157.4.21	GetNormalCount . . . . .	741
10.157.4.22	GetPrimitiveType . . . . .	741
10.157.4.23	GetTexCoord . . . . .	741
10.157.4.24	GetTexCoordCount . . . . .	741
10.157.4.25	GetVertex . . . . .	741
10.157.4.26	GetVertexCount . . . . .	742
10.157.4.27	GetVertexIndex . . . . .	742
10.157.4.28	HasVertex . . . . .	742
10.157.4.29	RecalculateNormals . . . . .	742

10.157.4.30	Scale . . . . .	742
10.157.4.31	SetIndexCount . . . . .	742
10.157.4.32	SetMaterialIndex . . . . .	742
10.157.4.33	SetNormal . . . . .	743
10.157.4.34	SetNormalCount . . . . .	743
10.157.4.35	SetPrimitiveType . . . . .	743
10.157.4.36	SetScale . . . . .	743
10.157.4.37	SetSubMeshCenter . . . . .	743
10.157.4.38	SetTexCoord . . . . .	743
10.157.4.39	SetTexCoordCount . . . . .	744
10.157.4.40	SetVertex . . . . .	744
10.157.4.41	SetVertexCount . . . . .	744
10.158	gazebo::transport::SubscribeOptions Class Reference . . . . .	744
10.158.1	Detailed Description . . . . .	745
10.158.2	Constructor & Destructor Documentation . . . . .	745
10.158.2.1	SubscribeOptions . . . . .	745
10.158.3	Member Function Documentation . . . . .	745
10.158.3.1	GetLatching . . . . .	745
10.158.3.2	GetMsgType . . . . .	745
10.158.3.3	GetNode . . . . .	745
10.158.3.4	GetTopic . . . . .	746
10.158.3.5	init . . . . .	746
10.158.3.6	init . . . . .	746
10.159	gazebo::transport::Subscriber Class Reference . . . . .	746
10.159.1	Detailed Description . . . . .	747
10.159.2	Constructor & Destructor Documentation . . . . .	747
10.159.2.1	Subscriber . . . . .	747
10.159.2.2	~Subscriber . . . . .	747
10.159.3	Member Function Documentation . . . . .	747
10.159.3.1	GetCallbackId . . . . .	747
10.159.3.2	GetTopic . . . . .	747
10.159.3.3	SetCallbackId . . . . .	747
10.159.3.4	Unsubscribe . . . . .	747
10.160	gazebo::transport::SubscriptionTransport Class Reference . . . . .	748
10.160.1	Detailed Description . . . . .	748
10.160.2	Constructor & Destructor Documentation . . . . .	749
10.160.2.1	SubscriptionTransport . . . . .	749

10.160.2.2~SubscriptionTransport . . . . .	749
10.160.3 Member Function Documentation . . . . .	749
10.160.3.1GetConnection . . . . .	749
10.160.3.2HandleData . . . . .	749
10.160.3.3Init . . . . .	749
10.160.3.4sLocal . . . . .	749
10.161 gazebo::physics::SurfaceParams Class Reference . . . . .	750
10.161.1 Detailed Description . . . . .	751
10.161.2 Constructor & Destructor Documentation . . . . .	751
10.161.2.1SurfaceParams . . . . .	751
10.161.2.2~SurfaceParams . . . . .	751
10.161.3 Member Function Documentation . . . . .	751
10.161.3.1FillMsg . . . . .	751
10.161.3.2Load . . . . .	751
10.161.3.3ProcessMsg . . . . .	751
10.161.4 Member Data Documentation . . . . .	751
10.161.4.1bounce . . . . .	752
10.161.4.2bounceThreshold . . . . .	752
10.161.4.3cfm . . . . .	752
10.161.4.4erp . . . . .	752
10.161.4.5dir1 . . . . .	752
10.161.4.6d . . . . .	752
10.161.4.7kp . . . . .	753
10.161.4.8maxVel . . . . .	753
10.161.4.9minDepth . . . . .	753
10.161.4.10u1 . . . . .	753
10.161.4.11u2 . . . . .	753
10.161.4.12ip1 . . . . .	754
10.161.4.13ip2 . . . . .	754
10.162 gazebo::common::SystemPaths Class Reference . . . . .	754
10.162.1 Detailed Description . . . . .	755
10.162.2 Member Function Documentation . . . . .	756
10.162.2.1AddGazeboPaths . . . . .	756
10.162.2.2AddOgrePaths . . . . .	756
10.162.2.3AddPluginPaths . . . . .	756
10.162.2.4AddSearchPathSuffix . . . . .	756
10.162.2.5ClearGazeboPaths . . . . .	756

10.162.2.6	ClearOgrePaths	756
10.162.2.7	ClearPluginPaths	756
10.162.2.8	FindFile	757
10.162.2.9	FindFileURI	757
10.162.2.10	GetGazeboPaths	757
10.162.2.10	GetLogPath	757
10.162.2.10	GetModelPaths	757
10.162.2.10	GetOgrePaths	758
10.162.2.10	GetPluginPaths	758
10.162.2.10	GetWorldPathExtension	758
10.162.3	Member Data Documentation	758
10.162.3.1	gazeboPathsFromEnv	758
10.162.3.2	modelPathsFromEnv	758
10.162.3.3	ogrePathsFromEnv	758
10.162.3.4	pluginPathsFromEnv	758
10.163	gazebo::SystemPlugin Class Reference	759
10.163.1	Detailed Description	759
10.163.2	Constructor & Destructor Documentation	759
10.163.2.1	SystemPlugin	759
10.163.2.2	~SystemPlugin	760
10.163.3	Member Function Documentation	760
10.163.3.1	Init	760
10.163.3.2	Load	760
10.163.3.3	Reset	760
10.164	gazebo::common::Time Class Reference	760
10.164.1	Detailed Description	764
10.164.2	Constructor & Destructor Documentation	764
10.164.2.1	Time	764
10.164.2.2	Time	764
10.164.2.3	Time	764
10.164.2.4	Time	764
10.164.2.5	Time	764
10.164.2.6	Time	765
10.164.2.7	~Time	765
10.164.3	Member Function Documentation	765
10.164.3.1	Double	765
10.164.3.2	Float	765

---

10.164.3.3	GetWallTime	765
10.164.3.4	MicToNano	765
10.164.3.5	MilToNano	766
10.164.3.6	MSleep	766
10.164.3.7	NSleep	766
10.164.3.8	NSleep	766
10.164.3.9	operator!=	766
10.164.3.10	operator!=	767
10.164.3.11	operator!=	767
10.164.3.12	operator!=	767
10.164.3.13	operator*	767
10.164.3.14	operator*	768
10.164.3.15	operator*	768
10.164.3.16	operator*= <del>operator*=</del>	768
10.164.3.17	operator*= <del>operator*=</del>	768
10.164.3.18	operator*= <del>operator*=</del>	769
10.164.3.19	operator+	769
10.164.3.20	operator+	769
10.164.3.21	operator+	769
10.164.3.22	operator+=	770
10.164.3.23	operator+=	770
10.164.3.24	operator+=	770
10.164.3.25	operator-	770
10.164.3.26	operator-	771
10.164.3.27	operator-	771
10.164.3.28	operator-=	771
10.164.3.29	operator-=	771
10.164.3.30	operator-=	772
10.164.3.31	operator/	772
10.164.3.32	operator/	772
10.164.3.33	operator/	772
10.164.3.34	operator/=	773
10.164.3.35	operator/=	773
10.164.3.36	operator/=	773
10.164.3.37	operator<	773
10.164.3.38	operator<	774
10.164.3.39	operator<	774

---

10.164.3.40	operator<	774
10.164.3.41	operator<=	774
10.164.3.42	operator<=	775
10.164.3.43	operator<=	775
10.164.3.44	operator<=	775
10.164.3.45	operator=	775
10.164.3.46	operator=	776
10.164.3.47	operator=	776
10.164.3.48	operator==	776
10.164.3.49	operator==	776
10.164.3.50	operator==	777
10.164.3.51	operator==	777
10.164.3.52	operator>	777
10.164.3.53	operator>	777
10.164.3.54	operator>	778
10.164.3.55	operator>	778
10.164.3.56	operator>=	778
10.164.3.57	operator>=	778
10.164.3.58	operator>=	779
10.164.3.59	operator>=	779
10.164.3.60	SecToNano	779
10.164.3.61	Set	779
10.164.3.62	Set	780
10.164.3.63	SetToWallTime	780
10.164.4	Friends And Related Function Documentation	780
10.164.4.1	operator<<	780
10.164.4.2	operator>>	780
10.164.5	Member Data Documentation	780
10.164.5.1	insec	780
10.164.5.2	sec	781
10.165	Gazebo::common::Timer Class Reference	781
10.165.1	Detailed Description	781
10.165.2	Constructor & Destructor Documentation	782
10.165.2.1	Timer	782
10.165.2.2	~Timer	782
10.165.3	Member Function Documentation	782
10.165.3.1	GetElapsed	782

10.165.3.2	Start	782
10.165.4	Friends And Related Function Documentation	782
10.165.4.1	operator<<	782
10.166	Gazebo::transport::TopicManager Class Reference	782
10.166.1	Detailed Description	784
10.166.2	Member Typedef Documentation	784
10.166.2.1	SubNodeMap	784
10.166.3	Member Function Documentation	784
10.166.3.1	AddNode	784
10.166.3.2	Advertise	785
10.166.3.3	ClearBuffers	785
10.166.3.4	ConnectPubToSub	785
10.166.3.5	ConnectSubscribers	785
10.166.3.6	ConnectSubToPub	785
10.166.3.7	DisconnectPubFromSub	786
10.166.3.8	DisconnectSubFromPub	786
10.166.3.9	FindPublication	786
10.166.3.10	Ini	786
10.166.3.11	GetAdvertisedTopics	786
10.166.3.12	GetTopicNamespaces	787
10.166.3.13	Init	787
10.166.3.14	IsAdvertised	787
10.166.3.15	PauseIncoming	787
10.166.3.16	ProcessNodes	787
10.166.3.17	Publish	787
10.166.3.18	RegisterTopicNamespace	788
10.166.3.19	RemoveNode	788
10.166.3.20	Subscribe	788
10.166.3.21	Unadvertise	788
10.166.3.22	Unsubscribe	788
10.166.3.23	UpdatePublications	789
10.167	Gazebo::physics::TrajectoryInfo Struct Reference	789
10.167.1	Member Data Documentation	789
10.167.1.1	duration	789
10.167.1.2	endTime	789
10.167.1.3	id	789
10.167.1.4	startTime	789



10.167.1.5	translated	789
10.167.1.6	type	790
10.168	gazebo::physics::TrimeshShape Class Reference	790
10.168.1	Detailed Description	791
10.168.2	Constructor & Destructor Documentation	791
10.168.2.1	TrimeshShape	791
10.168.2.2	~TrimeshShape	791
10.168.3	Member Function Documentation	791
10.168.3.1	FillMsg	791
10.168.3.2	GetFilename	792
10.168.3.3	GetSize	792
10.168.3.4	Init	792
10.168.3.5	ProcessMsg	792
10.168.3.6	SetFilename	792
10.168.3.7	SetScale	792
10.168.3.8	Update	793
10.168.4	Member Data Documentation	793
10.168.4.1	mesh	793
10.169	gazebo::physics::UniversalJoint< T > Class Template Reference	793
10.169.1	Detailed Description	794
10.169.2	Constructor & Destructor Documentation	794
10.169.2.1	UniversalJoint	794
10.169.2.2	~UniversalJoint	794
10.169.3	Member Function Documentation	794
10.169.3.1	GetAngleCount	794
10.169.3.2	Load	794
10.170	urdf2gazebo::URDF2Gazebo Class Reference	794
10.170.1	Constructor & Destructor Documentation	795
10.170.1.1	URDF2Gazebo	795
10.170.1.2	~URDF2Gazebo	795
10.170.2	Member Function Documentation	795
10.170.2.1	InitModelDoc	795
10.170.2.2	InitModelFile	795
10.170.2.3	InitModelString	796
10.171	gazebo::rendering::UserCamera Class Reference	796
10.171.1	Detailed Description	798
10.171.2	Constructor & Destructor Documentation	798

10.171.2.1	UserCamera	798
10.171.2.2	~UserCamera	798
10.171.3	Member Function Documentation	798
10.171.3.1	AnimationComplete	798
10.171.3.2	AttachToVisualImpl	799
10.171.3.3	EnableViewController	799
10.171.3.4	Finis	799
10.171.3.5	GetAvgFPS	799
10.171.3.6	GetGUIOverlay	799
10.171.3.7	GetTriangleCount	800
10.171.3.8	GetViewControllerTypeString	800
10.171.3.9	GetVisual	800
10.171.3.10	GetVisual	800
10.171.3.11	HandleKeyPressEvent	800
10.171.3.12	HandleKeyReleaseEvent	801
10.171.3.13	HandleMouseEvent	801
10.171.3.14	Hit	801
10.171.3.15	Load	801
10.171.3.16	Load	801
10.171.3.17	MoveToPosition	801
10.171.3.18	MoveToVisual	802
10.171.3.19	MoveToVisual	802
10.171.3.20	PostRender	802
10.171.3.21	Resize	802
10.171.3.22	SetFocalPoint	802
10.171.3.23	SetRenderTarget	802
10.171.3.24	SetViewController	803
10.171.3.25	SetViewController	803
10.171.3.26	SetViewportDimensions	803
10.171.3.27	SetWorldPose	803
10.171.3.28	TrackVisualImpl	803
10.171.3.29	Update	804
10.172	gazebo::math::Vector2d Class Reference	804
10.172.1	Detailed Description	806
10.172.2	Constructor & Destructor Documentation	806
10.172.2.1	Vector2d	806
10.172.2.2	Vector2d	806

10.172.2.3	Vector2d	806
10.172.2.4	Vector2d	806
10.172.3	Member Function Documentation	806
10.172.3.1	Cross	806
10.172.3.2	Distance	806
10.172.3.3	IsFinite	807
10.172.3.4	Normalize	807
10.172.3.5	operator!=	807
10.172.3.6	operator*	807
10.172.3.7	operator*	807
10.172.3.8	operator*==	808
10.172.3.9	operator*==	808
10.172.3.10	operator+	808
10.172.3.11	operator+=	808
10.172.3.12	operator-	809
10.172.3.13	operator-=	809
10.172.3.14	operator/	809
10.172.3.15	operator/	809
10.172.3.16	operator/=	810
10.172.3.17	operator/=	810
10.172.3.18	operator=	810
10.172.3.19	operator=	811
10.172.3.20	operator==	811
10.172.3.21	operator[]	811
10.172.3.22	set	811
10.172.4	Friends And Related Function Documentation	811
10.172.4.1	operator<<	812
10.172.4.2	operator>>	812
10.172.5	Member Data Documentation	812
10.172.5.1	x	812
10.172.5.2	y	812
10.173	Gazebo::math::Vector2i Class Reference	812
10.173.1	Detailed Description	814
10.173.2	Constructor & Destructor Documentation	814
10.173.2.1	Vector2i	814
10.173.2.2	Vector2i	814
10.173.2.3	Vector2i	814

10.173.2.4	~Vector2i	815
10.173.3	Member Function Documentation	815
10.173.3.1	Cross	815
10.173.3.2	Distance	815
10.173.3.3	IsFinite	815
10.173.3.4	Normalize	815
10.173.3.5	operator!=	815
10.173.3.6	operator*	816
10.173.3.7	operator*	816
10.173.3.8	operator*==	816
10.173.3.9	operator*==	817
10.173.3.10	operator+	817
10.173.3.11	operator+=	817
10.173.3.12	operator-	817
10.173.3.13	operator-=	818
10.173.3.14	operator/	818
10.173.3.15	operator/	818
10.173.3.16	operator/=	818
10.173.3.17	operator/=	819
10.173.3.18	operator=	819
10.173.3.19	operator=	819
10.173.3.20	operator==	820
10.173.3.21	operator[]	820
10.173.3.22	set	820
10.173.4	Friends And Related Function Documentation	820
10.173.4.1	operator<<	820
10.173.4.2	operator>>	821
10.173.5	Member Data Documentation	821
10.173.5.1	x	821
10.173.5.2	y	821
10.174	Gazebo::math::Vector3 Class Reference	821
10.174.1	Detailed Description	824
10.174.2	Constructor & Destructor Documentation	824
10.174.2.1	Vector3	824
10.174.2.2	Vector3	824
10.174.2.3	Vector3	824
10.174.2.4	~Vector3	824

10.174.3 Member Function Documentation . . . . .	824
10.174.3.1 Correct . . . . .	825
10.174.3.2 Cross . . . . .	825
10.174.3.3 Distance . . . . .	825
10.174.3.4 Distance . . . . .	825
10.174.3.5 Dot . . . . .	825
10.174.3.6 Equal . . . . .	826
10.174.3.7 GetAbs . . . . .	826
10.174.3.8 GetDistToLine . . . . .	826
10.174.3.9 GetLength . . . . .	826
10.174.3.10 GetMax . . . . .	826
10.174.3.11 GetMin . . . . .	827
10.174.3.12 GetNormal . . . . .	827
10.174.3.13 GetPerpendicular . . . . .	827
10.174.3.14 GetRounded . . . . .	827
10.174.3.15 GetSquaredLength . . . . .	827
10.174.3.16 GetSum . . . . .	828
10.174.3.17 Finite . . . . .	828
10.174.3.18 Normalize . . . . .	828
10.174.3.19 operator!= . . . . .	828
10.174.3.20 operator* . . . . .	828
10.174.3.21 operator* . . . . .	828
10.174.3.22 operator*= . . . . .	829
10.174.3.23 operator*= . . . . .	829
10.174.3.24 operator+ . . . . .	829
10.174.3.25 operator+= . . . . .	830
10.174.3.26 operator- . . . . .	830
10.174.3.27 operator- . . . . .	830
10.174.3.28 operator-= . . . . .	830
10.174.3.29 operator/ . . . . .	830
10.174.3.30 operator/ . . . . .	831
10.174.3.31 operator/= . . . . .	831
10.174.3.32 operator/= . . . . .	831
10.174.3.33 operator= . . . . .	831
10.174.3.34 operator= . . . . .	832
10.174.3.35 operator== . . . . .	832
10.174.3.36 operator[] . . . . .	832

10.174.3.37	Bound	832
10.174.3.38	Bound	832
10.174.3.39	Set	833
10.174.3.40	SetToMax	833
10.174.3.41	SetToMin	833
10.174.4	Friends And Related Function Documentation	833
10.174.4.1	operator*	833
10.174.4.2	operator<<	833
10.174.4.3	operator>>	834
10.174.5	Member Data Documentation	834
10.174.5.1x		834
10.174.5.2y		834
10.174.5.3z		834
10.174.5.4	Zero	834
10.175	Gazebo::math::Vector4 Class Reference	834
10.175.1	Detailed Description	836
10.175.2	Constructor & Destructor Documentation	836
10.175.2.1	Vector4	836
10.175.2.2	Vector4	837
10.175.2.3	Vector4	837
10.175.2.4	~Vector4	837
10.175.3	Member Function Documentation	837
10.175.3.1	Distance	837
10.175.3.2	GetLength	837
10.175.3.3	GetSquaredLength	837
10.175.3.4	IsFinite	838
10.175.3.5	Normalize	838
10.175.3.6	operator!=	838
10.175.3.7	operator*	838
10.175.3.8	operator*	838
10.175.3.9	operator*	839
10.175.3.10	operator*= operator*=	839
10.175.3.11	operator+ operator+=	839
10.175.3.12	operator- operator-=	840

10.175.3.16	operator/	840
10.175.3.17	operator/	841
10.175.3.18	operator/=	841
10.175.3.19	operator/=	841
10.175.3.20	operator=	842
10.175.3.21	operator=	842
10.175.3.22	operator==	842
10.175.3.23	operator[]	842
10.175.3.24	set	843
10.175.4	Friends And Related Function Documentation	843
10.175.4.1	operator<<	843
10.175.4.2	operator>>	843
10.175.5	Member Data Documentation	843
10.175.5.1	w	843
10.175.5.2	x	843
10.175.5.3	y	844
10.175.5.4	z	844
10.176	gazebo::common::Video Class Reference	844
10.176.1	Detailed Description	844
10.176.2	Constructor & Destructor Documentation	844
10.176.2.1	Video	844
10.176.2.2	~Video	844
10.176.3	Member Function Documentation	845
10.176.3.1	GetHeight	845
10.176.3.2	GetNextFrame	845
10.176.3.3	GetWidth	845
10.176.3.4	Load	845
10.177	gazebo::rendering::VideoVisual Class Reference	845
10.177.1	Detailed Description	846
10.177.2	Constructor & Destructor Documentation	846
10.177.2.1	VideoVisual	846
10.177.2.2	~VideoVisual	847
10.178	gazebo::rendering::ViewController Class Reference	847
10.178.1	Detailed Description	848
10.178.2	Constructor & Destructor Documentation	848
10.178.2.1	ViewController	848
10.178.2.2	~ViewController	848

10.178.3	Member Function Documentation . . . . .	848
10.178.3.1	GetTypeString . . . . .	848
10.178.3.2	HandleKeyPressEvent . . . . .	848
10.178.3.3	HandleKeyReleaseEvent . . . . .	849
10.178.3.4	HandleMouseEvent . . . . .	849
10.178.3.5	Init . . . . .	849
10.178.3.6	Init . . . . .	849
10.178.3.7	SetEnabled . . . . .	849
10.178.3.8	Update . . . . .	850
10.178.4	Member Data Documentation . . . . .	850
10.178.4.1	camera . . . . .	850
10.178.4.2	enabled . . . . .	850
10.178.4.3	typeString . . . . .	850
10.179	gazebo::rendering::Visual Class Reference . . . . .	850
10.179.1	Detailed Description . . . . .	855
10.179.2	Constructor & Destructor Documentation . . . . .	855
10.179.2.1	Visual . . . . .	855
10.179.2.2	Visual . . . . .	856
10.179.2.3	~Visual . . . . .	856
10.179.3	Member Function Documentation . . . . .	856
10.179.3.1	AttachAxes . . . . .	856
10.179.3.2	AttachLineVertex . . . . .	856
10.179.3.3	AttachMesh . . . . .	856
10.179.3.4	AttachObject . . . . .	856
10.179.3.5	AttachVisual . . . . .	857
10.179.3.6	ClearParent . . . . .	857
10.179.3.7	Clone . . . . .	857
10.179.3.8	CreateDynamicLine . . . . .	857
10.179.3.9	DeleteDynamicLine . . . . .	857
10.179.3.10	DetachObjects . . . . .	857
10.179.3.11	DetachVisual . . . . .	858
10.179.3.12	DetachVisual . . . . .	858
10.179.3.13	DisableTrackVisual . . . . .	858
10.179.3.14	EnableTrackVisual . . . . .	858
10.179.3.15	Init . . . . .	858
10.179.3.16	GetAttachedObjectCount . . . . .	858
10.179.3.17	GetBoundingBox . . . . .	858



10.179.3.10	GetChild	859
10.179.3.10	GetChildCount	859
10.179.3.20	GetMaterialName	859
10.179.3.20	GetMeshName	859
10.179.3.20	GetName	859
10.179.3.23	GetNormalMap	859
10.179.3.24	GetParent	860
10.179.3.25	GetPose	860
10.179.3.26	GetPosition	860
10.179.3.27	GetRootVisual	860
10.179.3.28	GetRotation	860
10.179.3.29	GetScale	860
10.179.3.30	GetScene	861
10.179.3.30	GetSceneNode	861
10.179.3.30	GetShaderType	861
10.179.3.33	GetTransparency	861
10.179.3.34	GetVisibilityFlags	861
10.179.3.35	GetVisible	862
10.179.3.36	GetWorldPose	862
10.179.3.37	HasAttachedObject	862
10.179.3.38	Hit	862
10.179.3.39	InsertMesh	862
10.179.3.40	InsertMesh	862
10.179.3.41	IsPlane	862
10.179.3.42	IsStatic	863
10.179.3.43	Load	863
10.179.3.44	Load	863
10.179.3.45	LoadFromMsg	863
10.179.3.46	LoadPlugin	863
10.179.3.47	MakeStatic	863
10.179.3.48	MoveToPosition	864
10.179.3.49	MoveToPositions	864
10.179.3.50	RemovePlugin	864
10.179.3.53	SetAmbient	864
10.179.3.53	SetCastShadows	864
10.179.3.53	SetDiffuse	864
10.179.3.53	SetEmissive	865

10.179.3.55	SetHighlighted	865
10.179.3.56	SetMaterial	865
10.179.3.57	SetName	865
10.179.3.58	SetNormalMap	865
10.179.3.59	SetPose	866
10.179.3.60	SetPosition	866
10.179.3.61	SetRibbonTrail	866
10.179.3.62	SetRotation	866
10.179.3.63	SetScale	866
10.179.3.64	SetScene	866
10.179.3.65	SetShaderType	867
10.179.3.66	SetSkeletonPose	867
10.179.3.67	SetSpecular	867
10.179.3.68	SetTransparency	867
10.179.3.69	SetVisibilityFlags	867
10.179.3.70	SetVisible	868
10.179.3.71	SetWorldPose	868
10.179.3.72	SetWorldPosition	868
10.179.3.73	SetWorldRotation	868
10.179.3.74	ShowBoundingBox	868
10.179.3.75	ShowCollision	868
10.179.3.76	ShowCOM	869
10.179.3.77	ShowJoints	869
10.179.3.78	ShowSkeleton	869
10.179.3.79	ToggleVisible	869
10.179.3.80	Update	869
10.179.3.81	UpdateFromMsg	869
10.179.4	Member Data Documentation	869
10.179.4.1	parent	869
10.179.4.2	scene	870
10.179.4.3	sceneNode	870
10.180	Gazebo::VisualPlugin Class Reference	870
10.180.1	Detailed Description	870
10.180.2	Constructor & Destructor Documentation	871
10.180.2.1	VisualPlugin	871
10.180.3	Member Function Documentation	871
10.180.3.1	Init	871

10.180.3.2	Load . . . . .	871
10.180.3.3	Reset . . . . .	871
10.180	gazebo::rendering::WindowManager Class Reference . . . . .	871
10.181.1	Detailed Description . . . . .	872
10.181.2	Member Function Documentation . . . . .	872
10.181.2.1	CreateWindow . . . . .	872
10.181.2.2	Finis . . . . .	873
10.181.2.3	GetAvgFPS . . . . .	873
10.181.2.4	GetTriangleCount . . . . .	873
10.181.2.5	GetWindow . . . . .	873
10.181.2.6	Moved . . . . .	873
10.181.2.7	Resize . . . . .	874
10.181.2.8	SetCamera . . . . .	874
10.180	gazebo::rendering::WireBox Class Reference . . . . .	874
10.182.1	Detailed Description . . . . .	874
10.182.2	Constructor & Destructor Documentation . . . . .	875
10.182.2.1	WireBox . . . . .	875
10.182.2.2	~WireBox . . . . .	875
10.182.3	Member Function Documentation . . . . .	875
10.182.3.1	Init . . . . .	875
10.182.3.2	SetVisible . . . . .	875
10.180	gazebo::physics::World Class Reference . . . . .	875
10.183.1	Detailed Description . . . . .	878
10.183.2	Constructor & Destructor Documentation . . . . .	878
10.183.2.1	World . . . . .	878
10.183.2.2	~World . . . . .	879
10.183.3	Member Function Documentation . . . . .	879
10.183.3.1	Clear . . . . .	879
10.183.3.2	DisableAllModels . . . . .	879
10.183.3.3	EnableAllModels . . . . .	879
10.183.3.4	EnablePhysicsEngine . . . . .	879
10.183.3.5	EnqueueMsg . . . . .	879
10.183.3.6	Finis . . . . .	879
10.183.3.7	GetByName . . . . .	879
10.183.3.8	GetEnablePhysicsEngine . . . . .	880
10.183.3.9	GetEntity . . . . .	880
10.183.3.10	GetEntityBelowPoint . . . . .	880

10.183.3.10	GetModel . . . . .	880
10.183.3.11	GetModel . . . . .	881
10.183.3.12	GetModelBelowPoint . . . . .	881
10.183.3.13	GetModelCount . . . . .	881
10.183.3.14	GetModels . . . . .	881
10.183.3.15	GetName . . . . .	882
10.183.3.16	GetPauseTime . . . . .	882
10.183.3.17	GetPhysicsEngine . . . . .	882
10.183.3.18	GetRealTime . . . . .	882
10.183.3.19	GetSelectedEntity . . . . .	882
10.183.3.20	GetSetWorldPoseMutex . . . . .	883
10.183.3.21	GetSimTime . . . . .	883
10.183.3.22	GetStartTime . . . . .	883
10.183.3.23	Hit . . . . .	883
10.183.3.24	InsertModelFile . . . . .	883
10.183.3.25	InsertModelSDF . . . . .	883
10.183.3.26	InsertModelString . . . . .	884
10.183.3.27	IsLoaded . . . . .	884
10.183.3.28	IsPaused . . . . .	884
10.183.3.29	Load . . . . .	884
10.183.3.30	LoadPlugin . . . . .	884
10.183.3.31	PrintEntityTree . . . . .	885
10.183.3.32	RemovePlugin . . . . .	885
10.183.3.33	Reset . . . . .	885
10.183.3.34	ResetEntities . . . . .	885
10.183.3.35	ResetTime . . . . .	885
10.183.3.36	Run . . . . .	885
10.183.3.37	Save . . . . .	885
10.183.3.38	SetPaused . . . . .	886
10.183.3.39	SetSimTime . . . . .	886
10.183.3.40	SetState . . . . .	886
10.183.3.41	StepWorld . . . . .	886
10.183.3.42	Stop . . . . .	886
10.183.3.43	StripWorldName . . . . .	886
10.183.3.44	UpdateStateSDF . . . . .	887
10.183.4	Member Data Documentation . . . . .	887
10.183.4.1	dirtyPoses . . . . .	887

10.184	gazebo::WorldPlugin Class Reference . . . . .	887
10.184.1	Detailed Description . . . . .	888
10.184.2	Constructor & Destructor Documentation . . . . .	888
10.184.2.1	WorldPlugin . . . . .	888
10.184.2.2	~WorldPlugin . . . . .	888
10.184.3	Member Function Documentation . . . . .	888
10.184.3.1	Init . . . . .	888
10.184.3.2	Load . . . . .	888
10.184.3.3	Reset . . . . .	888
10.185	gazebo::physics::WorldState Class Reference . . . . .	888
10.185.1	Detailed Description . . . . .	890
10.185.2	Constructor & Destructor Documentation . . . . .	890
10.185.2.1	WorldState . . . . .	890
10.185.2.2	WorldState . . . . .	890
10.185.2.3	WorldState . . . . .	890
10.185.2.4	~WorldState . . . . .	890
10.185.3	Member Function Documentation . . . . .	890
10.185.3.1	FillSDF . . . . .	890
10.185.3.2	GetModelState . . . . .	891
10.185.3.3	GetModelState . . . . .	891
10.185.3.4	GetModelStateCount . . . . .	891
10.185.3.5	GetModelStates . . . . .	891
10.185.3.6	HasModelState . . . . .	892
10.185.3.7	IsZero . . . . .	892
10.185.3.8	Load . . . . .	892
10.185.3.9	operator+ . . . . .	892
10.185.3.10	operator- . . . . .	893
10.185.3.11	operator= . . . . .	893
10.185.4	Friends And Related Function Documentation . . . . .	893
10.185.4.1	operator<< . . . . .	893
<b>11</b>	<b>File Documentation</b>	<b>895</b>
11.1	Actor.hh File Reference . . . . .	895
11.2	Angle.hh File Reference . . . . .	896
11.2.1	Macro Definition Documentation . . . . .	898
11.2.1.1	GZ_DTOR . . . . .	898
11.2.1.2	GZ_NORMALIZE . . . . .	898

11.2.1.3 GZ_RTOD . . . . .	898
11.3 Animation.hh File Reference . . . . .	899
11.4 ArrowVisual.hh File Reference . . . . .	900
11.5 Assert.hh File Reference . . . . .	901
11.5.1 Macro Definition Documentation . . . . .	902
11.5.1.1 GZ_ASSERT . . . . .	902
11.6 AxisVisual.hh File Reference . . . . .	902
11.7 BallJoint.hh File Reference . . . . .	903
11.8 Base.hh File Reference . . . . .	904
11.9 Box.hh File Reference . . . . .	905
11.10BoxShape.hh File Reference . . . . .	905
11.11BVHLoader.hh File Reference . . . . .	907
11.11.1 Macro Definition Documentation . . . . .	908
11.11.1.1 X_POSITION . . . . .	908
11.11.1.2 X_ROTATION . . . . .	908
11.11.1.3 Y_POSITION . . . . .	908
11.11.1.4 Y_ROTATION . . . . .	908
11.11.1.5 Z_POSITION . . . . .	908
11.11.1.6 Z_ROTATION . . . . .	908
11.12CallbackHelper.hh File Reference . . . . .	908
11.13Camera.hh File Reference . . . . .	910
11.14CameraSensor.hh File Reference . . . . .	911
11.15CameraVisual.hh File Reference . . . . .	912
11.16cegui.h File Reference . . . . .	912
11.17ColladaLoader.hh File Reference . . . . .	913
11.18Collision.hh File Reference . . . . .	915
11.19CollisionState.hh File Reference . . . . .	916
11.20Color.hh File Reference . . . . .	918
11.21Common.hh File Reference . . . . .	918
11.22CommonTypes.hh File Reference . . . . .	920
11.22.1 Macro Definition Documentation . . . . .	921
11.22.1.1 GAZEBO_DEPRECATED . . . . .	921
11.22.1.2 GAZEBO_FORCEINLINE . . . . .	921
11.22.1.3 NULL . . . . .	921
11.23COMVisual.hh File Reference . . . . .	921
11.24Connection.hh File Reference . . . . .	922
11.24.1 Macro Definition Documentation . . . . .	924

---

11.24.1.1 HEADER_LENGTH . . . . .	924
11.25ConnectionManager.hh File Reference . . . . .	924
11.26Console.hh File Reference . . . . .	925
11.27Contact.hh File Reference . . . . .	927
11.27.1 Macro Definition Documentation . . . . .	928
11.27.1.1 MAX_COLLIDE_RETURNS . . . . .	928
11.27.1.2 MAX_CONTACT_JOINTS . . . . .	928
11.28ContactManager.hh File Reference . . . . .	928
11.29ContactSensor.hh File Reference . . . . .	929
11.30ContactVisual.hh File Reference . . . . .	930
11.31Conversions.hh File Reference . . . . .	931
11.32Converter.hh File Reference . . . . .	931
11.33CylinderShape.hh File Reference . . . . .	932
11.34DepthCamera.hh File Reference . . . . .	933
11.35DepthCameraSensor.hh File Reference . . . . .	934
11.36Diagnostics.hh File Reference . . . . .	935
11.37DynamicLines.hh File Reference . . . . .	937
11.38DynamicRenderable.hh File Reference . . . . .	937
11.39Entity.hh File Reference . . . . .	938
11.40Event.hh File Reference . . . . .	940
11.41Events.hh File Reference . . . . .	940
11.42Exception.hh File Reference . . . . .	941
11.43FPSViewController.hh File Reference . . . . .	943
11.44gazebo.hh File Reference . . . . .	944
11.45gazebo_core.hh File Reference . . . . .	945
11.46GazeboGenerator.hh File Reference . . . . .	945
11.47GpuLaser.hh File Reference . . . . .	946
11.48GpuRaySensor.hh File Reference . . . . .	947
11.49Grid.hh File Reference . . . . .	948
11.50Gripper.hh File Reference . . . . .	948
11.51GUIOverlay.hh File Reference . . . . .	950
11.52Heightmap.hh File Reference . . . . .	950
11.53HeightmapShape.hh File Reference . . . . .	951
11.54Helpers.hh File Reference . . . . .	952
11.54.1 Macro Definition Documentation . . . . .	954
11.54.1.1 GZ_DBL_MAX . . . . .	954
11.54.1.2 GZ_DBL_MIN . . . . .	954

11.54.1.3 GZ_FLT_MAX . . . . .	954
11.54.1.4 GZ_FLT_MIN . . . . .	954
11.55Hinge2Joint.hh File Reference . . . . .	954
11.56HingeJoint.hh File Reference . . . . .	956
11.57Image.hh File Reference . . . . .	957
11.58ImuSensor.hh File Reference . . . . .	958
11.59Inertial.hh File Reference . . . . .	959
11.60IOManager.hh File Reference . . . . .	960
11.61Joint.hh File Reference . . . . .	961
11.62JointController.hh File Reference . . . . .	962
11.63JointState.hh File Reference . . . . .	963
11.64JointVisual.hh File Reference . . . . .	964
11.65JointWrench.hh File Reference . . . . .	965
11.66KeyFrame.hh File Reference . . . . .	966
11.67LaserVisual.hh File Reference . . . . .	967
11.68Light.hh File Reference . . . . .	968
11.69Link.hh File Reference . . . . .	969
11.70LinkState.hh File Reference . . . . .	970
11.71LogPlay.hh File Reference . . . . .	972
11.72LogRecord.hh File Reference . . . . .	973
11.72.1 Macro Definition Documentation . . . . .	975
11.72.1.1 GZ_LOG_VERSION . . . . .	975
11.73mainpage.html File Reference . . . . .	975
11.74MapShape.hh File Reference . . . . .	975
11.75Master.hh File Reference . . . . .	976
11.76Material.hh File Reference . . . . .	977
11.77Material.hh File Reference . . . . .	979
11.78MathTypes.hh File Reference . . . . .	979
11.78.1 Detailed Description . . . . .	979
11.79Matrix3.hh File Reference . . . . .	980
11.80Matrix4.hh File Reference . . . . .	980
11.81Mesh.hh File Reference . . . . .	981
11.82MeshCSG.hh File Reference . . . . .	983
11.82.1 Typedef Documentation . . . . .	983
11.82.1.1 GPtrArray . . . . .	983
11.82.1.2 GtsSurface . . . . .	984
11.83MeshLoader.hh File Reference . . . . .	984



11.84MeshManager.hh File Reference . . . . .	985
11.85Model.hh File Reference . . . . .	986
11.86ModelDatabase.hh File Reference . . . . .	988
11.86.1 Macro Definition Documentation . . . . .	988
11.86.1.1 GZ_MODEL_DB_MANIFEST_FILENAME . . . . .	988
11.86.1.2 GZ_MODEL_MANIFEST_FILENAME . . . . .	989
11.87ModelState.hh File Reference . . . . .	989
11.88MouseEvent.hh File Reference . . . . .	991
11.89MovableText.hh File Reference . . . . .	992
11.90MsgFactory.hh File Reference . . . . .	992
11.91msgs.hh File Reference . . . . .	993
11.92MultiCameraSensor.hh File Reference . . . . .	996
11.93MultiRayShape.hh File Reference . . . . .	996
11.94Node.hh File Reference . . . . .	998
11.95ogre_gazebo.h File Reference . . . . .	999
11.96OrbitViewController.hh File Reference . . . . .	1000
11.97Param.hh File Reference . . . . .	1000
11.98parser.hh File Reference . . . . .	1002
11.99parser_urdf.hh File Reference . . . . .	1003
11.100Physics.hh File Reference . . . . .	1004
11.101PhysicsEngine.hh File Reference . . . . .	1005
11.102PhysicsFactory.hh File Reference . . . . .	1007
11.103PhysicsTypes.hh File Reference . . . . .	1008
11.103.1 Detailed Description . . . . .	1010
11.103.2 Macro Definition Documentation . . . . .	1010
11.103.2.1GZ_ALL_COLLIDE . . . . .	1010
11.103.2.2GZ_FIXED_COLLIDE . . . . .	1010
11.103.2.3GZ_GHOST_COLLIDE . . . . .	1010
11.103.2.4GZ_NONE_COLLIDE . . . . .	1010
11.103.2.5GZ_SENSOR_COLLIDE . . . . .	1010
11.104PID.hh File Reference . . . . .	1011
11.105Plane.hh File Reference . . . . .	1012
11.106PlaneShape.hh File Reference . . . . .	1013
11.107Plugin.hh File Reference . . . . .	1014
11.107.1 Macro Definition Documentation . . . . .	1016
11.107.1.1GZ_REGISTER_MODEL_PLUGIN . . . . .	1016
11.107.1.2GZ_REGISTER_SENSOR_PLUGIN . . . . .	1016

11.107.1.3GZ_REGISTER_SYSTEM_PLUGIN . . . . .	1017
11.107.1.4GZ_REGISTER_VISUAL_PLUGIN . . . . .	1017
11.107.1.5GZ_REGISTER_WORLD_PLUGIN . . . . .	1017
11.108Plugin.hh File Reference . . . . .	1018
11.109Pose.hh File Reference . . . . .	1018
11.110Projector.hh File Reference . . . . .	1019
11.111Publication.hh File Reference . . . . .	1020
11.112PublicationTransport.hh File Reference . . . . .	1021
11.113Publisher.hh File Reference . . . . .	1023
11.114Quaternion.hh File Reference . . . . .	1025
11.115Sand.hh File Reference . . . . .	1026
11.116RaySensor.hh File Reference . . . . .	1027
11.117RayShape.hh File Reference . . . . .	1028
11.118RenderEngine.hh File Reference . . . . .	1029
11.119RenderEvents.hh File Reference . . . . .	1030
11.120Rendering.hh File Reference . . . . .	1031
11.121RenderTypes.hh File Reference . . . . .	1032
11.121. Macro Definition Documentation . . . . .	1033
11.121.1.1GZ_VISIBILITY_ALL . . . . .	1033
11.121.1.2GZ_VISIBILITY_GUI . . . . .	1033
11.121.1.3GZ_VISIBILITY_NOT_SELECTABLE . . . . .	1033
11.121.1.4GZ_VISIBILITY_SELECTION . . . . .	1034
11.122RFIDSensor.hh File Reference . . . . .	1034
11.123RFIDTag.hh File Reference . . . . .	1034
11.124RFIDTagVisual.hh File Reference . . . . .	1035
11.125RFIDVisual.hh File Reference . . . . .	1036
11.126Road.hh File Reference . . . . .	1036
11.127Road2d.hh File Reference . . . . .	1038
11.128RotationSpline.hh File Reference . . . . .	1038
11.129RTShaderSystem.hh File Reference . . . . .	1039
11.130Scene.hh File Reference . . . . .	1040
11.131ScrewJoint.hh File Reference . . . . .	1041
11.132df.hh File Reference . . . . .	1042
11.133SDF.hh File Reference . . . . .	1043
11.133. Macro Definition Documentation . . . . .	1044
11.133.1.1SDF_VERSION . . . . .	1044
11.134SelectionObj.hh File Reference . . . . .	1044

11.135	Sensor.hh File Reference	1045
11.136	SensorFactory.hh File Reference	1046
11.137	SensorManager.hh File Reference	1047
11.138	Sensors.hh File Reference	1048
11.139	SensorTypes.hh File Reference	1049
11.139.1	Detailed Description	1051
11.140	Server.hh File Reference	1051
11.141	Shape.hh File Reference	1052
11.142	SingletonT.hh File Reference	1053
11.143	Skeleton.hh File Reference	1053
11.144	SkeletonAnimation.hh File Reference	1055
11.145	SliderJoint.hh File Reference	1056
11.146	SphereShape.hh File Reference	1058
11.147	Spline.hh File Reference	1059
11.148	State.hh File Reference	1060
11.149	STLLoader.hh File Reference	1061
11.149.1	Macro Definition Documentation	1062
11.149.1.1	COR3_MAX	1062
11.149.1.2	FACE_MAX	1063
11.149.1.3	LINE_MAX_LEN	1063
11.149.1.4	ORDER_MAX	1063
11.150	SubscribeOptions.hh File Reference	1063
11.151	Subscriber.hh File Reference	1064
11.152	SubscriptionTransport.hh File Reference	1066
11.153	SurfaceParams.hh File Reference	1067
11.154	SystemPaths.hh File Reference	1069
11.154.1	Macro Definition Documentation	1070
11.154.1.1	GetCurrentDir	1070
11.154.1.2	LINUX	1070
11.155	Time.hh File Reference	1070
11.156	Timer.hh File Reference	1071
11.157	TopicManager.hh File Reference	1072
11.158	Transport.hh File Reference	1074
11.159	TransportTypes.hh File Reference	1076
11.159.1	Detailed Description	1077
11.160	TrimeshShape.hh File Reference	1078
11.161	UniversalJoint.hh File Reference	1079

11.162	UserCamera.hh File Reference . . . . .	1080
11.163	Vector2d.hh File Reference . . . . .	1080
11.164	Vector2i.hh File Reference . . . . .	1081
11.165	Vector3.hh File Reference . . . . .	1082
11.166	Vector4.hh File Reference . . . . .	1083
11.167	Video.hh File Reference . . . . .	1085
11.168	VideoVisual.hh File Reference . . . . .	1086
11.169	ViewController.hh File Reference . . . . .	1087
11.170	Visual.hh File Reference . . . . .	1088
11.171	WindowManager.hh File Reference . . . . .	1089
11.172	WireBox.hh File Reference . . . . .	1089
11.173	World.hh File Reference . . . . .	1090
11.174	WorldState.hh File Reference . . . . .	1092

**Index****1093**

# 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. 107)** 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. 107)** set of rendering related class, functions, and definitions.

**Sensors: A (p. 107)** 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. 107)** 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. 377) (int \_index)**

: not yet implemented. Get the internal forces at a 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. 185) () const**

to be implemented

**Class gazebo::SystemPlugin (p. 759)**

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 . . . . .	73



## Chapter 4

# Namespace Index

### 4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<b>boost</b>	79
<b>gazebo</b>	
Forward declarations for the common classes	79
<b>gazebo::common</b>	
Common namespace	81
<b>gazebo::event</b>	
<b>Event</b> (p. 285) namespace	84
<b>gazebo::math</b>	
Math namespace	85
<b>gazebo::msgs</b>	
Messages namespace	87
<b>gazebo::physics</b>	
Namespace for physics	89
<b>gazebo::rendering</b>	
Rendering namespace	94
<b>gazebo::sensors</b>	
Sensors namespace	98
<b>gazebo::transport</b>	100
<b>google</b>	102
<b>google::protobuf</b>	102
<b>google::protobuf::compiler</b>	102
<b>google::protobuf::compiler::cpp</b>	102
<b>Ogre</b>	103
<b>ogre</b>	103
<b>sdf</b>	
Namespace for Simulation Description Format parser	103
<b>SkyX</b>	105
<b>urdf2gazebo</b>	
Namespace for URDF to SDF parser	105



# Chapter 5

## Hierarchical Index

### 5.1 Class Hierarchy

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

A	107
gazebo::math::Angle	114
gazebo::common::Animation	121
gazebo::common::NumericAnimation	531
gazebo::common::PoseAnimation	581
gazebo::math::Box	144
gazebo::common::BVHLoader	152
gazebo::transport::CallbackHelper	153
gazebo::transport::CallbackHelperT< M >	155
gazebo::transport::RawCallbackHelper	614
gazebo::transport::SubscriptionTransport	748
CodeGenerator	
google::protobuf::compiler::cpp::GazeboGenerator	315
gazebo::common::Color	203
gazebo::event::Connection	216
gazebo::physics::Contact	229
gazebo::physics::ContactManager	232
gazebo::rendering::Conversions	240
sdf::Converter	242
enable_shared_from_this	
gazebo::physics::Base	133
gazebo::physics::Entity	274
gazebo::physics::Collision	190
gazebo::physics::Link	404
gazebo::physics::Model	469
gazebo::physics::Actor	107
gazebo::physics::Joint	371
gazebo::physics::Road	640
gazebo::physics::Shape	693
gazebo::physics::BoxShape	149
gazebo::physics::CylinderShape	243
gazebo::physics::HeightmapShape	344
gazebo::physics::MultiRayShape	507

gazebo::physics::PlaneShape . . . . .	566
gazebo::physics::RayShape . . . . .	622
gazebo::physics::SphereShape . . . . .	722
gazebo::physics::TrimeshShape . . . . .	790
gazebo::physics::World . . . . .	875
gazebo::rendering::Camera . . . . .	157
gazebo::rendering::DepthCamera . . . . .	246
gazebo::rendering::GpuLaser . . . . .	316
gazebo::rendering::UserCamera . . . . .	796
gazebo::rendering::Scene . . . . .	651
gazebo::rendering::Visual . . . . .	850
gazebo::rendering::ArrowVisual . . . . .	125
gazebo::rendering::AxisVisual . . . . .	129
gazebo::rendering::CameraVisual . . . . .	187
gazebo::rendering::COMVisual . . . . .	214
gazebo::rendering::ContactVisual . . . . .	238
gazebo::rendering::JointVisual . . . . .	391
gazebo::rendering::LaserVisual . . . . .	396
gazebo::rendering::RFIDTagVisual . . . . .	637
gazebo::rendering::RFIDVisual . . . . .	639
gazebo::rendering::VideoVisual . . . . .	845
gazebo::sensors::Sensor . . . . .	672
gazebo::sensors::CameraSensor . . . . .	183
gazebo::sensors::ContactSensor . . . . .	234
gazebo::sensors::DepthCameraSensor . . . . .	250
gazebo::sensors::GpuRaySensor . . . . .	320
gazebo::sensors::ImuSensor . . . . .	357
gazebo::sensors::MultiCameraSensor . . . . .	503
gazebo::sensors::RaySensor . . . . .	616
gazebo::sensors::RFIDSensor . . . . .	633
gazebo::sensors::RFIDTag . . . . .	635
gazebo::transport::Connection . . . . .	217
gazebo::transport::Node . . . . .	515
sdf::Element . . . . .	266
gazebo::event::Event . . . . .	285
gazebo::event::EventT< void()> . . . . .	302
gazebo::event::EventT< void(bool)> . . . . .	302
gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)> . . . . .	302
gazebo::event::EventT< void(const std::string &)> . . . . .	302
gazebo::event::EventT< void(const std::string &, const Contact &)> . . . . .	302
gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)> . . . . .	302
gazebo::event::EventT< void(std::string)> . . . . .	302
gazebo::event::EventT< void(std::string, std::string)> . . . . .	302
gazebo::event::EventT< T > . . . . .	302
gazebo::rendering::Events . . . . .	288
gazebo::event::Events . . . . .	290
gazebo::common::Exception . . . . .	310
gazebo::common::InternalError . . . . .	368
gazebo::common::AssertionInternalError . . . . .	127
urdf2gazebo::GazeboExtension . . . . .	315
gazebo::rendering::Grid . . . . .	332
gazebo::physics::Gripper . . . . .	336

gazebo::rendering::GUIOverlay . . . . .	337
gazebo::rendering::Heightmap . . . . .	342
gazebo::common::Image . . . . .	352
gazebo::physics::Inertial . . . . .	360
gazebo::transport::IOManager . . . . .	369
gazebo::physics::JointController . . . . .	385
gazebo::physics::JointWrench . . . . .	393
gazebo::common::KeyFrame . . . . .	394
gazebo::common::NumericKeyFrame . . . . .	533
gazebo::common::PoseKeyFrame . . . . .	584
gazebo::rendering::Light . . . . .	397
Logplay . . . . .	430
gazebo::Master . . . . .	433
gazebo::common::Material . . . . .	435
gazebo::math::Matrix3 . . . . .	444
gazebo::math::Matrix4 . . . . .	448
gazebo::common::Mesh . . . . .	455
gazebo::common::MeshCSG . . . . .	461
gazebo::common::MeshLoader . . . . .	463
gazebo::common::ColladaLoader . . . . .	188
gazebo::common::STLLoader . . . . .	733
gazebo::common::MouseEvent . . . . .	492
MovableObject	
gazebo::rendering::MovableText . . . . .	495
gazebo::msgs::MsgFactory . . . . .	501
gazebo::common::NodeAnimation . . . . .	521
gazebo::common::NodeAssignment . . . . .	525
gazebo::common::NodeTransform . . . . .	526
sdf::Param . . . . .	538
sdf::ParamT< std::string > . . . . .	544
sdf::ParamT< T > . . . . .	544
ParamT< T > . . . . .	544
gazebo::physics::PhysicsEngine . . . . .	547
gazebo::physics::PhysicsFactory . . . . .	558
gazebo::common::PID . . . . .	559
gazebo::math::Plane . . . . .	563
gazebo::PluginT< T > . . . . .	571
gazebo::PluginT< ModelPlugin > . . . . .	571
gazebo::ModelPlugin . . . . .	484
gazebo::PluginT< SensorPlugin > . . . . .	571
gazebo::SensorPlugin . . . . .	686
gazebo::PluginT< SystemPlugin > . . . . .	571
gazebo::SystemPlugin . . . . .	759
gazebo::PluginT< VisualPlugin > . . . . .	571
gazebo::VisualPlugin . . . . .	870
gazebo::PluginT< WorldPlugin > . . . . .	571
gazebo::WorldPlugin . . . . .	887
gazebo::math::Pose . . . . .	573
gazebo::rendering::Projector . . . . .	586
gazebo::transport::Publication . . . . .	588
gazebo::transport::PublicationTransport . . . . .	593
gazebo::transport::Publisher . . . . .	595

gazebo::math::Quaternion . . . . .	598
gazebo::math::Rand . . . . .	612
Renderable	
gazebo::rendering::MovableText . . . . .	495
RenderObjectListener	
gazebo::rendering::GpuLaser . . . . .	316
Road . . . . .	640
gazebo::rendering::Road2d . . . . .	642
gazebo::math::RotationSpline . . . . .	643
sdf::SDF . . . . .	669
SDFBase	
sdf::Plugin . . . . .	569
gazebo::rendering::SelectionObj . . . . .	670
SensorFactor . . . . .	681
gazebo::sensors::SensorFactory . . . . .	681
gazebo::Server . . . . .	688
ShaderHelperCg	
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg . . . . .	689
ShaderHelperGLSL	
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL . . . . .	691
SimpleRenderable	
gazebo::rendering::DynamicRenderable . . . . .	262
gazebo::rendering::DynamicLines . . . . .	258
SingletonT< T > . . . . .	696
gazebo::common::Console . . . . .	228
gazebo::common::DiagnosticManager . . . . .	254
gazebo::common::LogPlay . . . . .	427
gazebo::common::LogRecord . . . . .	431
gazebo::common::MeshManager . . . . .	464
gazebo::common::ModelDatabase . . . . .	482
gazebo::common::SystemPaths . . . . .	754
gazebo::rendering::RenderEngine . . . . .	628
gazebo::rendering::RTShaderSystem . . . . .	647
gazebo::sensors::SensorManager . . . . .	683
gazebo::transport::ConnectionManager . . . . .	223
gazebo::transport::TopicManager . . . . .	782
SingletonT< ConnectionManager > . . . . .	696
SingletonT< Console > . . . . .	696
SingletonT< DiagnosticManager > . . . . .	696
SingletonT< LogPlay > . . . . .	696
SingletonT< LogRecord > . . . . .	696
SingletonT< MeshManager > . . . . .	696
SingletonT< ModelDatabase > . . . . .	696
SingletonT< RenderEngine > . . . . .	696
SingletonT< RTShaderSystem > . . . . .	696
SingletonT< SensorManager > . . . . .	696
SingletonT< SystemPaths > . . . . .	696
SingletonT< TopicManager > . . . . .	696
SingletonT< WindowManager > . . . . .	696
gazebo::rendering::WindowManager . . . . .	871
gazebo::common::Skeleton . . . . .	698
gazebo::common::SkeletonAnimation . . . . .	705
gazebo::common::SkeletonNode . . . . .	709
SM2Profile	



gazebo::rendering::GzTerrainMatGen::SM2Profile . . . . .	720
gazebo::math::Spline . . . . .	725
gazebo::physics::State . . . . .	729
gazebo::physics::CollisionState . . . . .	199
gazebo::physics::JointState . . . . .	387
gazebo::physics::LinkState . . . . .	422
gazebo::physics::ModelState . . . . .	485
gazebo::physics::WorldState . . . . .	888
gazebo::common::SubMesh . . . . .	735
gazebo::transport::SubscribeOptions . . . . .	744
gazebo::transport::Subscriber . . . . .	746
gazebo::physics::SurfaceParams . . . . .	750
T	
gazebo::physics::BallJoint< T > . . . . .	131
gazebo::physics::Hinge2Joint< T > . . . . .	348
gazebo::physics::HingeJoint< T > . . . . .	350
gazebo::physics::ScrewJoint< T > . . . . .	666
gazebo::physics::SliderJoint< T > . . . . .	718
gazebo::physics::UniversalJoint< T > . . . . .	793
TerrainMaterialGeneratorA	
gazebo::rendering::GzTerrainMatGen . . . . .	341
gazebo::common::Time . . . . .	760
gazebo::common::Timer . . . . .	781
gazebo::common::DiagnosticTimer . . . . .	257
gazebo::physics::TrajectoryInfo . . . . .	789
urdf2gazebo::URDF2Gazebo . . . . .	794
gazebo::math::Vector2d . . . . .	804
gazebo::math::Vector2i . . . . .	812
gazebo::math::Vector3 . . . . .	821
gazebo::math::Vector4 . . . . .	834
gazebo::common::Video . . . . .	844
gazebo::rendering::ViewController . . . . .	847
gazebo::rendering::FPSViewController . . . . .	312
gazebo::rendering::OrbitViewController . . . . .	534
gazebo::rendering::WireBox . . . . .	874



# 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 . . . . .	107
<b>gazebo::physics::Actor</b>	
<b>Actor</b> (p. 107) class enables GPU based mesh model / skeleton scriptable animation . . . . .	107
<b>gazebo::math::Angle</b>	
An angle and related functions . . . . .	114
<b>gazebo::common::Animation</b>	
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes . . . . .	121
<b>gazebo::rendering::ArrowVisual</b>	
Basic arrow visualization . . . . .	125
<b>gazebo::common::AssertionInternalError</b>	
Class for generating Exceptions which come from gazebo assertions . . . . .	127
<b>gazebo::rendering::AxisVisual</b>	
Basic axis visualization . . . . .	129
<b>gazebo::physics::BallJoint&lt; T &gt;</b>	
<b>Base</b> (p. 133) class for a ball joint . . . . .	131
<b>gazebo::physics::Base</b>	
<b>Base</b> (p. 133) class for most physics classes . . . . .	133
<b>gazebo::math::Box</b>	
Mathematical representation of a box and related functions . . . . .	144
<b>gazebo::physics::BoxShape</b>	
Box geometry primitive . . . . .	149
<b>gazebo::common::BVHLoader</b>	
Handles loading BVH animation files . . . . .	152
<b>gazebo::transport::CallbackHelper</b>	
<b>A</b> (p. 107) helper class to handle callbacks when messages arrive . . . . .	153
<b>gazebo::transport::CallbackHelperT&lt; M &gt;</b>	
Callback helper Template . . . . .	155
<b>gazebo::rendering::Camera</b>	
Basic camera sensor . . . . .	157
<b>gazebo::sensors::CameraSensor</b>	
Basic camera sensor . . . . .	183

<b>gazebo::rendering::CameraVisual</b>	
Basic camera visualization	187
<b>gazebo::common::ColladaLoader</b>	
Class used to load Collada mesh files	188
<b>gazebo::physics::Collision</b>	
<b>Base</b> (p. 133) class for all collision entities	190
<b>gazebo::physics::CollisionState</b>	
Store state information of a <b>physics::Collision</b> (p. 190) object	199
<b>gazebo::common::Color</b>	
Defines a color	203
<b>gazebo::rendering::COMVisual</b>	
Basic Center of Mass visualization	214
<b>gazebo::event::Connection</b>	
<b>A</b> (p. 107) class that encapsulates a connection	216
<b>gazebo::transport::Connection</b>	
Single TCP/IP connection manager	217
<b>gazebo::transport::ConnectionManager</b>	
Manager of connections	223
<b>gazebo::common::Console</b>	
Message, error, warning functionality	228
<b>gazebo::physics::Contact</b>	
<b>A</b> (p. 107) contact between two collisions	229
<b>gazebo::physics::ContactManager</b>	
Aggregates all the contact information generated by the collision detection engine	232
<b>gazebo::sensors::ContactSensor</b>	
Contact sensor	234
<b>gazebo::rendering::ContactVisual</b>	
Contact visualization	238
<b>gazebo::rendering::Conversions</b>	
<b>Conversions</b> (p. 240) <b>Conversions.hh</b> (p. 931) <b>rendering/Conversions.hh</b> (p. 931)	240
<b>sdf::Converter</b>	
Convert from one version of <b>SDF</b> (p. 669) to another	242
<b>gazebo::physics::CylinderShape</b>	
Cylinder collision	243
<b>gazebo::rendering::DepthCamera</b>	
Depth camera used to render depth data into an image buffer	246
<b>gazebo::sensors::DepthCameraSensor</b>	250
<b>gazebo::common::DiagnosticManager</b>	
<b>A</b> (p. 107) diagnostic manager class	254
<b>gazebo::common::DiagnosticTimer</b>	
<b>A</b> (p. 107) timer designed for diagnostics	257
<b>gazebo::rendering::DynamicLines</b>	
Class for drawing lines that can change	258
<b>gazebo::rendering::DynamicRenderable</b>	
Abstract base class providing mechanisms for dynamically growing hardware buffers	262
<b>sdf::Element</b>	
<b>SDF</b> (p. 669) <b>Element</b> (p. 266) class	266
<b>gazebo::physics::Entity</b>	
<b>Base</b> (p. 133) class for all physics objects in Gazebo	274
<b>gazebo::event::Event</b>	
Base class for all events	285
<b>gazebo::rendering::Events</b>	
Base class for rendering events	288

<b>gazebo::event::Events</b>	
An <b>Event</b> (p. 285) class to get notifications for simulator events	290
<b>gazebo::event::EventT&lt; T &gt;</b>	
<b>A</b> (p. 107) class for event processing	302
<b>gazebo::common::Exception</b>	
Class for generating exceptions	310
<b>gazebo::rendering::FPSViewController</b>	
First Person Shooter style view controller	312
<b>urdf2gazebo::GazeboExtension</b>	315
<b>google::protobuf::compiler::cpp::GazeboGenerator</b>	
Google protobuf message generator for <b>gazebo::msgs</b> (p. 87)	315
<b>gazebo::rendering::GpuLaser</b>	
GPU based laser distance sensor	316
<b>gazebo::sensors::GpuRaySensor</b>	320
<b>gazebo::rendering::Grid</b>	
Displays a grid of cells, drawn with lines	332
<b>gazebo::physics::Gripper</b>	
<b>A</b> (p. 107) gripper abstraction	336
<b>gazebo::rendering::GUIOverlay</b>	
<b>A</b> (p. 107) class that creates a CEGUI overlay on a render window	337
<b>gazebo::rendering::GzTerrainMatGen</b>	341
<b>gazebo::rendering::Heightmap</b>	
Rendering a terrain using heightmap information	342
<b>gazebo::physics::HeightmapShape</b>	
<b>HeightmapShape</b> (p. 344) collision shape builds a heightmap from an image	344
<b>gazebo::physics::Hinge2Joint&lt; T &gt;</b>	
<b>A</b> (p. 107) two axis hinge joint	348
<b>gazebo::physics::HingeJoint&lt; T &gt;</b>	
<b>A</b> (p. 107) single axis hinge joint	350
<b>gazebo::common::Image</b>	
Encapsulates an image	352
<b>gazebo::sensors::ImuSensor</b>	
An IMU sensor	357
<b>gazebo::physics::Inertial</b>	
<b>A</b> (p. 107) class for inertial information about a link	360
<b>gazebo::common::InternalError</b>	
Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs	368
<b>gazebo::transport::IOManager</b>	
Manages boost::asio IO	369
<b>gazebo::physics::Joint</b>	
<b>Base</b> (p. 133) class for all joints	371
<b>gazebo::physics::JointController</b>	
<b>A</b> (p. 107) class for manipulating <b>physics::Joint</b> (p. 371)	385
<b>gazebo::physics::JointState</b>	
Keeps track of state of a <b>physics::Joint</b> (p. 371)	387
<b>gazebo::rendering::JointVisual</b>	
Visualization for joints	391
<b>gazebo::physics::JointWrench</b>	
Wrench information from a joint	393
<b>gazebo::common::KeyFrame</b>	
<b>A</b> (p. 107) key frame in an animation	394
<b>gazebo::rendering::LaserVisual</b>	
Visualization for laser data	396

<b>gazebo::rendering::Light</b>	
<b>A</b> (p. 107) light source	397
<b>gazebo::physics::Link</b>	
<b>Link</b> (p. 404) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body	404
<b>gazebo::physics::LinkState</b>	
Store state information of a <b>physics::Link</b> (p. 404) object	422
<b>gazebo::common::LogPlay</b>	427
<b>Logplay</b>	
Open and playback log files that were recorded using LogRecord	430
<b>gazebo::common::LogRecord</b>	
Addtogroup gazebo_common	431
<b>gazebo::Master</b>	
<b>A</b> (p. 107) ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication	433
<b>gazebo::common::Material</b>	
Encapsulates description of a material	435
<b>gazebo::math::Matrix3</b>	
<b>A</b> (p. 107) 3x3 matrix class	444
<b>gazebo::math::Matrix4</b>	
<b>A</b> (p. 107) 3x3 matrix class	448
<b>gazebo::common::Mesh</b>	
<b>A</b> (p. 107) 3D mesh	455
<b>gazebo::common::MeshCSG</b>	
Creates CSG meshes	461
<b>gazebo::common::MeshLoader</b>	
Base class for loading meshes	463
<b>gazebo::common::MeshManager</b>	
Maintains and manages all meshes	464
<b>gazebo::physics::Model</b>	
<b>A</b> (p. 107) model is a collection of links, joints, and plugins	469
<b>gazebo::common::ModelDatabase</b>	
Connects to model database, and has utility functions to find models	482
<b>gazebo::ModelPlugin</b>	
<b>A</b> (p. 107) plugin with access to <b>physics::Model</b> (p. 469)	484
<b>gazebo::physics::ModelState</b>	
Store state information of a <b>physics::Model</b> (p. 469) object	485
<b>gazebo::common::MouseEvent</b>	
Generic description of a mouse event	492
<b>gazebo::rendering::MovableText</b>	
Movable text	495
<b>gazebo::msgs::MsgFactory</b>	
<b>A</b> (p. 107) factory that generates protobuf message based on a string type	501
<b>gazebo::sensors::MultiCameraSensor</b>	
Multiple camera sensor	503
<b>gazebo::physics::MultiRayShape</b>	
Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner	507
<b>gazebo::transport::Node</b>	
<b>A</b> (p. 107) node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics	515
<b>gazebo::common::NodeAnimation</b>	
Node animation	521
<b>gazebo::common::NodeAssignment</b>	
Vertex to node weighted assignment for skeleton animation visualization	525

<b>gazebo::common::NodeTransform</b>	
<b>NodeTransform</b> (p. 526) <b>Skeleton.hh</b> (p. 1053) common/common.hh	526
<b>gazebo::common::NumericAnimation</b>	
<b>A</b> (p. 107) numeric animation	531
<b>gazebo::common::NumericKeyFrame</b>	
<b>A</b> (p. 107) keyframe for a <b>NumericAnimation</b> (p. 531)	533
<b>gazebo::rendering::OrbitViewController</b>	
Orbit view controller	534
<b>sdf::Param</b>	
<b>A</b> (p. 107) parameter class	538
<b>ParamT&lt; T &gt;</b>	544
<b>sdf::ParamT&lt; T &gt;</b>	
Templatized parameter class	544
<b>gazebo::physics::PhysicsEngine</b>	
<b>Base</b> (p. 133) class for a physics engine	547
<b>gazebo::physics::PhysicsFactory</b>	
The physics factory instantiates different physics engines	558
<b>gazebo::common::PID</b>	
Generic <b>PID</b> (p. 559) controller class	559
<b>gazebo::math::Plane</b>	
<b>A</b> (p. 107) plane and related functions	563
<b>gazebo::physics::PlaneShape</b>	
<b>Collision</b> (p. 190) for an infinite plane	566
<b>sdf::Plugin</b>	569
<b>gazebo::PluginT&lt; T &gt;</b>	
<b>A</b> (p. 107) class which all plugins must inherit from	571
<b>gazebo::math::Pose</b>	
Encapsulates a position and rotation in three space	573
<b>gazebo::common::PoseAnimation</b>	
<b>A</b> (p. 107) pose animation	581
<b>gazebo::common::PoseKeyFrame</b>	
<b>A</b> (p. 107) keyframe for a <b>PoseAnimation</b> (p. 581)	584
<b>gazebo::rendering::Projector</b>	
Projects a material onto surface, light a light projector	586
<b>gazebo::transport::Publication</b>	
<b>A</b> (p. 107) publication for a topic	588
<b>gazebo::transport::PublicationTransport</b>	
Transport/transport.hh	593
<b>gazebo::transport::Publisher</b>	
<b>A</b> (p. 107) publisher of messages on a topic	595
<b>gazebo::math::Quaternion</b>	
<b>A</b> (p. 107) quaternion class	598
<b>gazebo::math::Rand</b>	
Random number generator class	612
<b>gazebo::transport::RawCallbackHelper</b>	
Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher	614
<b>gazebo::sensors::RaySensor</b>	
<b>Sensor</b> (p. 672) with one or more rays	616
<b>gazebo::physics::RayShape</b>	
<b>Base</b> (p. 133) class for Ray collision geometry	622
<b>gazebo::rendering::RenderEngine</b>	
Adaptor to Ogre3d	628

<b>gazebo::sensors::RFIDSensor</b>	
<b>Sensor</b> (p. 672) class for RFID type of sensor	633
<b>gazebo::sensors::RFIDTag</b>	
<b>RFIDTag</b> (p. 635) to interact with RFIDTagSensors	635
<b>gazebo::rendering::RFIDTagVisual</b>	
Visualization for RFID tags sensor	637
<b>gazebo::rendering::RFIDVisual</b>	
Visualization for RFID sensor	639
<b>Road</b>	
Used to render a strip of road	640
<b>gazebo::physics::Road</b>	
For building a <b>Road</b> (p. 640) from SDF	640
<b>gazebo::rendering::Road2d</b>	642
<b>gazebo::math::RotationSpline</b>	
<b>Spline</b> (p. 725) for rotations	643
<b>gazebo::rendering::RTShaderSystem</b>	
Implements <b>Ogre</b> (p. 103)'s Run-Time Shader system	647
<b>gazebo::rendering::Scene</b>	
Representation of an entire scene graph	651
<b>gazebo::physics::ScrewJoint&lt; T &gt;</b>	
<b>A</b> (p. 107) screw joint, which has both prismatic and rotational DOFs	666
<b>sdf::SDF</b>	
Base <b>SDF</b> (p. 669) class	669
<b>gazebo::rendering::SelectionObj</b>	
<b>A</b> (p. 107) graphical selection object	670
<b>gazebo::sensors::Sensor</b>	
Base class for sensors	672
<b>SensorFactor</b>	
The sensor factory; the class is just for namespacing purposes	681
<b>gazebo::sensors::SensorFactory</b>	681
<b>gazebo::sensors::SensorManager</b>	
Class to manage and update all sensors	683
<b>gazebo::SensorPlugin</b>	
<b>A</b> (p. 107) plugin with access to physics::Sensor	686
<b>gazebo::Server</b>	688
<b>gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg</b>	
Keeping the CG shader for reference	689
<b>gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL</b>	
Utility class to help with generating shaders for GLSL	691
<b>gazebo::physics::Shape</b>	
<b>Base</b> (p. 133) class for all shapes	693
<b>SingletonT&lt; T &gt;</b>	
Singleton template class	696
<b>gazebo::common::Skeleton</b>	
<b>A</b> (p. 107) skeleton	698
<b>gazebo::common::SkeletonAnimation</b>	
<b>Skeleton</b> (p. 698) animation	705
<b>gazebo::common::SkeletonNode</b>	
<b>A</b> (p. 107) skeleton node	709
<b>gazebo::physics::SliderJoint&lt; T &gt;</b>	
<b>A</b> (p. 107) slider joint	718
<b>gazebo::rendering::GzTerrainMatGen::SM2Profile</b>	
Shader model 2 profile target	720



<b>gazebo::physics::SphereShape</b>	
Sphere collision shape	722
<b>gazebo::math::Spline</b>	
Splines	725
<b>gazebo::physics::State</b>	
<b>State</b> (p. 729) of an entity	729
<b>gazebo::common::STLLoader</b>	
Class used to load STL mesh files	733
<b>gazebo::common::SubMesh</b>	
<b>A</b> (p. 107) child mesh	735
<b>gazebo::transport::SubscribeOptions</b>	
Options for a subscription	744
<b>gazebo::transport::Subscriber</b>	
<b>A</b> (p. 107) subscriber to a topic	746
<b>gazebo::transport::SubscriptionTransport</b>	
Transport/transport.hh	748
<b>gazebo::physics::SurfaceParams</b>	
<b>SurfaceParams</b> (p. 750) defines various Surface contact parameters	750
<b>gazebo::common::SystemPaths</b>	
Functions to handle getting system paths, keeps track of:	754
<b>gazebo::SystemPlugin</b>	
<b>A</b> (p. 107) plugin loaded within the gzserver on startup	759
<b>gazebo::common::Time</b>	
<b>A</b> (p. 107) <b>Time</b> (p. 760) class, can be used to hold wall- or sim-time	760
<b>gazebo::common::Timer</b>	
<b>A</b> (p. 107) timer class, used to time things in real world walltime	781
<b>gazebo::transport::TopicManager</b>	
Manages topics and their subscriptions	782
<b>gazebo::physics::TrajectoryInfo</b>	789
<b>gazebo::physics::TrimeshShape</b>	
Triangle mesh collision shape	790
<b>gazebo::physics::UniversalJoint&lt; T &gt;</b>	
<b>A</b> (p. 107) universal joint	793
<b>urdf2gazebo::URDF2Gazebo</b>	794
<b>gazebo::rendering::UserCamera</b>	
<b>A</b> (p. 107) camera used for user visualization of a scene	796
<b>gazebo::math::Vector2d</b>	
Generic double x, y vector	804
<b>gazebo::math::Vector2i</b>	
Generic integer x, y vector	812
<b>gazebo::math::Vector3</b>	
Generic vector containing 3 elements	821
<b>gazebo::math::Vector4</b>	
Double Generic x, y, z, w vector	834
<b>gazebo::common::Video</b>	
Handle video encoding and decoding using libavcodec	844
<b>gazebo::rendering::VideoVisual</b>	
<b>A</b> (p. 107) visual element that displays a video as a texture	845
<b>gazebo::rendering::ViewController</b>	
Base class for view controllers	847
<b>gazebo::rendering::Visual</b>	
<b>A</b> (p. 107) renderable object	850
<b>gazebo::VisualPlugin</b>	
<b>A</b> (p. 107) plugin loaded within the gzserver on startup	870

---

<b>gazebo::rendering::WindowManager</b>	
Class to mangage render windows . . . . .	871
<b>gazebo::rendering::WireBox</b>	
Draws a wireframe box . . . . .	874
<b>gazebo::physics::World</b>	
The world provides access to all other object within a simulated environment . . . . .	875
<b>gazebo::WorldPlugin</b>	
<b>A</b> (p. 107) plugin with access to <b>physics::World</b> (p. 875) . . . . .	887
<b>gazebo::physics::WorldState</b>	
Store state information of a <b>physics::World</b> (p. 875) object . . . . .	888

# Chapter 7

## File Index

### 7.1 File List

Here is a list of all files with brief descriptions:

<b>Actor.hh</b>	895
<b>Angle.hh</b>	896
<b>Animation.hh</b>	899
<b>ArrowVisual.hh</b>	900
<b>Assert.hh</b>	901
<b>AxisVisual.hh</b>	902
<b>BallJoint.hh</b>	903
<b>Base.hh</b>	904
<b>Box.hh</b>	905
<b>BoxShape.hh</b>	905
<b>BVHLoader.hh</b>	907
<b>CallbackHelper.hh</b>	908
<b>Camera.hh</b>	910
<b>CameraSensor.hh</b>	911
<b>CameraVisual.hh</b>	912
<b>cegui.h</b>	912
<b>ColladaLoader.hh</b>	913
<b>Collision.hh</b>	915
<b>CollisionState.hh</b>	916
<b>Color.hh</b>	918
<b>Common.hh</b>	918
<b>CommonTypes.hh</b>	920
<b>COMVisual.hh</b>	921
<b>Connection.hh</b>	922
<b>ConnectionManager.hh</b>	924
<b>Console.hh</b>	925
<b>Contact.hh</b>	927
<b>ContactManager.hh</b>	928
<b>ContactSensor.hh</b>	929
<b>ContactVisual.hh</b>	930
<b>Conversions.hh</b>	931
<b>Converter.hh</b>	931
<b>CylinderShape.hh</b>	932
<b>DepthCamera.hh</b>	933

DepthCameraSensor.hh	934
Diagnostics.hh	935
DynamicLines.hh	937
DynamicRenderable.hh	937
Entity.hh	938
Event.hh	940
Events.hh	940
Exception.hh	941
FPSViewController.hh	943
gazebo.hh	944
gazebo_core.hh	945
GazeboGenerator.hh	945
GpuLaser.hh	946
GpuRaySensor.hh	947
Grid.hh	948
Gripper.hh	948
GUIOverlay.hh	950
Heightmap.hh	950
HeightmapShape.hh	951
Helpers.hh	952
Hinge2Joint.hh	954
HingeJoint.hh	956
Image.hh	957
ImuSensor.hh	958
Inertial.hh	959
IOManager.hh	960
Joint.hh	961
JointController.hh	962
JointState.hh	963
JointVisual.hh	964
JointWrench.hh	965
KeyFrame.hh	966
LaserVisual.hh	967
Light.hh	968
Link.hh	969
LinkState.hh	970
LogPlay.hh	972
LogRecord.hh	973
mainpage.html	975
MapShape.hh	975
Master.hh	976
common/Material.hh	977
rendering/Material.hh	979
MathTypes.hh	
Forward declarations for the math classes	979
Matrix3.hh	980
Matrix4.hh	980
Mesh.hh	981
MeshCSG.hh	983
MeshLoader.hh	984
MeshManager.hh	985
Model.hh	986
ModelDatabase.hh	988
ModelState.hh	989

<b>MouseEvent.hh</b>	991
<b>MovableText.hh</b>	992
<b>MsgFactory.hh</b>	992
<b>msgs.hh</b>	993
<b>MultiCameraSensor.hh</b>	996
<b>MultiRayShape.hh</b>	996
<b>Node.hh</b>	998
<b>ogre_gazebo.h</b>	999
<b>OrbitViewController.hh</b>	1000
<b>Param.hh</b>	1000
<b>parser.hh</b>	1002
<b>parser_urdf.hh</b>	1003
<b>Physics.hh</b>	1004
<b>PhysicsEngine.hh</b>	1005
<b>PhysicsFactory.hh</b>	1007
<b>PhysicsTypes.hh</b>	
Default namespace for gazebo	1008
<b>PID.hh</b>	1011
<b>Plane.hh</b>	1012
<b>PlaneShape.hh</b>	1013
<b>common/Plugin.hh</b>	1014
<b>sdf/interface/Plugin.hh</b>	1018
<b>Pose.hh</b>	1018
<b>Projector.hh</b>	1019
<b>Publication.hh</b>	1020
<b>PublicationTransport.hh</b>	1021
<b>Publisher.hh</b>	1023
<b>Quaternion.hh</b>	1025
<b>Rand.hh</b>	1026
<b>RaySensor.hh</b>	1027
<b>RayShape.hh</b>	1028
<b>RenderEngine.hh</b>	1029
<b>RenderEvents.hh</b>	1030
<b>Rendering.hh</b>	1031
<b>RenderTypes.hh</b>	1032
<b>RFIDSensor.hh</b>	1034
<b>RFIDTag.hh</b>	1034
<b>RFIDTagVisual.hh</b>	1035
<b>RFIDVisual.hh</b>	1036
<b>Road.hh</b>	1036
<b>Road2d.hh</b>	1038
<b>RotationSpline.hh</b>	1038
<b>RTShaderSystem.hh</b>	1039
<b>Scene.hh</b>	1040
<b>ScrewJoint.hh</b>	1041
<b>sdf.hh</b>	1042
<b>SDF.hh</b>	1043
<b>SelectionObj.hh</b>	1044
<b>Sensor.hh</b>	1045
<b>SensorFactory.hh</b>	1046
<b>SensorManager.hh</b>	1047
<b>Sensors.hh</b>	1048
<b>SensorTypes.hh</b>	
Forward declarations and typedefs for sensors	1049

<b>Server.hh</b>	1051
<b>Shape.hh</b>	1052
<b>SingletonT.hh</b>	1053
<b>Skeleton.hh</b>	1053
<b>SkeletonAnimation.hh</b>	1055
<b>SliderJoint.hh</b>	1056
<b>SphereShape.hh</b>	1058
<b>Spline.hh</b>	1059
<b>State.hh</b>	1060
<b>STLLoader.hh</b>	1061
<b>SubscribeOptions.hh</b>	1063
<b>Subscriber.hh</b>	1064
<b>SubscriptionTransport.hh</b>	1066
<b>SurfaceParams.hh</b>	1067
<b>SystemPaths.hh</b>	1069
<b>Time.hh</b>	1070
<b>Timer.hh</b>	1071
<b>TopicManager.hh</b>	1072
<b>Transport.hh</b>	1074
<b>TransportTypes.hh</b>	
Forward declarations for transport	1076
<b>TrimeshShape.hh</b>	1078
<b>UniversalJoint.hh</b>	1079
<b>UserCamera.hh</b>	1080
<b>Vector2d.hh</b>	1080
<b>Vector2i.hh</b>	1081
<b>Vector3.hh</b>	1082
<b>Vector4.hh</b>	1083
<b>Video.hh</b>	1085
<b>VideoVisual.hh</b>	1086
<b>ViewController.hh</b>	1087
<b>Visual.hh</b>	1088
<b>WindowManager.hh</b>	1089
<b>WireBox.hh</b>	1089
<b>World.hh</b>	1090
<b>WorldState.hh</b>	1092

# 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::DiagnosticManager**  
*A (p. 107) diagnostic manager class.*
- class **gazebo::common::DiagnosticTimer**  
*A (p. 107) timer designed for diagnostics.*
- 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. 107) *key frame in an animation.*
  - class **gazebo::common::Material**

*Encapsulates description of a material.*
  - class **gazebo::common::Mesh**

**A** (p. 107) *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. 107) *plugin with access to **physics::Model** (p. 469).*
  - class **gazebo::common::MouseEvent**

*Generic description of a mouse event.*
  - class **gazebo::common::NodeAnimation**

*Node animation.*
  - struct **gazebo::common::NodeAssignment**

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

**NodeTransform** (p. 526) **Skeleton.hh** (p. 1053) *common/common.hh*
  - class **gazebo::common::NumericAnimation**

**A** (p. 107) *numeric animation.*
  - class **gazebo::common::NumericKeyFrame**

**A** (p. 107) *keyframe for a **NumericAnimation** (p. 531).*
  - class **gazebo::common::PID**

*Generic **PID** (p. 559) controller class.*
  - class **gazebo::PluginT < T >**

**A** (p. 107) *class which all plugins must inherit from.*
  - class **gazebo::common::PoseAnimation**

**A** (p. 107) *pose animation.*
  - class **gazebo::common::PoseKeyFrame**

**A** (p. 107) *keyframe for a **PoseAnimation** (p. 581).*
  - class **gazebo::SensorPlugin**

**A** (p. 107) *plugin with access to **physics::Sensor**.*
  - class **SingletonT < T >**

*Singleton template class.*
  - class **gazebo::common::Skeleton**

**A** (p. 107) *skeleton.*
  - class **gazebo::common::SkeletonAnimation**

**Skeleton** (p. 698) *animation.*



- class **gazebo::common::SkeletonNode**  
A (p. 107) skeleton node.
- class **gazebo::common::STLLoader**  
Class used to load STL mesh files.
- class **gazebo::common::SubMesh**  
A (p. 107) child mesh.
- class **gazebo::common::SystemPaths**  
Functions to handle getting system paths, keeps track of:
- class **gazebo::SystemPlugin**  
A (p. 107) plugin loaded within the gzserver on startup.
- class **gazebo::common::Time**  
A (p. 107) **Time** (p. 760) class, can be used to hold wall- or sim-time.
- class **gazebo::common::Timer**  
A (p. 107) 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. 107) plugin loaded within the gzserver on startup.
- class **gazebo::WorldPlugin**  
A (p. 107) plugin with access to **physics::World** (p. 875).

## Macros

- #define **DIAG\_TIMER**(name) DiagnosticManager::Instance()->CreateTimer(name);  
Create an instance of common::DiagnosticManager.
- #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.

## Typedefs

- typedef DiagnosticTimer \* **gazebo::common::DiagnosticTimerPtr**

## 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. 754)*
- 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. 754)*
- std::string **gazebo::common::find\_file\_path** (const std::string &\_file)  
*search for a file in **common::SystemPaths** (p. 754)*
- 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)  
*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::ostream & **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 DIAG\_TIMER( name ) DiagnosticManager::Instance()->CreateTimer(name);**

Create an instance of common::DiagnosticManager.

8.1.2.2 **#define gzclr\_end "\033[0m"**

End marker.

8.1.2.3 **#define gzclr\_start( clr ) "\033[1;33m"**

Start marker.

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

Output a debug message.

8.1.2.5 **#define gzerr**

#### Value:

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

Output an error message.

Referenced by gazebo::transport::Connection::AsyncRead(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::physics::ScrewJoint< T >::Load(), sdf::ParamT< std::string >::Set(), and sdf::ParamT< std::string >::Update().

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

Output a message to a log file.

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

Output a message.

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

### 8.1.2.8 #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().

### 8.1.2.9 #define gzwarn

**Value:**

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

Output a warning message.

## 8.1.3 Typedef Documentation

### 8.1.3.1 typedef DiagnosticTimer\* gazebo::common::DiagnosticTimerPtr

## 8.1.4 Enumeration Type Documentation

### 8.1.4.1 enum gazebo::PluginType

Used to specify the type of plugin.

Enumerator:

**WORLD\_PLUGIN** A (p. 107) World plugin.

**MODEL\_PLUGIN** A (p. 107) Model plugin.

**SENSOR\_PLUGIN** A (p. 107) Sensor plugin.

**SYSTEM\_PLUGIN** A (p. 107) System plugin.

**VISUAL\_PLUGIN** A (p. 107) Visual plugin.

## 8.1.5 Function Documentation

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

constructor

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

add path prefix to **common::SystemPaths** (p. 754)

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

Use this to output an error to the terminal.

#### Parameters

<code>in</code>	<code><i>_lbl</i></code>	Text label
<code>in</code>	<code><i>_file</i></code>	File containing the error
<code>in</code>	<code><i>_line</i></code>	Line containing the error
<code>in</code>	<code><i>_color</i></code>	<b>Color</b> (p. 203) to make the label

#### Returns

Reference to an output stream

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

Use this to output a colored message to the terminal.

#### Parameters

<code>in</code>	<code><i>_lbl</i></code>	Text label
<code>in</code>	<code><i>_color</i></code>	<b>Color</b> (p. 203) to make the label

#### Returns

Reference to an output stream

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

Download all dependencies for a give model path.

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

#### Parameters

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

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

search for file in **common::SystemPaths** (p. 754)

## Parameters

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

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

search for a file in **common::SystemPaths** (p. 754)

## Parameters

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

## Returns

The path containing the file

8.1.5.8 `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.

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

Deprecated.

## See Also

**ModelDatabase::GetModelConfig** (p. 34)

**ModelDatabase::GetDBConfig** (p. 34)

8.1.5.10 `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.

8.1.5.11 `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

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

**Returns**

The full path and filename to the SDF file

#### 8.1.5.12 `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 `models://gazebo_models`

**Parameters**

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

**Returns**

the model's name.

#### 8.1.5.13 `std::string gazebo::common::ModelDatabase::GetModelPath ( const std::string & _uri )`

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.  
param[in] `_uri` the model uri

**Returns**

path to a model directory

#### 8.1.5.14 `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. 482) to download the model list.

**Returns**

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

#### 8.1.5.15 `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.
-----------------	--------------------	---

#### 8.1.5.16 `std::string gazebo::common::ModelDatabase::GetURI ( )`

Returns the the global model database URI.

##### Returns

the URI.

#### 8.1.5.17 `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.

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

Load the message parameters.

#### 8.1.5.19 `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

#### 8.1.5.20 `void gazebo::common::Console::SetQuiet ( bool _q )`

Set quiet output.

##### Parameters

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



## 8.1.6 Variable Documentation

### 8.1.6.1 `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. 353).

## 8.2 Events

### Namespaces

- namespace **gazebo::event**  
*Event* (p. 285) namespace.

### Classes

- class **gazebo::event::Connection**  
*A* (p. 107) class that encapsulates a connection.
- class **gazebo::event::Event**  
*Base class for all events.*
- class **gazebo::event::Events**  
*An Event* (p. 285) class to get notifications for simulator events.
- class **gazebo::event::EventT< T >**  
*A* (p. 107) class for event processing.

### Functions

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

#### 8.2.1 Detailed Description

#### 8.2.2 Function Documentation

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

Destructor.

Destructor. Deletes all the associated connections.

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

Connect a callback to this event.

Adds a connection.

## Parameters

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

## Returns

A (p. 107) **Connection** (p. 216) object, which will automatically call Disconnect when it goes out of scope

## Parameters

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

Referenced by `gazebo::event::Events::ConnectAddEntity()`, `gazebo::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::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. 285).

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

Disconnect a callback to this event.

Removes a connection.

## Parameters

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

Implements **gazebo::event::Event** (p. 287).

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::DisconnectNewImageFrame()`, `gazebo::rendering::GpuLaser::DisconnectNewLaserFrame()`, `gazebo::physics::MultiRayShape`

::DisconnectNewLaserScans(), gazebo::rendering::DepthCamera::DisconnectNewRGBPointCloud(), gazebo::event::Events::DisconnectPause(), gazebo::event::Events::DisconnectPostRender(), gazebo::event::Events::DisconnectPreRender(), gazebo::rendering::Events::DisconnectRemoveScene(), gazebo::event::Events::DisconnectRender(), gazebo::event::Events::DisconnectSetSelectedEntity(), gazebo::transport::Connection::DisconnectShutdown(), gazebo::event::Events::DisconnectStep(), gazebo::event::Events::DisconnectStop(), gazebo::sensors::Sensor::DisconnectUpdated(), gazebo::event::Events::DisconnectWorldCreated(), gazebo::event::Events::DisconnectWorldUpdateEnd(), and gazebo::event::Events::DisconnectWorldUpdateStart().

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

Disconnect a callback to this event.

Removes a connection.

#### Parameters

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

Implements **gazebo::event::Event** (p. 287).

## 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. 107) class enables GPU based mesh model / skeleton scriptable animation.
- class **gazebo::physics::BallJoint**< T >  
*Base* (p. 133) class for a ball joint.
- class **gazebo::physics::Base**  
*Base* (p. 133) class for most physics classes.
- class **gazebo::physics::BoxShape**  
*Box geometry primitive.*
- class **gazebo::physics::Collision**  
*Base* (p. 133) class for all collision entities.
- class **gazebo::physics::CollisionState**  
*Store state information of a **physics::Collision** (p. 190) object.*
- class **gazebo::physics::Contact**  
*A* (p. 107) 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. 133) class for all physics objects in Gazebo.
- class **gazebo::physics::Gripper**  
*A* (p. 107) gripper abstraction.
- class **gazebo::physics::HeightmapShape**  
*HeightmapShape* (p. 344) collision shape builds a heightmap from an image.
- class **gazebo::physics::Hinge2Joint**< T >  
*A* (p. 107) two axis hinge joint.
- class **gazebo::physics::HingeJoint**< T >  
*A* (p. 107) single axis hinge joint.
- class **gazebo::physics::Inertial**  
*A* (p. 107) class for inertial information about a link.
- class **gazebo::physics::Joint**  
*Base* (p. 133) class for all joints.
- class **gazebo::physics::JointController**

- A* (p. 107) class for manipulating *physics::Joint* (p. 371).
- class **gazebo::physics::JointState**
  - keeps track of state of a physics::Joint* (p. 371)
- class **gazebo::physics::JointWrench**
  - Wrench information from a joint.*
- class **gazebo::physics::Link**
  - Link* (p. 404) 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. 404) object.
- class **Logplay**
  - Open and playback log files that were recorded using LogRecord.*
- class **gazebo::common::LogPlay**
- class **gazebo::physics::Model**
  - A* (p. 107) model is a collection of links, joints, and plugins.
- class **gazebo::physics::ModelState**
  - Store state information of a physics::Model* (p. 469) 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. 133) class for a physics engine.
- class **gazebo::physics::PhysicsFactory**
  - The physics factory instantiates different physics engines.*
- class **gazebo::physics::PlaneShape**
  - Collision* (p. 190) for an infinite plane.
- class **gazebo::physics::RayShape**
  - Base* (p. 133) class for Ray collision geometry.
- class **gazebo::physics::Road**
  - for building a Road* (p. 640) from SDF
- class **gazebo::physics::ScrewJoint** < T >
  - A* (p. 107) screw joint, which has both prismatic and rotational DOFs.
- class **gazebo::physics::Shape**
  - Base* (p. 133) class for all shapes.
- class **gazebo::physics::SliderJoint** < T >
  - A* (p. 107) slider joint.
- class **gazebo::physics::SphereShape**
  - Sphere collision shape.*
- class **gazebo::physics::State**
  - State* (p. 729) of an entity.
- class **gazebo::physics::SurfaceParams**
  - SurfaceParams* (p. 750) defines various Surface contact parameters.
- class **gazebo::physics::TrimeshShape**
  - Triangle mesh collision shape.*
- class **gazebo::physics::UniversalJoint** < T >
  - A* (p. 107) 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. 875) 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. 75).*
- 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. 759)'s and call **gazebo::transport::init** (p. 76).*
- void **gazebo::physics::load\_world** (WorldPtr \_world, sdf::ElementPtr \_sdf)  
*Load world from **sdf::Element** (p. 266) pointer.*
- void **gazebo::physics::load\_worlds** (sdf::ElementPtr \_sdf)  
*load multiple worlds from single **sdf::Element** (p. 266) pointer*
- void **gazebo::physics::pause\_world** (WorldPtr \_world, bool \_pause)  
*Pause world by calling **World::SetPaused** (p. 886).*
- 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. 885) 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. 886) 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. 75).

#### 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>	<b>World</b> (p. 875) 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. 759)'s and call **gazebo::transport::init** (p. 76).

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

Load world from **sdf::Element** (p. 266) 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. 266) 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. 886).

## Parameters

in	<code>_world</code>	<b>World</b> (p. 875) 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. 885) given a pointer to it.

**Parameters**

in	<i>_world</i>	<b>World</b> (p. 875) 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. 886) given a pointer to it.

**Parameters**

in	<i>_world</i>	<b>World</b> (p. 875) 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. 107) 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. 107) 3x3 matrix class.*
- class **gazebo::math::Matrix4**  
***A** (p. 107) 3x3 matrix class.*
- class **gazebo::math::Plane**  
***A** (p. 107) plane and related functions.*
- class **gazebo::math::Pose**  
*Encapsulates a position and rotation in three space.*
- class **gazebo::math::Quaternion**  
***A** (p. 107) quaternion class.*
- class **gazebo::math::Rand**  
*Random number generator class.*
- class **gazebo::math::RotationSpline**  
***Spline** (p. 725) 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. 821) 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. 107) 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. 87).*
- class **gazebo::msgs::MsgFactory**  
*A (p. 107) 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. 821) to a **msgs::Vector3d**.*
- **msgs::Quaternion gazebo::msgs::Convert** (const math::Quaternion &\_q)  
*Convert a **math::Quaternion** (p. 598) to a **msgs::Quaternion**.*
- **msgs::Pose gazebo::msgs::Convert** (const math::Pose &\_p)  
*Convert a **math::Pose** (p. 573) to a **msgs::Pose**.*
- **msgs::Color gazebo::msgs::Convert** (const common::Color &\_c)  
*Convert a **common::Color** (p. 203) to a **msgs::Color**.*
- **msgs::Time gazebo::msgs::Convert** (const common::Time &\_t)  
*Convert a **common::Time** (p. 760) to a **msgs::Time**.*
- **msgs::PlaneGeom gazebo::msgs::Convert** (const math::Plane &\_p)  
*Convert a **math::Plane** (p. 563) 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. 598).*
- **math::Pose gazebo::msgs::Convert** (const msgs::Pose &\_p)  
*Convert a **msgs::Pose** to a **math::Pose** (p. 573).*
- **common::Color gazebo::msgs::Convert** (const msgs::Color &\_c)  
*Convert a **msgs::Color** to a **common::Color** (p. 203).*
- **common::Time gazebo::msgs::Convert** (const msgs::Time &\_t)  
*Convert a **msgs::Time** to a **common::Time** (p. 760).*
- **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. 352).*
- `void gazebo::msgs::Set` (`msgs::Image *_msg, const common::Image &_i`)  
*Set a `msgs::Image` from a `common::Image` (p. 352).*
- `void gazebo::msgs::Set` (`msgs::Vector3d *_pt, const math::Vector3 &_v`)  
*Set a `msgs::Vector3d` from a `math::Vector3` (p. 821).*
- `void gazebo::msgs::Set` (`msgs::Vector2d *_pt, const math::Vector2d &_v`)  
*Set a `msgs::Vector2d` from a `math::Vector3` (p. 821).*
- `void gazebo::msgs::Set` (`msgs::Quaternion *_q, const math::Quaternion &_v`)  
*Set a `msgs::Quaternion` from a `math::Quaternion` (p. 598).*
- `void gazebo::msgs::Set` (`msgs::Pose *_p, const math::Pose &_v`)  
*Set a `msgs::Pose` from a `math::Pose` (p. 573).*
- `void gazebo::msgs::Set` (`msgs::Color *_c, const common::Color &_v`)  
*Set a `msgs::Color` from a `common::Color` (p. 203).*
- `void gazebo::msgs::Set` (`msgs::Time *_t, const common::Time &_v`)  
*Set a `msgs::Time` from a `common::Time` (p. 760).*
- `void gazebo::msgs::Set` (`msgs::PlaneGeom *_p, const math::Plane &_v`)  
*Set a `msgs::Plane` from a `math::Plane` (p. 563).*
- `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. 821) to a msgs::Vector3d.

#### Parameters

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

#### Returns

**A** (p. 107) msgs::Vector3d object

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

Convert a **math::Quaternion** (p. 598) to a msgs::Quaternion.

#### Parameters

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

#### Returns

**A** (p. 107) msgs::Quaternion object

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

Convert a **math::Pose** (p. 573) to a `msgs::Pose`.

## Parameters

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

## Returns

**A** (p. 107) `msgs::Pose` object

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

Convert a **common::Color** (p. 203) to a `msgs::Color`.

## Parameters

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

## Returns

**A** (p. 107) `msgs::Color` object

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

Convert a **common::Time** (p. 760) to a `msgs::Time`.

## Parameters

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

## Returns

**A** (p. 107) `msgs::Time` object

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

Convert a **math::Plane** (p. 563) to a `msgs::PlaneGeom`.

## Parameters

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

## Returns

**A** (p. 107) `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. 107) **math::Vector3** (p. 821) object

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

Convert a `msgs::Quaternion` to a **math::Quaternion** (p. 598).

#### Parameters

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

#### Returns

**A** (p. 107) **math::Quaternion** (p. 598) object

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

Convert a `msgs::Pose` to a **math::Pose** (p. 573).

#### Parameters

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

#### Returns

**A** (p. 107) **math::Pose** (p. 573) object

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

Convert a `msgs::Color` to a **common::Color** (p. 203).

#### Parameters

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

#### Returns

**A** (p. 107) **common::Color** (p. 203) object

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

Convert a `msgs::Time` to a **common::Time** (p. 760).

## Parameters

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

## Returns

**A** (p. 107) **common::Time** (p. 760) 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. 107) `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. 107) 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

<i>in</i>	<i>_message</i>	<b>A</b> (p. 107) google protobuf message
-----------	-----------------	---

#### Returns

**A** (p. 107) 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

<i>in</i>	<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

<i>in</i>	<i>_message</i>	Message to initialize
<i>in</i>	<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

<i>in</i>	<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. 352).

## Parameters

out	<code>_img</code>	The <b><code>common::Image</code></b> (p. 352) 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. 352).

## Parameters

out	<code>_msg</code>	<b>A</b> (p. 107) <code>msgs::Image</code> pointer
in	<code>_i</code>	<b>A</b> (p. 107) <b><code>common::Image</code></b> (p. 352) 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. 821).

## Parameters

out	<code>_pt</code>	<b>A</b> (p. 107) <code>msgs::Vector3d</code> pointer
in	<code>_v</code>	<b>A</b> (p. 107) <b><code>math::Vector3</code></b> (p. 821) 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. 821).

## Parameters

out	<code>_pt</code>	<b>A</b> (p. 107) <code>msgs::Vector2d</code> pointer
in	<code>_v</code>	<b>A</b> (p. 107) <b><code>math::Vector2d</code></b> (p. 804) 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. 598).

## Parameters

out	<code>_q</code>	<b>A</b> (p. 107) <code>msgs::Quaternion</code> pointer
in	<code>_v</code>	<b>A</b> (p. 107) <code>math::Quaternion</code> (p. 598) 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. 573).

## Parameters

out	<code>_p</code>	<b>A</b> (p. 107) <code>msgs::Pose</code> pointer
in	<code>_v</code>	<b>A</b> (p. 107) <code>math::Pose</code> (p. 573) 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. 203).

## Parameters

out	<code>_p</code>	<b>A</b> (p. 107) <code>msgs::Color</code> pointer
in	<code>_v</code>	<b>A</b> (p. 107) <code>common::Color</code> (p. 203) 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. 760).

## Parameters

out	<code>_p</code>	<b>A</b> (p. 107) <code>msgs::Time</code> pointer
in	<code>_v</code>	<b>A</b> (p. 107) <code>common::Time</code> (p. 760) 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. 563).

## Parameters

out	<code>_p</code>	<b>A</b> (p. 107) <code>msgs::Plane</code> pointer
in	<code>_v</code>	<b>A</b> (p. 107) <code>math::Plane</code> (p. 563) 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. 107) 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. 107) 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. 240) *Conversions.hh* (p. 931) *rendering/Conversions.hh* (p. 931).
- 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. 107) 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. 107) 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. 103)'s Run-Time Shader system.*
- class **gazebo::rendering::Scene**  
*Representation of an entire scene graph.*
- class **gazebo::rendering::SelectionObj**  
A (p. 107) graphical selection object.
- class **gazebo::rendering::UserCamera**  
A (p. 107) camera used for user visualization of a scene.
- class **gazebo::rendering::VideoVisual**  
A (p. 107) visual element that displays a video as a texture.
- class **gazebo::rendering::ViewController**  
*Base class for view controllers.*
- class **gazebo::rendering::Visual**  
A (p. 107) 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. 651) 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. 651) 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. 651) by name*

## 8.6.1 Detailed Description

**A** (p. 107) 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. 651) 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. 651) 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. 651) by name

## Parameters

<code>in</code>	<code>_name</code>	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. 669) Element (p. 266) class.*
- class **urdf2gazebo::GazeboExtension**
- class **sdf::SDF**  
*Base SDF (p. 669) 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. 107) 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. 672) with one or more rays.*
- class **gazebo::sensors::RFIDSensor**  
*Sensor (p. 672) class for RFID type of sensor.*
- class **gazebo::sensors::RFIDTag**  
*RFIDTag (p. 635) 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 ()`  
*Run sensor generation continuously. This is a blocking call.*
- `void gazebo::sensors::run_once (bool _force=false)`  
*Run the sensor generation one step.*
- `void gazebo::sensors::stop ()`  
*Stop the sensor generation loop.*

### 8.8.1 Detailed Description

**A** (p. 107) 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 <i>_name</i> = world_name::model_name::link_name::sensor_name. You may use the unscoped sensor name if that name is unique within the entire simulation. If the name is not unique a NULL pointer is returned.
-----------	--------------	---

#### Returns

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

8.8.3.4 `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 ( )**

Run sensor generation continuously. This is a blocking call.

**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::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. 107) 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. 107) node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.*
- class **gazebo::transport::Publication**  
*A (p. 107) publication for a topic.*
- class **gazebo::transport::PublicationTransport**  
*transport/transport.hh*
- class **gazebo::transport::Publisher**  
*A (p. 107) 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. 107) 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. 153)*

## 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. 153)

### 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. 107) 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. 107) 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 <code>GAZEBO_MASTER_URI</code> env var.
<code>in</code>	<code>_master_port</code>	The port of the master. Leave empty to use pull address from the <code>GAZEBO_MASTER_URI</code> 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

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

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

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

8.9.3.14 `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.



## 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. 285) namespace.*
- namespace **math**  
*Math namespace.*
- namespace **msgs**  
*Messages namespace.*
- namespace **physics**  
*namespace for physics*
- namespace **rendering**  
*Rendering namespace.*
- namespace **sensors**  
*Sensors namespace.*
- namespace **transport**

#### Classes

- class **Master**  
*A (p. 107) 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. 107) plugin with access to **physics::Model** (p. 469).*
- class **PluginT**

*A (p. 107) class which all plugins must inherit from.*

- class **SensorPlugin**

*A (p. 107) plugin with access to `physics::Sensor`.*

- class **Server**

- class **SystemPlugin**

*A (p. 107) plugin loaded within the gzserver on startup.*

- class **VisualPlugin**

*A (p. 107) plugin loaded within the gzserver on startup.*

- class **WorldPlugin**

*A (p. 107) plugin with access to `physics::World` (p. 875).*

## 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.

## 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 **DiagnosticManager**
  - A (p. 107) diagnostic manager class.*
- class **DiagnosticTimer**
  - A (p. 107) timer designed for diagnostics.*
- 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. 107) key frame in an animation.*
- class **LogPlay**
- class **LogRecord**
  - addtogroup gazebo\_common*
- class **Material**
  - Encapsulates description of a material.*
- class **Mesh**
  - A (p. 107) 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. 526) Skeleton.hh (p. 1053) common/common.hh*
- class **NumericAnimation**
  - A (p. 107) numeric animation.*
- class **NumericKeyFrame**
  - A (p. 107) keyframe for a **NumericAnimation** (p. 531).*
- class **PID**
  - Generic **PID** (p. 559) controller class.*
- class **PoseAnimation**
  - A (p. 107) pose animation.*
- class **PoseKeyFrame**

- *A (p. 107) keyframe for a **PoseAnimation** (p. 581).*
- class **Skeleton**
  - *A (p. 107) skeleton.*
- class **SkeletonAnimation**
  - ***Skeleton** (p. 698) animation.*
- class **SkeletonNode**
  - *A (p. 107) skeleton node.*
- class **STLLoader**
  - *Class used to load STL mesh files.*
- class **SubMesh**
  - *A (p. 107) child mesh.*
- class **SystemPaths**
  - *Functions to handle getting system paths, keeps track of:*
- class **Time**
  - *A (p. 107) **Time** (p. 760) class, can be used to hold wall- or sim-time.*
- class **Timer**
  - *A (p. 107) timer class, used to time things in real world walltime.*
- 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. 754)*
- std::string **find\_file** (const std::string &\_file, bool \_searchLocalPath=true)  
*search for file in **common::SystemPaths** (p. 754)*
- std::string **find\_file\_path** (const std::string &\_file)  
*search for a file in **common::SystemPaths** (p. 754)*

## 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 std::map<unsigned int, SkeletonNode\*> gazebo::common::NodeMap

9.3.2.3 typedef std::map<unsigned int, SkeletonNode\*>::iterator gazebo::common::NodeMapItr

9.3.2.4 typedef NumericAnimation\* gazebo::common::NumericAnimationPtr

9.3.2.5 typedef std::vector<common::Param\*> gazebo::common::Param\_V

9.3.2.6 typedef PoseAnimation\* gazebo::common::PoseAnimationPtr

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

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

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

9.3.2.10 typedef std::map<std::string, std::string> gazebo::common::StrStr\_M

## 9.4 gazebo::event Namespace Reference

**Event** (p. 285) namespace.

### Classes

- class **Connection**  
*A (p. 107) class that encapsulates a connection.*
- class **Event**  
*Base class for all events.*
- class **Events**  
*An **Event** (p. 285) class to get notifications for simulator events.*
- class **EventT**  
*A (p. 107) class for event processing.*



## Typedefs

- typedef std::vector  
< **ConnectionPtr** > **Connection\_V**
- typedef **Connection** \* **ConnectionPtr**

### 9.4.1 Detailed Description

**Event** (p. 285) 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. 107) 3x3 matrix class.
- class **Matrix4**  
**A** (p. 107) 3x3 matrix class.
- class **Plane**  
**A** (p. 107) plane and related functions.
- class **Pose**  
*Encapsulates a position and rotation in three space.*
- class **Quaternion**  
**A** (p. 107) quaternion class.
- class **Rand**  
*Random number generator class.*
- class **RotationSpline**  
**Spline** (p. 725) for rotations.
- class **Spline**  
*Splines.*
- class **Vector2d**  
*Generic double x, y vector.*
- class **Vector2i**  
*Generic integer x, y vector.*
- class **Vector3**

The **Vector3** (p. 821) 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. 107) 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. 821) to a msgs::Vector3d.*
- msgs::Quaternion **Convert** (const **math::Quaternion** &\_q)  
*Convert a **math::Quaternion** (p. 598) to a msgs::Quaternion.*
- msgs::Pose **Convert** (const **math::Pose** &\_p)  
*Convert a **math::Pose** (p. 573) to a msgs::Pose.*
- msgs::Color **Convert** (const **common::Color** &\_c)  
*Convert a **common::Color** (p. 203) to a msgs::Color.*
- msgs::Time **Convert** (const **common::Time** &\_t)  
*Convert a **common::Time** (p. 760) to a msgs::Time.*
- msgs::PlaneGeom **Convert** (const **math::Plane** &\_p)  
*Convert a **math::Plane** (p. 563) 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. 598).*
- **math::Pose Convert** (const msgs::Pose &\_p)  
*Convert a msgs::Pose to a **math::Pose** (p. 573).*
- **common::Color Convert** (const msgs::Color &\_c)  
*Convert a msgs::Color to a **common::Color** (p. 203).*
- **common::Time Convert** (const msgs::Time &\_t)  
*Convert a msgs::Time to a **common::Time** (p. 760).*
- **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. 352).*

- void **Set** (msgs::Image \*\_msg, const **common::Image** &\_i)  
*Set a msgs::Image from a **common::Image** (p. 352).*
- void **Set** (msgs::Vector3d \*\_pt, const **math::Vector3** &\_v)  
*Set a msgs::Vector3d from a **math::Vector3** (p. 821).*
- void **Set** (msgs::Vector2d \*\_pt, const **math::Vector2d** &\_v)  
*Set a msgs::Vector2d from a **math::Vector3** (p. 821).*
- void **Set** (msgs::Quaternion \*\_q, const **math::Quaternion** &\_v)  
*Set a msgs::Quaternion from a **math::Quaternion** (p. 598).*
- void **Set** (msgs::Pose \*\_p, const **math::Pose** &\_v)  
*Set a msgs::Pose from a **math::Pose** (p. 573).*
- void **Set** (msgs::Color \*\_c, const **common::Color** &\_v)  
*Set a msgs::Color from a **common::Color** (p. 203).*
- void **Set** (msgs::Time \*\_t, const **common::Time** &\_v)  
*Set a msgs::Time from a **common::Time** (p. 760).*
- void **Set** (msgs::PlaneGeom \*\_p, const **math::Plane** &\_v)  
*Set a msgs::Plane from a **math::Plane** (p. 563).*
- 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. 107) class enables GPU based mesh model / skeleton scriptable animation.*
- class **BallJoint**  
***Base** (p. 133) class for a ball joint.*
- class **Base**  
***Base** (p. 133) class for most physics classes.*
- class **BoxShape**

- Box geometry primitive.*
- class **Collision**
  - Base (p. 133) class for all collision entities.*
- class **CollisionState**
  - Store state information of a **physics::Collision** (p. 190) object.*
- class **Contact**
  - A (p. 107) 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. 133) class for all physics objects in Gazebo.*
- class **Gripper**
  - A (p. 107) gripper abstraction.*
- class **HeightmapShape**
  - HeightmapShape (p. 344) collision shape builds a heightmap from an image.*
- class **Hinge2Joint**
  - A (p. 107) two axis hinge joint.*
- class **HingeJoint**
  - A (p. 107) single axis hinge joint.*
- class **Inertial**
  - A (p. 107) class for inertial information about a link.*
- class **Joint**
  - Base (p. 133) class for all joints.*
- class **JointController**
  - A (p. 107) class for manipulating **physics::Joint** (p. 371).*
- class **JointState**
  - keeps track of state of a **physics::Joint** (p. 371)*
- class **JointWrench**
  - Wrench information from a joint.*
- class **Link**
  - Link (p. 404) 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. 404) object.*
- class **Model**
  - A (p. 107) model is a collection of links, joints, and plugins.*
- class **ModelState**
  - Store state information of a **physics::Model** (p. 469) object.*
- class **MultiRayShape**
  - Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.*
- class **PhysicsEngine**
  - Base (p. 133) class for a physics engine.*
- class **PhysicsFactory**
  - The physics factory instantiates different physics engines.*
- class **PlaneShape**

- Collision* (p. 190) for an infinite plane.
- class **RayShape**
  - Base* (p. 133) class for Ray collision geometry.
- class **Road**
  - for building a **Road** (p. 640) from SDF
- class **ScrewJoint**
  - A** (p. 107) screw joint, which has both prismatic and rotational DOFs.
- class **Shape**
  - Base* (p. 133) class for all shapes.
- class **SliderJoint**
  - A** (p. 107) slider joint.
- class **SphereShape**
  - Sphere collision shape.*
- class **State**
  - State* (p. 729) of an entity.
- class **SurfaceParams**
  - SurfaceParams* (p. 750) defines various Surface contact parameters.
- struct **TrajectoryInfo**
- class **TrimeshShape**
  - Triangle mesh collision shape.*
- class **UniversalJoint**
  - A** (p. 107) 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. 875) 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. 75).*
- **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. 759)'s and call `gazebo::transport::init` (p. 76).*
- void **load\_world** (**WorldPtr** \_world, **sdf::ElementPtr** \_sdf)  
*Load world from `sdf::Element` (p. 266) pointer.*
- void **load\_worlds** (**sdf::ElementPtr** \_sdf)  
*load multiple worlds from single `sdf::Element` (p. 266) pointer*
- void **pause\_world** (**WorldPtr** \_world, bool \_pause)  
*Pause world by calling `World::SetPaused` (p. 886).*
- 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. 885) 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. 886) 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. 240) [Conversions.hh](#) (p. 931) [rendering/Conversions.hh](#) (p. 931).*
- 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. 107) *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. 107) *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. 103)'s Run-Time Shader system.*
- class **Scene**  
*Representation of an entire scene graph.*
- class **SelectionObj**  
**A** (p. 107) *graphical selection object.*
- class **UserCamera**  
**A** (p. 107) *camera used for user visualization of a scene.*
- class **VideoVisual**  
**A** (p. 107) *visual element that displays a video as a texture.*
- class **ViewController**  
*Base class for view controllers.*
- class **Visual**  
**A** (p. 107) *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. 651) by name.*
- bool **fini** ()  
*teardown rendering engine.*
- **rendering::ScenePtr get\_scene** (const std::string &\_name)  
*get pointer to **rendering::Scene** (p. 651) by name.*
- bool **init** ()  
*init rendering engine.*
- bool **load** ()  
*load rendering engine.*
- void **remove\_scene** (const std::string &\_name)  
*remove a **rendering::Scene** (p. 651) 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. 107) list of points, 1 vertex per point.

**RENDERING\_LINE\_LIST** A (p. 107) list of lines, 2 vertices per line.

**RENDERING\_LINE\_STRIP** A (p. 107) strip of connected lines, 1 vertex per line plus 1 start vertex.

**RENDERING\_TRIANGLE\_LIST** A (p. 107) list of triangles, 3 vertices per triangle.

**RENDERING\_TRIANGLE\_STRIP** A (p. 107) strip of triangles, 3 vertices for the first triangle, and 1 per triangle after that.

**RENDERING\_TRIANGLE\_FAN** A (p. 107) 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. 672) with one or more rays.*
- class **RFIDSensor**  
*Sensor (p. 672) class for RFID type of sensor.*
- class **RFIDTag**  
*RFIDTag (p. 635) to interact with RFIDTagSensors.*
- class **Sensor**  
*Base class for sensors.*
- class **SensorFactory**
- class **SensorManager**  
*Class to manage and update all sensors.*

### 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< **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**

## 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** ()  
*Run sensor generation continuously. This is a blocking call.*
- void **run\_once** (bool \_force=false)  
*Run the sensor generation one step.*
- 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<RaySensorPtr> gazebo::sensors::RaySensor_V`
- 9.9.2.10 `typedef RaySensor* gazebo::sensors::RaySensorPtr`
- 9.9.2.11 `typedef std::vector<RFIDSensor> gazebo::sensors::RFIDSensor_V`
- 9.9.2.12 `typedef RFIDSensor* gazebo::sensors::RFIDSensorPtr`
- 9.9.2.13 `typedef std::vector<RFIDTag> gazebo::sensors::RFIDTag_V`
- 9.9.2.14 `typedef RFIDTag* gazebo::sensors::RFIDTagPtr`
- 9.9.2.15 `typedef std::vector<SensorPtr> gazebo::sensors::Sensor_V`
- 9.9.2.16 `typedef Sensor*(* gazebo::sensors::SensorFactoryFn)()`
- 9.9.2.17 `typedef Sensor* gazebo::sensors::SensorPtr`

## 9.10 gazebo::transport Namespace Reference

### Classes

- class **CallbackHelper**  
*A (p. 107) helper class to handle callbacks when messages arrive.*
- class **CallbackHelperT**  
*Callback helper Template.*
- class **Connection**  
*Single TCP/IP connection manager.*
- class **ConnectionManager**  
*Manager of connections.*
- class **IOManager**  
*Manages boost::asio IO.*
- class **Node**  
*A (p. 107) node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.*
- class **Publication**  
*A (p. 107) publication for a topic.*
- class **PublicationTransport**  
*transport/transport.hh*
- class **Publisher**  
*A (p. 107) 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. 107) 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. 153)*
- 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 google Namespace Reference

### Namespaces

- namespace **protobuf**

## 9.12 google::protobuf Namespace Reference

### Namespaces

- namespace **compiler**

## 9.13 google::protobuf::compiler Namespace Reference

### Namespaces

- namespace **cpp**

## 9.14 google::protobuf::compiler::cpp Namespace Reference

## Classes

- class **GazeboGenerator**  
*Google protobuf message generator for `gazebo::msgs` (p. 87).*

## 9.15 Ogre Namespace Reference

## 9.16 ogre Namespace Reference

## 9.17 sdf Namespace Reference

namespace for Simulation Description Format parser

## Classes

- class **Converter**  
*Convert from one version of **SDF** (p. 669) to another.*
- class **Element**  
***SDF** (p. 669) **Element** (p. 266) class.*
- class **Param**  
***A** (p. 107) parameter class.*
- class **ParamT**  
*Templatized parameter class.*
- class **Plugin**
- class **SDF**  
*Base **SDF** (p. 669) 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. 669) 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. 669) values from a file.*
- bool **readString** (const std::string & \_xmlString, **SDFPtr** \_sdf)
  - Populate the **SDF** (p. 669) values from a string.*
- bool **readString** (const std::string & \_xmlString, **ElementPtr** \_sdf)
- bool **readXml** (TiXmlElement \* \_xml, **ElementPtr** \_sdf)

### 9.17.1 Detailed Description

namespace for Simulation Description Format parser

### 9.17.2 Typedef Documentation

- 9.17.2.1 typedef **Element\*** **sdf::ElementPtr**
- 9.17.2.2 typedef std::vector< **ElementPtr** > **sdf::ElementPtr\_V**
- 9.17.2.3 typedef std::vector< **ParamPtr** > **sdf::Param\_V**
- 9.17.2.4 typedef **Param\*** **sdf::ParamPtr**
- 9.17.2.5 typedef **SDF\*** **sdf::SDFPtr**

### 9.17.3 Function Documentation

- 9.17.3.1 void **sdf::addNestedModel** ( **ElementPtr** \_sdf, **ElementPtr** \_includeSDF )
- 9.17.3.2 void **sdf::copyChildren** ( **ElementPtr** \_sdf, TiXmlElement \* \_xml )
- 9.17.3.3 bool **sdf::init** ( **SDFPtr** \_sdf )

Init based on the installed sdf\_format.xml file.

- 9.17.3.4 bool **sdf::initDoc** ( TiXmlDocument \* \_xmlDoc, **SDFPtr** \_sdf )
- 9.17.3.5 bool **sdf::initDoc** ( TiXmlDocument \* \_xmlDoc, **ElementPtr** \_sdf )
- 9.17.3.6 bool **sdf::initFile** ( const std::string & \_filename, **SDFPtr** \_sdf )
- 9.17.3.7 bool **sdf::initFile** ( const std::string & \_filename, **ElementPtr** \_sdf )
- 9.17.3.8 bool **sdf::initString** ( const std::string & \_xmlString, **SDFPtr** \_sdf )
- 9.17.3.9 bool **sdf::initXml** ( TiXmlElement \* \_xml, **ElementPtr** \_sdf )

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

Populate the **SDF** (p. 669) values from a TinyXML document.

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

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

Populate the **SDF** (p. 669) values from a file.

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

Populate the **SDF** (p. 669) values from a string.

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

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

## 9.18 SkyX Namespace Reference

## 9.19 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**

### 9.19.1 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:

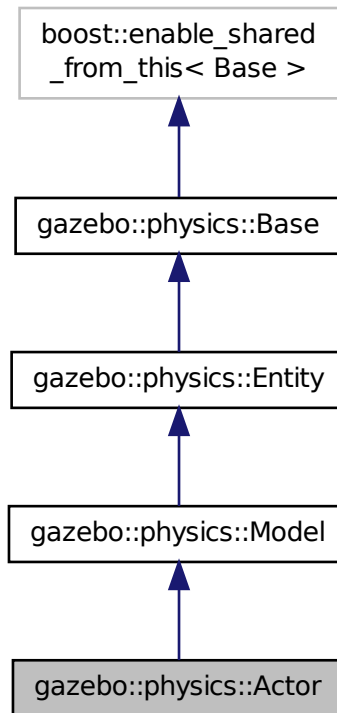
- `parser_urdf.hh`

### 10.2 gazebo::physics::Actor Class Reference

**Actor** (p. 107) 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. 133) 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. 107) 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	_parent	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. 473).

#### 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. 476).

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

Initialize the actor.

Reimplemented from **gazebo::physics::Model** (p. 477).

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

Reimplemented from **gazebo::physics::Entity** (p. 281).

**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. 143).

**10.2.3.9 virtual void gazebo::physics::Actor::UpdateParameters ( sdf::ElementPtr \_sdf ) [virtual]**

update the parameters using new sdf values.

**Parameters**

in	<code>_sdf</code>	SDF values to update from.
----	-------------------	----------------------------

Reimplemented from **gazebo::physics::Model** (p. 482).

## 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. 133) 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\*=  
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

<code>in</code>	<code>_radian</code>	Radians
-----------------	----------------------	---------

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

Copy constructor.

#### Parameters

<i>in</i>	<i>_angle</i>	<b>Angle</b> (p. 114) 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>	<b>Angle</b> (p. 114) 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>	<b>Angle</b> (p. 114) 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>	<b>Angle</b> (p. 114) 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>	<b>Angle</b> (p. 114) 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>	<b>Angle</b> (p. 114) 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>	<b>Angle</b> (p. 114) 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>	<b>Angle</b> (p. 114) 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>	<b>Angle</b> (p. 114) 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>	<b>Angle</b> (p. 114) for division
----	---------------	------------------------------------

## Returns

angle

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

Less than operator.

## Parameters

in	<i>_angle</i>	<b>Angle</b> (p. 114) 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

in	<i>_angle</i>	<b>Angle</b> (p. 114) 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

in	<i>_angle</i>	<b>Angle</b> (p. 114) 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

in	<i>_angle</i>	<b>Angle</b> (p. 114) 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>	<b>Angle</b> (p. 114) 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:

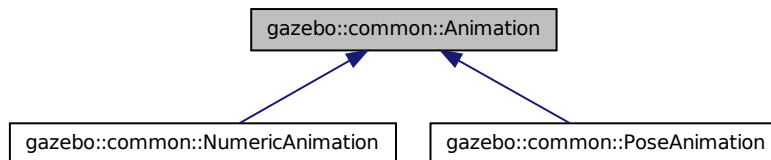
- **Angle.hh**

## 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 &\_firstKeyIndex) const

*Get the two key frames that bound a time value.*

## Protected Attributes

- bool **build**

*determines if the interpolation splines need building*

- **KeyFrame\_V** **keyFrames**

*array of key frames*

- double **length**

*animation duration*

- bool **loop**

*true if animation repeats*

- std::string **name**

*animation name*

- double **timePos**

*current time position*

### 10.4.1 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. 107) 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:

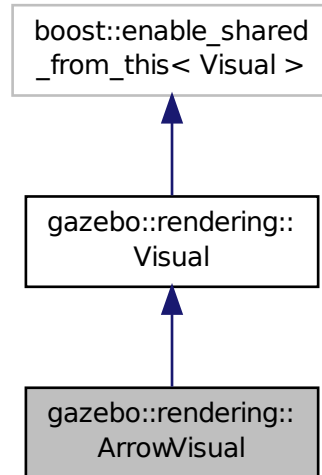
- **Animation.hh**

## 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. 863).

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:

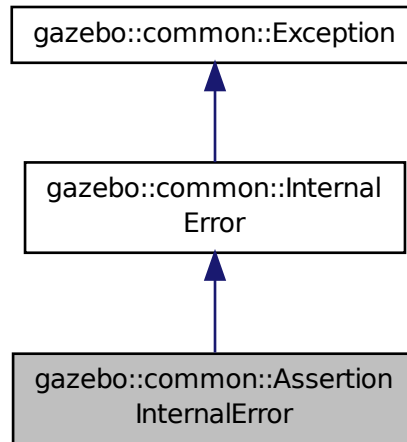
- **ArrowVisual.hh**

## 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:

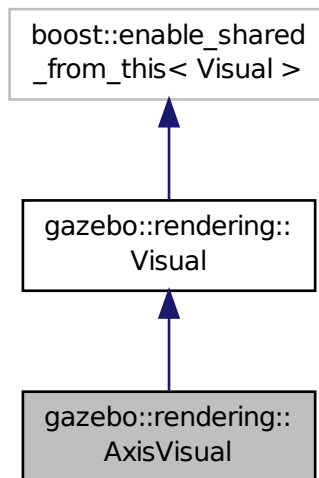
- **Exception.hh**

## 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 <b>AxisVisual</b> (p. 129)
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. 863).

#### 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:

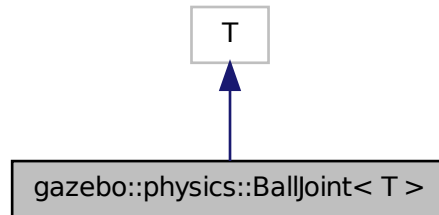
- **AxisVisual.hh**

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

**Base** (p. 133) 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. 131).*
- 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. 133) 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. 131).

#### Parameters

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

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

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

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

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

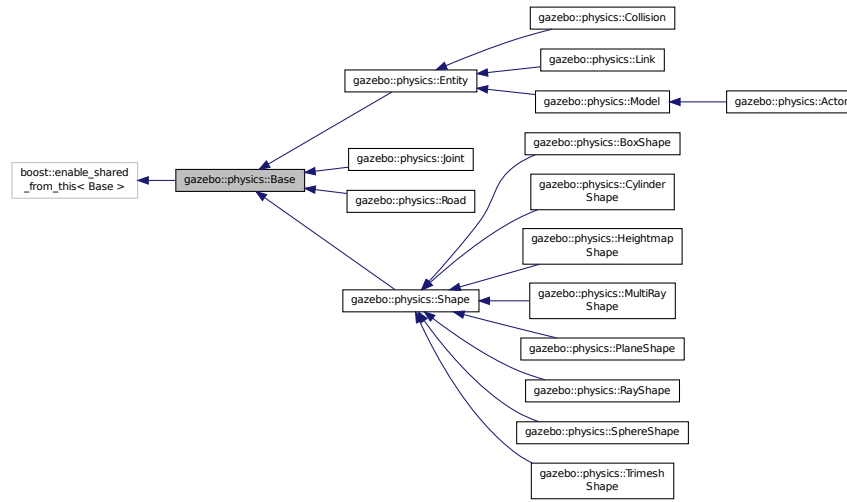
- **BallJoint.hh**

## 10.9 gazebo::physics::Base Class Reference

**Base** (p. 133) 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

- Get a child or self by id.*

  - **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. 875) 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. 136)'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. 133) 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. 133) type.
- ENTITY** **Entity** (p. 274) type.
- MODEL** **Model** (p. 469) type.
- LINK** **Link** (p. 404) type.
- COLLISION** **Collision** (p. 190) type.

**ACTOR Actor** (p. 107) type.

**LIGHT Light** type.

**VISUAL Visual** type.

**JOINT Joint** (p. 371) type.

**BALL\_JOINT BallJoint** (p. 131) type.

**HINGE2\_JOINT Hing2Joint** type.

**HINGE\_JOINT HingeJoint** (p. 350) type.

**SLIDER\_JOINT SliderJoint** (p. 718) type.

**SCREW\_JOINT ScrewJoint** (p. 666) type.

**UNIVERSAL\_JOINT UniversalJoint** (p. 793) type.

**SHAPE Shape** (p. 693) type.

**BOX\_SHAPE BoxShape** (p. 149) type.

**CYLINDER\_SHAPE CylinderShape** (p. 243) type.

**HEIGHTMAP\_SHAPE HeightmapShape** (p. 344) type.

**MAP\_SHAPE MapShape** type.

**MULTIRAY\_SHAPE MultiRayShape** (p. 507) type.

**RAY\_SHAPE RayShape** (p. 622) type.

**PLANE\_SHAPE PlaneShape** (p. 566) type.

**SPHERE\_SHAPE SphereShape** (p. 722) type.

**TRIMESH\_SHAPE TrimeshShape** (p. 790) 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. 110), [gazebo::physics::Model](#) (p. 473), [gazebo::physics::Entity](#) (p. 277), [gazebo::physics::Link](#) (p. 411), and [gazebo::physics::Collision](#) (p. 193).

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

Get a child or self by id.

##### Parameters

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

##### Returns

**A** (p. 107) 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. 107) 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. 107) 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. 107) 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. 110), and `gazebo::physics::Model` (p. 476).

#### 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. 875) this object is in.

##### Returns

The **World** (p. 875) 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. 380), **gazebo::physics::RayShape** (p. 626), **gazebo::physics::Actor** (p. 111), **gazebo::physics::Model** (p. 477), **gazebo::physics::Link** (p. 416), **gazebo::physics::Collision** (p. 196), **gazebo::physics::HeightmapShape** (p. 347), **gazebo::physics::TrimeshShape** (p. 792), **gazebo::physics::Multi-RayShape** (p. 514), **gazebo::physics::PlaneShape** (p. 568), **gazebo::physics::Road** (p. 642), **gazebo::physics::Shape** (p. 695), **gazebo::physics::SphereShape** (p. 724), **gazebo::physics::BoxShape** (p. 151), and **gazebo::physics::CylinderShape** (p. 245).

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. 381), **gazebo::physics::Actor** (p. 111), **gazebo::physics::Entity** (p. 281), **gazebo::physics::Model** (p. 477), **gazebo::physics::Link** (p. 416), **gazebo::physics::Collision** (p. 197), **gazebo::physics::HeightmapShape** (p. 347), and **gazebo::physics::Road** (p. 642).

10.9.4.22 `bool gazebo::physics::Base::operator==( const Base & _ent ) const`

Returns true if the entities are the same.

Checks only the name.

## Parameters

<code>in</code>	<code>_ent</code>	<b>Base</b> (p. 133) 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**

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

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

Remove a child from this entity.

**Parameters**

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

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

Remove a child by name.

**Parameters**

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

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

Remove all children.

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

Reset the object.

Reimplemented in `gazebo::physics::Joint` (p.381), `gazebo::physics::Model` (p.478), `gazebo::physics::Entity` (p.282), and `gazebo::physics::Link` (p.417).

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

Calls recursive Reset on one of the `Base::EntityType` (p.136)'s.

**Parameters**

<code>in</code>	<code><i>_resetType</i></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

<code>in</code>	<code>_name</code>	New name.
-----------------	--------------------	-----------

Reimplemented in **`gazebo::physics::Entity`** (p. 283).

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

Set the parent.

Parameters

<code>in</code>	<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

<code>in</code>	<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

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

Reimplemented in **`gazebo::physics::Link`** (p. 420).

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

<code>in</code>	<code>_newWorld</code>	The new <b>World</b> (p. 875) 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. 384), **`gazebo::physics::MultiRayShape`** (p. 514), **`gazebo::physics-`**

**::Actor** (p. 111), **gazebo::physics::RayShape** (p. 627), **gazebo::physics::Link** (p. 420), **gazebo::physics::Model** (p. 482), and **gazebo::physics::TrimeshShape** (p. 793).

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

Update the parameters using new sdf values.

#### Parameters

<code>in</code>	<code>_sdf</code>	Update the object's parameters based on SDF values.
-----------------	-------------------	---

Reimplemented in **gazebo::physics::Joint** (p. 384), **gazebo::physics::Actor** (p. 111), **gazebo::physics::Model** (p. 482), **gazebo::physics::Entity** (p. 284), **gazebo::physics::Link** (p. 421), and **gazebo::physics::Collision** (p. 198).

## 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** &\_box)  
*Merge a box with this box.*
- **Box operator+** (const **Box** &\_b) const  
*Addition operator.*
- const **Box** & **operator+=** (const **Box** &\_b)  
*Addition set operator.*
- **Box operator-** (const **Vector3** &\_v)  
*Subtract a vector from the min and max values.*
- **Box** & **operator=** (const **Box** &\_b)  
*Assignment operator.*
- bool **operator==** (const **Box** &\_b)  
*Equality test operator.*

## Public Attributes

- **Vector3 max**  
*Maximum corner of the box.*
- **Vector3 min**  
*Minimum corner of the box.*

## Friends

- std::ostream & **operator<<** (std::ostream &\_out, const **gazebo::math::Box** &\_b)  
*Output operator.*

### 10.10.1 Detailed Description

Mathematical representation of a box and related functions.

### 10.10.2 Constructor & Destructor Documentation

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

Default constructor.

#### 10.10.2.2 gazebo::math::Box::Box ( const Vector3 & *\_min*, const Vector3 & *\_max* )

Constructor.

##### Parameters

in	<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>	<b>Box</b> (p. 144) 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>	<b>Box</b> (p. 144) 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>	<b>Box</b> (p. 144) to add
-----------------	-----------------	----------------------------

#### Returns

The new box

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

Addition set operator.

this = this + `_b`

#### Parameters

<code>in</code>	<code>_b</code>	<b>Box</b> (p. 144) 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

<code>_v</code>	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

<code>in</code>	<code>_b</code>	<b>Box</b> (p. 144) to copy
-----------------	-----------------	-----------------------------

#### Returns

The new box.

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

Equality test operator.

#### Parameters

<code>in</code>	<code>_b</code>	<b>Box</b> (p. 144) 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>	<b>Box</b> (p. 144) 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:

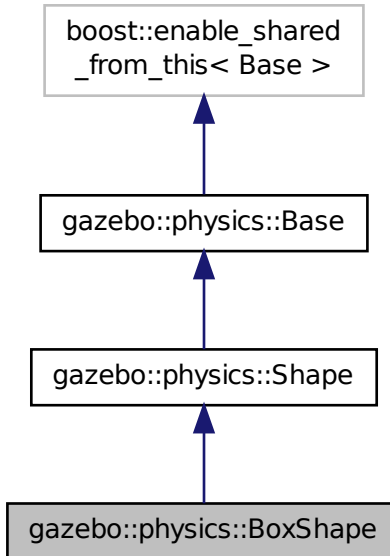
- **Box.hh**

## 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 <b>Collision</b> (p. 190).
----	----------------	-----------------------------------

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. 695).

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. 695).

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. 695).

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. 107) pointer to a new **Skeleton** (p. 698)

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

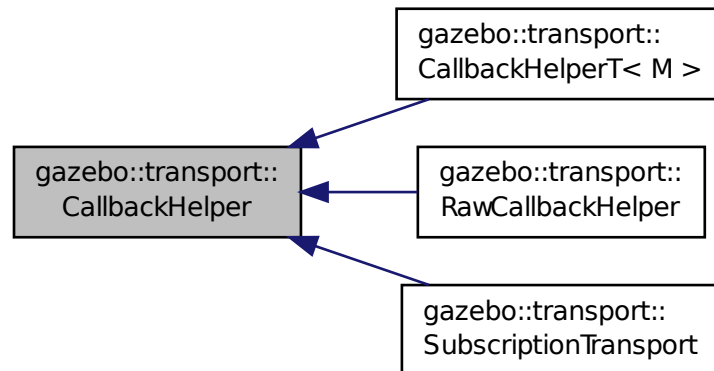
- **BVHLoader.hh**

## 10.13 gazebo::transport::CallbackHelper Class Reference

**A** (p. 107) 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. 107) 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. 615), and **gazebo::transport::CallbackHelperT< M >** (p. 157).

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. 615), **gazebo::transport::CallbackHelperT< M >** (p. 157), and **gazebo::transport::SubscriptionTransport** (p. 749).

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. 615), **gazebo::transport::CallbackHelperT< M >** (p. 157), and **gazebo::transport::SubscriptionTransport** (p. 749).

## 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:

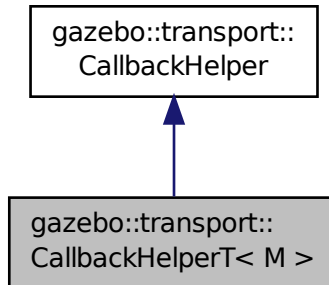
- **CallbackHelper.hh**

## 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. 155).

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. 155).

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. 155).

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

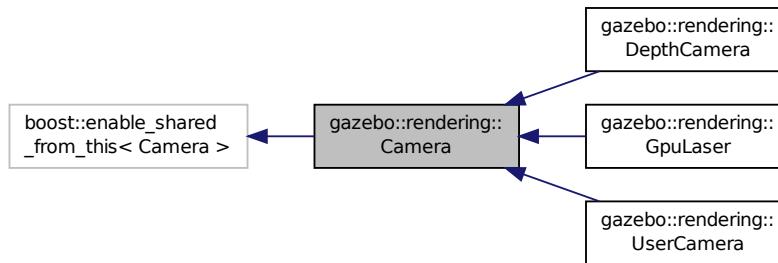
- `CallbackHelper.hh`

## 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.*
- `unsigned int GetImageHeight () const`  
*Get the height of the image.*
- `unsigned int GetImageWidth () const`  
*Get the width of the image.*
- `bool GetInitialized () const`  
*Returns true if 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
- Return true if the camera has been initialized.*

  - 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. 179) 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. 651) 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. 651) node that controls camera position.*
- sdf::ElementPtr **sdf**  
*Camera (p. 157)'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>	<b>Scene</b> (p. 651) 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. 798).

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. 799).

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. 107) 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. 319), `gazebo::rendering::DepthCamera` (p. 249), and `gazebo::rendering::UserCamera` (p. 799).

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

**Returns**

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

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

Get the height of the image.

**Returns**

Image height

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

Get the width of the image.

**Returns**

Image width

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

Returns true if initialized.

**Returns**

True if the camera is initialized

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. 598)

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. 319), **gazebo::rendering::DepthCamera** (p. 249), and **gazebo::rendering::UserCamera** (p. 801).

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]`

Return true if the camera has been initialized.

#### Returns

True if initialized was successful

References initialized.

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. 801).

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

Load the camera with default parameters.

Reimplemented in **`gazebo::rendering::GpuLaser`** (p. 319), **`gazebo::rendering::DepthCamera`** (p. 249), and **`gazebo::rendering::UserCamera`** (p. 801).

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. 175)

**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. 801).

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. 174)

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. 319), **gazebo::rendering::DepthCamera** (p. 249), and **gazebo::rendering::UserCamera** (p. 802).

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. 802).

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 <b>math::Pose</b> (p. 573) of the camera
----	-------	--

Reimplemented in **gazebo::rendering::UserCamera** (p. 803).

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. 179) 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. 803).

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. 804).

**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.

Referenced by `IsInitialized()`.

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. 651) 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. 651) node that controls camera position.

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

**Camera** (p. 157)'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:

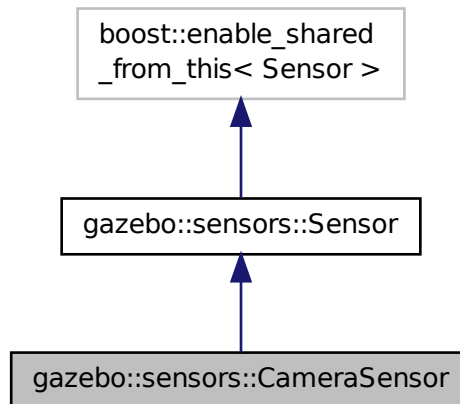
- **Camera.hh**

## 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. 157).*
- 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. 676).

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

Returns a pointer to the `rendering::Camera` (p. 157).

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. 677).

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

Initialize the camera.

Reimplemented from `gazebo::sensors::Sensor` (p. 678).

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. 678).

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 <b>Sensor</b> (p. 672) parameters
in	<code>_worldName</code>	Name of world to load from

Reimplemented from `gazebo::sensors::Sensor` (p. 678).

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. 678).

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. 679).

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. 679).

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

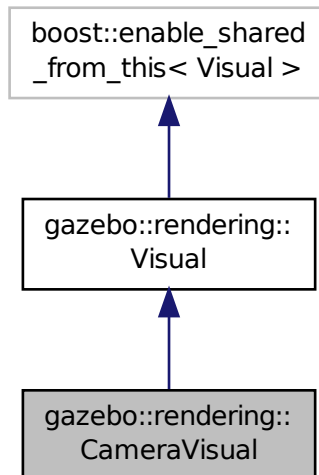
- `CameraSensor.hh`

## 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. 850).

## 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 <b>Visual</b> (p. 850)
<i>in</i>	<i>_vis</i>	Pointer to the parent <b>Visual</b> (p. 850)

#### 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. 850).

Parameters

<i>in</i>	<i>_width</i>	Width of the <b>Camera</b> (p. 157) image
<i>in</i>	<i>_height</i>	Height of the <b>Camera</b> (p. 157) image

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

- **CameraVisual.hh**

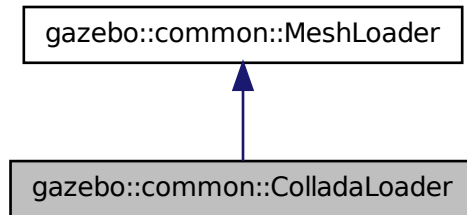
## 10.18 gazebo::common::ColladaLoader Class Reference

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. 455)

Implements **gazebo::common::MeshLoader** (p. 464).

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

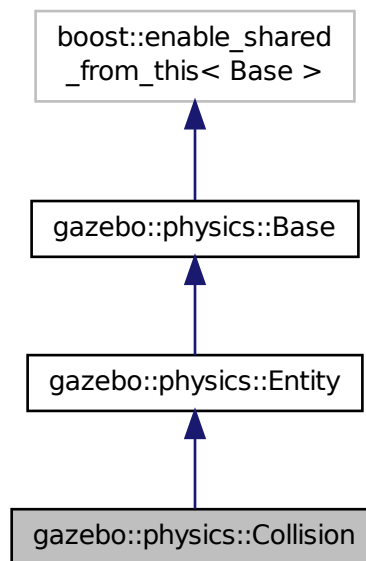
- **ColladaLoader.hh**

## 10.19 gazebo::physics::Collision Class Reference

**Base** (p. 133) 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. 693).*

## Additional Inherited Members

### 10.19.1 Detailed Description

**Base** (p. 133) 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>	<b>Link</b> (p. 404) 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.277).

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.277).

### 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. 404).

### 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. 279).

### 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. 279).

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. 279).

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. 280).

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. 136)

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. 280).

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. 280).

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. 280).

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. 281).

10.19.3.23 **virtual void** gazebo::physics::Collision::Init ( ) [virtual]

Initialize the collision.

Reimplemented from **gazebo::physics::Base** (p. 141).



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. 281).

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 m.
----	-------------------	----------------------------

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. 284).

## 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. 693).

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

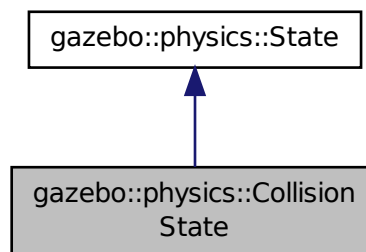
- **Collision.hh**

## 10.20 gazebo::physics::CollisionState Class Reference

Store state information of a **physics::Collision** (p. 190) 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. 190) 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. 190) object.

This class captures the entire state of a **Collision** (p. 190) at one specific time during a simulation run.

**State** (p. 729) of a **Collision** (p. 190) 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. 199) from an existing **Collision** (p. 190).

#### Parameters

in	_model	Pointer to the <b>Link</b> (p. 404) from which to gather state info.
----	--------	--

#### 10.20.2.3 gazebo::physics::CollisionState::CollisionState ( const **sdf::ElementPtr** \_sdf ) [explicit]

Constructor.

Build a **CollisionState** (p. 199) 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. 190) pose.

#### Returns

The pose of the **CollisionState** (p. 199)

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. 199) 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. 732).

### 10.20.3.5 CollisionState gazebo::physics::CollisionState::operator+ ( const CollisionState & \_state ) const

Addition operator.

#### Parameters

<i>in</i>	<i>_pt</i>	<b>A</b> (p. 107) 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>	<b>A</b> (p. 107) 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>	<b>State</b> (p. 729) 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>	<b>Collision</b> (p. 190) 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>	<b>Color</b> (p. 203) 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. 821) format

10.21.4.5 `uint32 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>	<b>Color</b> (p. 203) 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>	<b>Color</b> (p. 203) 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>	<b>Color</b> (p. 203) 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>	<b>Color</b> (p. 203) 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>	<b>Color</b> (p. 203) 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>	<b>Color</b> (p. 203) 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>	<b>Color</b> (p. 203) 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:

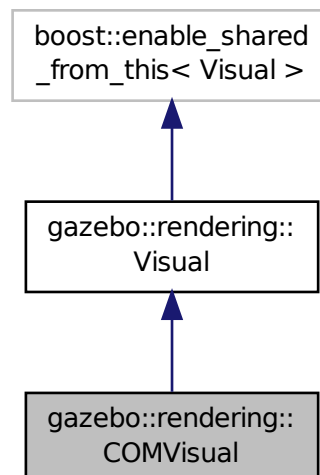
- **Color.hh**

## 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. 850) 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 <b>Visual</b> (p. 850)
in	<code>_vis</code>	Parent <b>Visual</b> (p. 850)

#### 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. 850) 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. 107) 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. 107) 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	<b>Event</b> (p. 285) 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:

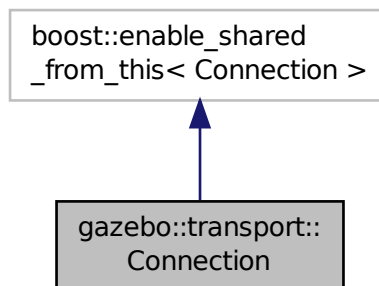
- **Event.hh**

## 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 **EnqueueMsg** (const std::string &\_buffer, bool \_force=false)  
*Write data to the socket.*
- unsigned int **GetId** () const  
*Get the ID of the connection.*
- std::string **GetLocalAddress** () const  
*Get the local address of this connection.*
- std::string **GetLocalHostname** () const  
*Get the local hostname.*
- 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** ()  
*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 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

<code>in</code>	<code><i>_host</i></code>	The host to connect to
<code>in</code>	<code><i>_port</i></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

<code>in</code>	<code><i>_subscriber</i></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

<code>in</code>	<code><i>_subscriber</i></code>	Handle previously returned by <b>ConnectToShutdown()</b> (p. 220)
-----------------	---------------------------------	---

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

<code>in</code>	<code><i>_buffer</i></code>	Data to write
<code>in</code>	<code><i>_force</i></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 `std::string gazebo::transport::Connection::GetLocalHostname ( ) const`

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 ( )`

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

<code>in</code>	<code><i>_cb</i></code>	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

<code>in</code>	<code><i>_ip</i></code>	Dotted quad to validate.
-----------------	-------------------------	--------------------------

#### Returns

True if the `_ip` is a valid.

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

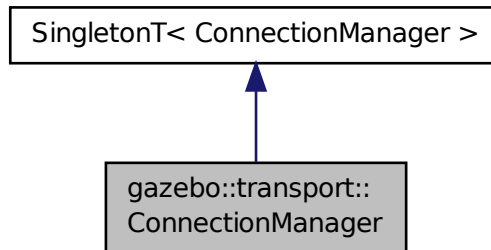
- **Connection.hh**

## 10.25 gazebo::transport::ConnectionManager Class Reference

Manager of connections.

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

Inheritance diagram for gazebo::transport::ConnectionManager:



## Public Member Functions

- void **Advertise** (const std::string &\_topic, const std::string &\_msgType)  
*Advertise a topic.*
- **ConnectionPtr ConnectToRemoteHost** (const std::string &\_host, unsigned int \_port)  
*Connect to a remote server.*
- void **Fini** ()  
*Finalize the connection manager.*
- void **GetAllPublishers** (std::list< msgs::Publish > &\_publishers)  
*Explicitly update the publisher list.*
- void **GetTopicNamespaces** (std::list< std::string > &\_namespaces)  
*Get all the topic namespaces.*
- bool **Init** (const std::string &\_masterHost, unsigned int \_masterPort)  
*Initialize the connection manager.*
- bool **IsRunning** () const  
*Is the manager running?*
- void **RegisterTopicNamespace** (const std::string &\_name)  
*Register a new topic namespace.*
- void **RemoveConnection** (**ConnectionPtr** &\_conn)  
*Remove a connection from the manager.*
- void **Run** ()  
*Run the connection manager loop.*
- void **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>	<b>A</b> (p. 107) subscription object
----	-------------	---------------------------------------

10.25.2.16 void gazebo::transport::ConnectionManager::Unsubscribe ( const std::string & *\_topic*, const std::string & *\_msgType* )

Unsubscribe from a topic.

#### Parameters

in	<i>_topic</i>	The topic to unsubscribe from
in	<i>_msgType</i>	The type of the topic

### 10.25.3 Member Data Documentation

10.25.3.1 `std::vector<event::ConnectionPtr> gazebo::transport::ConnectionManager::eventConnections` [protected]

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

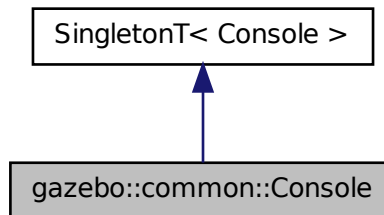
- **ConnectionManager.hh**

## 10.26 gazebo::common::Console Class Reference

Message, error, warning functionality.

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

Inheritance diagram for gazebo::common::Console:



### Public Member Functions

- `std::ostream & ColorErr (const std::string &_lbl, const std::string &_file, unsigned int _line, int _color)`  
*Use this to output an error to the terminal.*
- `std::ostream & ColorMsg (const std::string &_lbl, int _color)`  
*Use this to output a colored message to the terminal.*
- `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

#### 10.26.1 Detailed Description

Message, error, warning functionality.

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



- **Console.hh**

## 10.27 gazebo::physics::Contact Class Reference

**A** (p. 107) 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.*

### 10.27.1 Detailed Description

**A** (p. 107) contact between two collisions.

Each contact can consist of a number of contact points

### 10.27.2 Constructor & Destructor Documentation

#### 10.27.2.1 gazebo::physics::Contact::Contact ( )

Constructor.

#### 10.27.2.2 gazebo::physics::Contact::Contact ( const Contact & *\_contact* )

Copy constructor.

#### Parameters

in	<i>_contact</i>	<b>Contact</b> (p. 229) to copy.
----	-----------------	----------------------------------

#### 10.27.2.3 virtual gazebo::physics::Contact::~~Contact ( ) [virtual]

Destructor.

### 10.27.3 Member Function Documentation

#### 10.27.3.1 std::string gazebo::physics::Contact::DebugString ( ) const

Produce a debug string.

#### Returns

**A** (p. 107) string that contains the values of the contact.

#### 10.27.3.2 void gazebo::physics::Contact::FillMsg ( msgs::Contact & *\_msg* ) const

Populate a msgs::Contact with data from this.

#### Parameters

out	<i>_msg</i>	<b>Contact</b> (p. 229) message the will hold the data.
-----	-------------	---

#### 10.27.3.3 Contact& gazebo::physics::Contact::operator= ( const Contact & *\_contact* )

Operator =.

## Parameters

in	_contact	<b>Contact</b> (p. 229) to copy.
----	----------	----------------------------------

## Returns

Reference to this contact

#### 10.27.3.4 Contact& gazebo::physics::Contact::operator= ( const msgs::Contact & \_contact )

Operator =.

## Parameters

in	_contact	msgs::Contact to copy.
----	----------	------------------------

## Returns

Reference to this contact

#### 10.27.3.5 void gazebo::physics::Contact::Reset ( )

Reset to default values.

### 10.27.4 Member Data Documentation

#### 10.27.4.1 std::string gazebo::physics::Contact::collision1

Name of the first collision object.

#### 10.27.4.2 std::string gazebo::physics::Contact::collision2

Name of the second collision object.

#### 10.27.4.3 int gazebo::physics::Contact::count

Length of all the arrays.

#### 10.27.4.4 double gazebo::physics::Contact::depths[32]

Array of contact depths.

#### 10.27.4.5 math::Vector3 gazebo::physics::Contact::normals[32]

Array of force normals.

#### 10.27.4.6 `math::Vector3 gazebo::physics::Contact::positions[32]`

Array of force positions.

#### 10.27.4.7 `common::Time gazebo::physics::Contact::time`

Time at which the contact occurred.

#### 10.27.4.8 `JointWrench gazebo::physics::Contact::wrench[32]`

Array of forces for the contact.

All forces and torques are relative to the center of mass of the respective links that the collision elements are attached to.

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

- `Contact.hh`

## 10.28 `gazebo::physics::ContactManager` Class Reference

Aggregates all the contact information generated by the collision detection engine.

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

### Public Member Functions

- `ContactManager ()`  
*Constructor.*
- `virtual ~ContactManager ()`  
*Destructor.*
- `void Clear ()`  
*Clear all stored contacts.*
- `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. 232).*
- `Contact * NewContact (Collision *_collision1, Collision *_collision2, const common::Time &_time)`  
*Add a new contact.*
- `void PublishContacts ()`  
*Publish all contacts in a `msgs::Contacts` message.*
- `void ResetCount ()`  
*Set the contact count to zero.*

### 10.28.1 Detailed Description

Aggregates all the contact information generated by the collision detection engine.

### 10.28.2 Constructor & Destructor Documentation

#### 10.28.2.1 gazebo::physics::ContactManager::ContactManager ( )

Constructor.

#### 10.28.2.2 virtual gazebo::physics::ContactManager::~~ContactManager ( ) [virtual]

Destructor.

### 10.28.3 Member Function Documentation

#### 10.28.3.1 void gazebo::physics::ContactManager::Clear ( )

Clear all stored contacts.

#### 10.28.3.2 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. 233).

#### Parameters

<i>in</i>	<i>_index</i>	Index of the <b>Contact</b> (p. 229) to return.
-----------	---------------	---

#### Returns

Pointer to a contact, NULL If index is invalid.

#### 10.28.3.3 unsigned int gazebo::physics::ContactManager::GetContactCount ( ) const

Return the number of valid contacts.

#### 10.28.3.4 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. 233)

#### Returns

Vector of contact pointers.

10.28.3.5 `void gazebo::physics::ContactManager::Init ( WorldPtr _world )`

Initialize the **ContactManager** (p. 232).

This is required in order to publish contact messages via the **ContactManager::PublishContacts** (p. 234) method.

#### Parameters

<code>in</code>	<code>_world</code>	Pointer to the world that is initializing the contact manager.
-----------------	---------------------	--

10.28.3.6 `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.

10.28.3.7 `void gazebo::physics::ContactManager::PublishContacts ( )`

Publish all contacts in a `msgs::Contacts` message.

10.28.3.8 `void gazebo::physics::ContactManager::ResetCount ( )`

Set the contact count to zero.

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

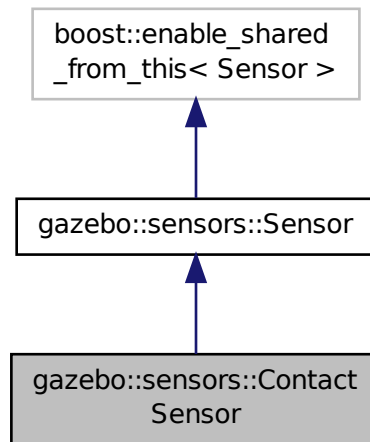
- **ContactManager.hh**

## 10.29 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.*
- 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

### 10.29.1 Detailed Description

Contact sensor.

This sensor detects and reports contacts between objects

### 10.29.2 Constructor & Destructor Documentation

#### 10.29.2.1 gazebo::sensors::ContactSensor::ContactSensor ( )

Constructor.

#### 10.29.2.2 virtual gazebo::sensors::ContactSensor::~~ContactSensor ( ) [virtual]

Destructor.

### 10.29.3 Member Function Documentation

#### 10.29.3.1 virtual void gazebo::sensors::ContactSensor::Fini ( ) [protected],[virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 676).

#### 10.29.3.2 unsigned int gazebo::sensors::ContactSensor::GetCollisionContactCount ( const std::string & \_collisionName ) const

Return the number of contacts for an observed collision.

#### Parameters

in	<code>_collisionName</code>	The name of the observed collision.
----	-----------------------------	-------------------------------------

#### Returns

The collision contact count.

#### 10.29.3.3 unsigned int gazebo::sensors::ContactSensor::GetCollisionCount ( ) const

Get the number of collisions that the sensor is observing.



**Returns**

Number of collisions.

**10.29.3.4** `std::string gazebo::sensors::ContactSensor::GetCollisionName ( unsigned int _index ) const`

Get a collision name at index *\_index*.

**Parameters**

<i>in</i>	<i>_index</i>	Index of collision in collection of collisions.
-----------	---------------	---

**Returns**

name of collision.

**10.29.3.5** `msgs::Contacts gazebo::sensors::ContactSensor::GetContacts ( ) const`

Get all the contacts.

**Returns**

Message that contains all the contact information

**10.29.3.6** `std::map<std::string, physics::Contact> gazebo::sensors::ContactSensor::GetContacts ( const std::string & _collisionName )`

Gets contacts of a collision.

**Parameters**

<i>in</i>	<i>_collisionName</i>	Name of collision
-----------	-----------------------	-------------------

**Returns**

Container of contacts

**10.29.3.7** `virtual void gazebo::sensors::ContactSensor::Init ( ) [virtual]`

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 678).

**10.29.3.8** `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. 678).

**10.29.3.9** `virtual void gazebo::sensors::ContactSensor::Load ( const std::string & _worldName, sdf::ElementPtr _sdf )`  
`[virtual]`

Load the sensor with SDF parameters.

**Parameters**

<code>in</code>	<code>_sdf</code>	SDF <b>Sensor</b> (p. 672) parameters
<code>in</code>	<code>_worldName</code>	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 678).

**10.29.3.10** `virtual void gazebo::sensors::ContactSensor::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. 678).

**10.29.3.11** `virtual void gazebo::sensors::ContactSensor::UpdateImpl ( bool _force )` `[protected]`, `[virtual]`

Update the sensor information.

**Parameters**

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

Reimplemented from **gazebo::sensors::Sensor** (p. 679).

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

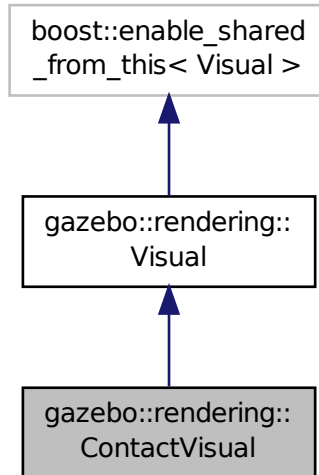
- **ContactSensor.hh**

## 10.30 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

### 10.30.1 Detailed Description

Contact visualization.

This class visualizes contact points by drawing arrows in the 3D environment.

### 10.30.2 Constructor & Destructor Documentation

10.30.2.1 gazebo::rendering::ContactVisual::ContactVisual ( const std::string & \_name, **VisualPtr** \_vis, const std::string & \_topicName )

Constructor.

## Parameters

in	<code>_name</code>	Name of the <b>ContactVisual</b> (p. 238)
in	<code>_vis</code>	Pointer the parent <b>Visual</b> (p. 850)
in	<code>_topicName</code>	Name of the topic which publishes the contact information.

10.30.2.2 `virtual gazebo::rendering::ContactVisual::~ContactVisual ( ) [virtual]`

Destructor.

### 10.30.3 Member Function Documentation

10.30.3.1 `void gazebo::rendering::ContactVisual::SetEnabled ( bool _enabled )`

Set to true to enable contact visualization.

## Parameters

in	<code>_enabled</code>	True to show contacts, false to hide.
----	-----------------------	---------------------------------------

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

- **ContactVisual.hh**

## 10.31 gazebo::rendering::Conversions Class Reference

**Conversions** (p. 240) **Conversions.hh** (p. 931) **rendering/Conversions.hh** (p. 931).

```
#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. 103) 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. 103) quaternion.*
- static `math::Quaternion Convert` (const `Ogre::Quaternion` &`_v`)  
***Ogre** (p. 103) quaternion to Gazebo quaternion.*

#### 10.31.1 Detailed Description

**Conversions** (p. 240) **Conversions.hh** (p. 931) **rendering/Conversions.hh** (p. 931).

**A** (p. 107) set of utility function to convert between Gazebo and **Ogre** (p. 103) data types

## 10.31.2 Member Function Documentation

10.31.2.1 `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. 103) color value

10.31.2.2 `static common::Color gazebo::rendering::Conversions::Convert ( const Ogre::ColourValue & _clr ) [static]`

Return the equivalent gazebo color.

### Parameters

in	_clr	<b>Ogre</b> (p. 103) color to convert
----	------	---------------------------------------

### Returns

Gazebo color value

10.31.2.3 `static Ogre::Vector3 gazebo::rendering::Conversions::Convert ( const math::Vector3 & _v ) [static]`

return **Ogre** (p. 103) Vector from Gazebo Vector3

### Parameters

in	_v	Gazebo vector
----	----	---------------

### Returns

**Ogre** (p. 103) vector

10.31.2.4 `static math::Vector3 gazebo::rendering::Conversions::Convert ( const Ogre::Vector3 & _v ) [static]`

return gazebo Vector from ogre Vector3

### Parameters

in	_v	<b>Ogre</b> (p. 103) vector
----	----	-----------------------------

### Returns

Gazebo vector

10.31.2.5 `static Ogre::Quaternion gazebo::rendering::Conversions::Convert ( const math::Quaternion & _v ) [static]`

Gazebo quaternion to **Ogre** (p. 103) quaternion.

#### Parameters

<code>in</code>	<code>_v</code>	Gazebo quaternion
-----------------	-----------------	-------------------

#### Returns

**Ogre** (p. 103) quaternion

10.31.2.6 `static math::Quaternion gazebo::rendering::Conversions::Convert ( const Ogre::Quaternion & _v ) [static]`

**Ogre** (p. 103) quaternion to Gazebo quaternion.

#### Parameters

<code>in</code>	<code>_v</code>	<b>Ogre</b> (p. 103) quaternion return Gazebo quaternion
-----------------	-----------------	--

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

- **Conversions.hh**

## 10.32 sdf::Converter Class Reference

Convert from one version of **SDF** (p. 669) to another.

```
#include <Converter.hh>
```

### Static Public Member Functions

- static bool **Convert** (TiXmlDocument \* \_doc, const std::string & \_toVersion, bool \_quiet=false)

#### 10.32.1 Detailed Description

Convert from one version of **SDF** (p. 669) to another.

#### 10.32.2 Member Function Documentation

10.32.2.1 `static bool sdf::Converter::Convert ( TiXmlDocument * _doc, const std::string & _toVersion, bool _quiet = false ) [static]`

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

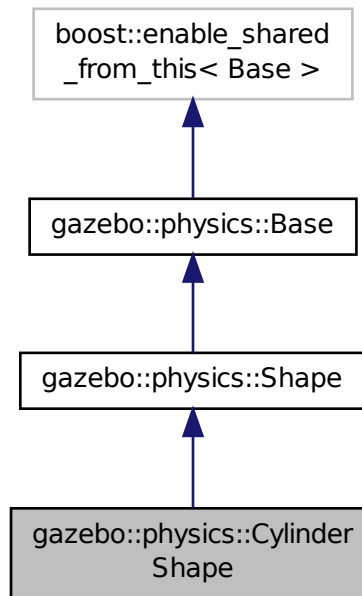
- **Converter.hh**

## 10.33 gazebo::physics::CylinderShape Class Reference

Cylinder collision.

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

Inheritance diagram for gazebo::physics::CylinderShape:



### Public Member Functions

- **CylinderShape** (**CollisionPtr** \_parent)  
*Constructor.*
- virtual **~CylinderShape** ()  
*Destructor.*
- void **FillMsg** (msgs::Geometry &\_msg)  
*Fill in the values for a geometry message.*
- double **GetLength** () const  
*Get length.*
- double **GetRadius** () const  
*Get radius.*
- void **Init** ()  
*Initialize the cylinder.*
- virtual void **ProcessMsg** (const msgs::Geometry &\_msg)  
*Update values based on a message.*
- void **SetLength** (double \_length)

*Set length.*

- void **SetRadius** (double *\_radius*)

*Set radius.*

- virtual void **SetSize** (double *\_radius*, double *\_length*)

*Set the size of the cylinder.*

## Additional Inherited Members

### 10.33.1 Detailed Description

Cylinder collision.

### 10.33.2 Constructor & Destructor Documentation

10.33.2.1 gazebo::physics::CylinderShape::CylinderShape ( CollisionPtr *\_parent* ) [explicit]

Constructor.

#### Parameters

in	<i>_parent</i>	Parent of the shape.
----	----------------	----------------------

10.33.2.2 virtual gazebo::physics::CylinderShape::~~CylinderShape ( ) [virtual]

Destructor.

### 10.33.3 Member Function Documentation

10.33.3.1 void gazebo::physics::CylinderShape::FillMsg ( msgs::Geometry & *\_msg* ) [virtual]

Fill in the values for a geometry message.

#### Parameters

out	<i>_msg</i>	The geometry message to fill.
-----	-------------	-------------------------------

Implements gazebo::physics::Shape (p. 695).

10.33.3.2 double gazebo::physics::CylinderShape::GetLength ( ) const

Get length.

#### Returns

The cylinder length.



10.33.3.3 `double gazebo::physics::CylinderShape::GetRadius ( ) const`

Get radius.

#### Returns

The cylinder radius.

10.33.3.4 `void gazebo::physics::CylinderShape::Init ( ) [virtual]`

Initialize the cylinder.

Implements **`gazebo::physics::Shape`** (p. 695).

10.33.3.5 `virtual void gazebo::physics::CylinderShape::ProcessMsg ( const msgs::Geometry & _msg ) [virtual]`

Update values based on a message.

#### Parameters

<code>in</code>	<code>_msg</code>	Message to update from.
-----------------	-------------------	-------------------------

Implements **`gazebo::physics::Shape`** (p. 695).

10.33.3.6 `void gazebo::physics::CylinderShape::SetLength ( double _length )`

Set length.

#### Parameters

<code>in</code>	<code>_length</code>	New length of the cylinder.
-----------------	----------------------	-----------------------------

10.33.3.7 `void gazebo::physics::CylinderShape::SetRadius ( double _radius )`

Set radius.

#### Parameters

<code>in</code>	<code>_radius</code>	New radius of the cylinder.
-----------------	----------------------	-----------------------------

10.33.3.8 `virtual void gazebo::physics::CylinderShape::SetSize ( double _radius, double _length ) [virtual]`

Set the size of the cylinder.

#### Parameters

<code>in</code>	<code>_radius</code>	New radius.
<code>in</code>	<code>_length</code>	New length.

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

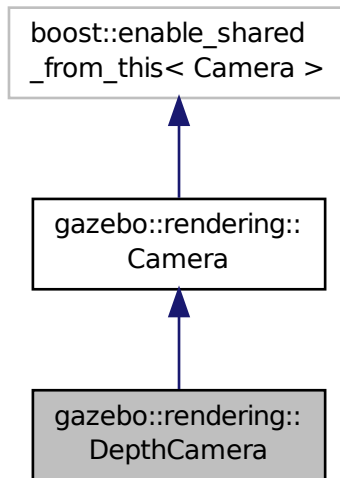
- [CylinderShape.hh](#)

## 10.34 gazebo::rendering::DepthCamera Class Reference

Depth camera used to render depth data into an image buffer.

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

Inheritance diagram for gazebo::rendering::DepthCamera:



### Public Member Functions

- **DepthCamera** (const std::string &\_namePrefix, **ScenePtr** \_scene, bool \_autoRender=true)  
*Constructor.*
- virtual ~**DepthCamera** ()  
*Destructor.*
- template<typename T >  
**event::ConnectionPtr ConnectNewDepthFrame** (T \_subscriber)  
*Connect a to the new depth image signal.*
- template<typename T >  
**event::ConnectionPtr ConnectNewRGBPointCloud** (T \_subscriber)  
*Connect a to the new rgb point cloud signal.*
- void **CreateDepthTexture** (const std::string &\_textureName)  
*Create a texture which will hold the depth data.*
- void **DisconnectNewDepthFrame** (**event::ConnectionPtr** &\_c)  
*Disconnect from an depth image singal.*
- void **DisconnectNewRGBPointCloud** (**event::ConnectionPtr** &c)

*Disconnect from an rgb point cloud singal.*

- void **Fini** ()  
*Finalize the camera.*
- virtual const float \* **GetDepthData** ()  
*All things needed to get back z buffer for depth data.*
- void **Init** ()  
*Initialize the camera.*
- void **Load** (sdf::ElementPtr &\_sdf)  
*Load the camera with a set of parmeters.*
- void **Load** ()  
*Load the camera with default parmeters.*
- virtual void **PostRender** ()  
*Render the camera.*
- virtual void **SetDepthTarget** (Ogre::RenderTarget \*\_target)  
*Set the render target, which renders the depth data.*

### Protected Attributes

- Ogre::RenderTarget \* **depthTarget**  
*Pointer to the depth target.*
- Ogre::Texture \* **depthTexture**  
*Pointer to the depth texture.*
- Ogre::Viewport \* **depthViewport**  
*Pointer to the depth viewport.*

### Additional Inherited Members

#### 10.34.1 Detailed Description

Depth camera used to render depth data into an image buffer.

#### 10.34.2 Constructor & Destructor Documentation

10.34.2.1 gazebo::rendering::DepthCamera::DepthCamera ( const std::string & *\_namePrefix*, ScenePtr *\_scene*, bool *\_autoRender* = true )

Constructor.

##### Parameters

in	<i>_namePrefix</i>	Unique prefix name for the camera.
in	<i>_scene</i>	<b>Scene</b> (p. 651) that will contain the camera
in	<i>_autoRender</i>	Almost everyone should leave this as true.

10.34.2.2 virtual gazebo::rendering::DepthCamera::~DepthCamera ( ) [virtual]

Destructor.

### 10.34.3 Member Function Documentation

10.34.3.1 `template<typename T> event::ConnectionPtr gazebo::rendering::DepthCamera::ConnectNewDepthFrame ( T  
_subscriber ) [inline]`

Connect a to the new depth image signal.

#### Parameters

in	_subscriber	Subscriber callback function
----	-------------	------------------------------

#### Returns

Pointer to the new Connection. This must be kept in scope

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

10.34.3.2 `template<typename T> event::ConnectionPtr gazebo::rendering::DepthCamera::ConnectNewRGBPointCloud ( T  
_subscriber ) [inline]`

Connect a to the new rgb point cloud signal.

#### Parameters

in	_subscriber	Subscriber callback function
----	-------------	------------------------------

#### Returns

Pointer to the new Connection. This must be kept in scope

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

10.34.3.3 `void gazebo::rendering::DepthCamera::CreateDepthTexture ( const std::string & _textureName )`

Create a texture which will hold the depth data.

#### Parameters

in	_textureName	Name of the texture to create
----	--------------	-------------------------------

10.34.3.4 `void gazebo::rendering::DepthCamera::DisconnectNewDepthFrame ( event::ConnectionPtr & _c ) [inline]`

Disconnect from an depth image signal.

#### Parameters

in	_c	The connection to disconnect
----	----	------------------------------

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

10.34.3.5 void gazebo::rendering::DepthCamera::DisconnectNewRGBPointCloud ( event::ConnectionPtr & c ) [inline]

Disconnect from an rgb point cloud singal.

#### Parameters

in	_c	The connection to disconnect
----	----	------------------------------

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

10.34.3.6 void gazebo::rendering::DepthCamera::Fini ( ) [virtual]

Finalize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 166).

10.34.3.7 virtual const float\* gazebo::rendering::DepthCamera::GetDepthData ( ) [virtual]

All things needed to get back z buffer for depth data.

#### Returns

The z-buffer as a float array

10.34.3.8 void gazebo::rendering::DepthCamera::Init ( ) [virtual]

Initialize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 173).

10.34.3.9 void gazebo::rendering::DepthCamera::Load ( sdf::ElementPtr & \_sdf )

Load the camera with a set of parmeters.

#### Parameters

in	_sdf	The SDF camera info
----	------	---------------------

10.34.3.10 void gazebo::rendering::DepthCamera::Load ( ) [virtual]

Load the camera with default parmeters.

Reimplemented from **gazebo::rendering::Camera** (p. 174).

10.34.3.11 virtual void gazebo::rendering::DepthCamera::PostRender ( ) [virtual]

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 175).

10.34.3.12 virtual void gazebo::rendering::DepthCamera::SetDepthTarget ( Ogre::RenderTarget \* *\_target* ) [virtual]

Set the render target, which renders the depth data.

#### Parameters

in	<i>_target</i>	Pointer to the render target
----	----------------	------------------------------

### 10.34.4 Member Data Documentation

10.34.4.1 Ogre::RenderTarget\* gazebo::rendering::DepthCamera::depthTarget [protected]

Pointer to the depth target.

10.34.4.2 Ogre::Texture\* gazebo::rendering::DepthCamera::depthTexture [protected]

Pointer to the depth texture.

10.34.4.3 Ogre::Viewport\* gazebo::rendering::DepthCamera::depthViewport [protected]

Pointer to the depth viewport.

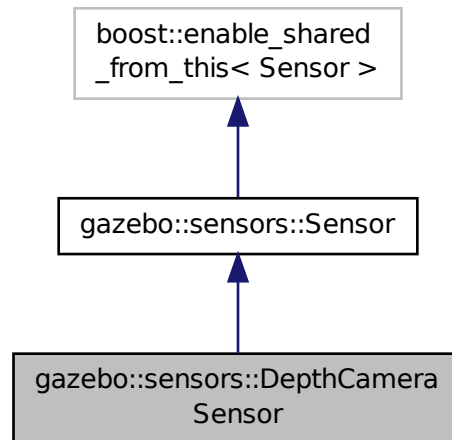
The documentation for this class was generated from the following file:

- **DepthCamera.hh**

## 10.35 gazebo::sensors::DepthCameraSensor Class Reference

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

Inheritance diagram for gazebo::sensors::DepthCameraSensor:



## Public Member Functions

- **DepthCameraSensor** ()  
*Constructor.*
- virtual **~DepthCameraSensor** ()  
*Destructor.*
- **rendering::DepthCameraPtr GetDepthCamera** () const  
*Returns a pointer to the **rendering::DepthCamera** (p. 246).*
- 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

### 10.35.1 Constructor & Destructor Documentation

#### 10.35.1.1 gazebo::sensors::DepthCameraSensor::DepthCameraSensor ( )

Constructor.

#### 10.35.1.2 virtual gazebo::sensors::DepthCameraSensor::~~DepthCameraSensor ( ) [virtual]

Destructor.

### 10.35.2 Member Function Documentation

#### 10.35.2.1 virtual void gazebo::sensors::DepthCameraSensor::Fini ( ) [protected],[virtual]

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 676).

#### 10.35.2.2 rendering::DepthCameraPtr gazebo::sensors::DepthCameraSensor::GetDepthCamera ( ) const [inline]

Returns a pointer to the **rendering::DepthCamera** (p. 246).

Returns

Depth Camera pointer

#### 10.35.2.3 virtual void gazebo::sensors::DepthCameraSensor::Init ( ) [protected],[virtual]

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 678).

#### 10.35.2.4 virtual void gazebo::sensors::DepthCameraSensor::Load ( const std::string & *\_worldName*, sdf::ElementPtr & *\_sdf* ) [protected],[virtual]

Load the sensor with SDF parameters.

Parameters

in	<i>_sdf</i>	SDF <b>Sensor</b> (p. 672) parameters
in	<i>_worldName</i>	Name of world to load from



10.35.2.5 virtual void gazebo::sensors::DepthCameraSensor::Load ( const std::string & *\_worldName* ) [protected],  
[virtual]

Load the sensor with default parameters.

#### Parameters

in	<i>_worldName</i>	Name of world to load from
----	-------------------	----------------------------

Reimplemented from **gazebo::sensors::Sensor** (p. 678).

10.35.2.6 bool gazebo::sensors::DepthCameraSensor::SaveFrame ( const std::string & *\_filename* )

Saves an image frame of depth camera sensor to file.

#### Parameters

in	<i>Name</i>	of file to save as
----	-------------	--------------------

#### Returns

True if saved, false if not

10.35.2.7 virtual void gazebo::sensors::DepthCameraSensor::SetActive ( bool *\_value* ) [virtual]

Set whether the sensor is active or not.

#### Parameters

in	<i>_value</i>	True if active, false if not
----	---------------	------------------------------

Reimplemented from **gazebo::sensors::Sensor** (p. 679).

10.35.2.8 virtual void gazebo::sensors::DepthCameraSensor::SetParent ( const std::string & *\_name* ) [virtual]

Set the parent of the sensor.

#### Parameters

in	<i>_name</i>	Name of parent
----	--------------	----------------

Reimplemented from **gazebo::sensors::Sensor** (p. 679).

10.35.2.9 virtual void gazebo::sensors::DepthCameraSensor::UpdateImpl ( bool *\_force* ) [protected],[virtual]

Update the sensor information.

#### Parameters

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

Reimplemented from `gazebo::sensors::Sensor` (p. 679).

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

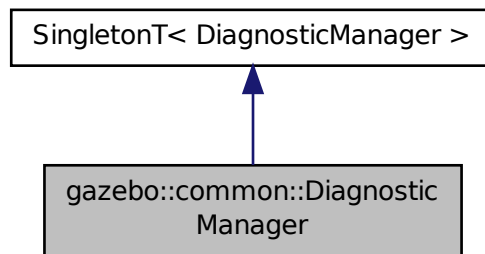
- `DepthCameraSensor.hh`

## 10.36 gazebo::common::DiagnosticManager Class Reference

**A** (p. 107) diagnostic manager class.

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

Inheritance diagram for `gazebo::common::DiagnosticManager`:



### Public Member Functions

- **DiagnosticTimerPtr CreateTimer** (const std::string &\_name)  
*Create a new timer instance.*
- bool **GetEnabled** () const  
*Get whether the timers are enabled.*
- std::string **GetLabel** (int \_index) const  
*Get a label for a timer.*
- **Time GetTime** (int \_index) const  
*Get the time of a timer instance.*
- **Time GetTime** (const std::string &\_label) const  
*Get a time based on a label.*
- int **GetTimerCount** () const  
*Get the number of timers.*
- void **SetEnabled** (bool \_e)  
*Set whether timers are enabled.*
- void **TimerStart** (**DiagnosticTimer** \*\_timer)  
***A** (p. 107) diagnostic timer has started.*
- void **TimerStop** (**DiagnosticTimer** \*\_timer)  
***A** (p. 107) diagnostic timer has stopped.*

## Additional Inherited Members

### 10.36.1 Detailed Description

**A** (p. 107) diagnostic manager class.

### 10.36.2 Member Function Documentation

#### 10.36.2.1 DiagnosticTimerPtr gazebo::common::DiagnosticManager::CreateTimer ( const std::string & *\_name* )

Create a new timer instance.

##### Parameters

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

##### Returns

**A** (p. 107) pointer to the new diagnostic timer

#### 10.36.2.2 bool gazebo::common::DiagnosticManager::GetEnabled ( ) const [inline]

Get whether the timers are enabled.

##### Returns

TRue if the timers are enabled

#### 10.36.2.3 std::string gazebo::common::DiagnosticManager::GetLabel ( int *\_index* ) const

Get a label for a timer.

##### Parameters

<i>in</i>	<i>_index</i>	Index of a timer instance
-----------	---------------	---------------------------

##### Returns

Label of the specified timer

#### 10.36.2.4 Time gazebo::common::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** (p. 760) of the specified timer

10.36.2.5 `Time gazebo::common::DiagnosticManager::GetTime ( const std::string & _label ) const`

Get a time based on a label.

## Parameters

<code>in</code>	<code>_label</code>	Name of the timer instance
-----------------	---------------------	----------------------------

## Returns

**Time** (p. 760) of the specified timer

10.36.2.6 `int gazebo::common::DiagnosticManager::GetTimerCount ( ) const`

Get the number of timers.

## Returns

The number of timers

10.36.2.7 `void gazebo::common::DiagnosticManager::SetEnabled ( bool _e ) [inline]`

Set whether timers are enabled.

## Parameters

<code>in</code>	<code>_e</code>	True = timers are enabled
-----------------	-----------------	---------------------------

10.36.2.8 `void gazebo::common::DiagnosticManager::TimerStart ( DiagnosticTimer * _timer )`

**A** (p. 107) diagnostic timer has started.

## Parameters

<code>in</code>	<code>_timer</code>	The timer to start
-----------------	---------------------	--------------------

Referenced by `gazebo::common::DiagnosticTimer::DiagnosticTimer()`.

10.36.2.9 `void gazebo::common::DiagnosticManager::TimerStop ( DiagnosticTimer * _timer )`

**A** (p. 107) diagnostic timer has stopped.

## Parameters

<code>in</code>	<code>_time</code>	The timer to stop
-----------------	--------------------	-------------------

Referenced by gazebo::common::DiagnosticTimer::~~DiagnosticTimer().

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

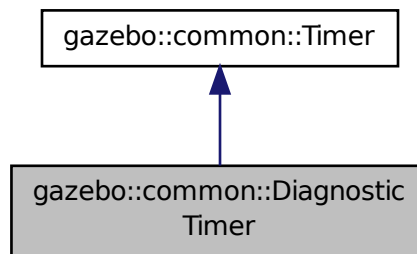
- **Diagnostics.hh**

## 10.37 gazebo::common::DiagnosticTimer Class Reference

**A** (p. 107) timer designed for diagnostics.

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

Inheritance diagram for gazebo::common::DiagnosticTimer:



### Public Member Functions

- **DiagnosticTimer** (const std::string &\_name)  
*Constructor.*
- virtual ~**DiagnosticTimer** ()  
*Destructor.*
- const std::string **GetName** () const  
*Get the name of the timer.*

#### 10.37.1 Detailed Description

**A** (p. 107) timer designed for diagnostics.

#### 10.37.2 Constructor & Destructor Documentation

10.37.2.1 gazebo::common::DiagnosticTimer::DiagnosticTimer ( const std::string & *\_name* ) [inline]

Constructor.

## Parameters

in	<i>_name</i>	Name of the timer
----	--------------	-------------------

References gazebo::common::Timer::Start(), and gazebo::common::DiagnosticManager::TimerStart().

10.37.2.2 virtual gazebo::common::DiagnosticTimer::~~DiagnosticTimer ( ) [inline],[virtual]

Destructor.

References gazebo::common::DiagnosticManager::TimerStop().

### 10.37.3 Member Function Documentation

10.37.3.1 const std::string gazebo::common::DiagnosticTimer::GetName ( ) const [inline]

Get the name of the timer.

## Returns

The name of timer

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

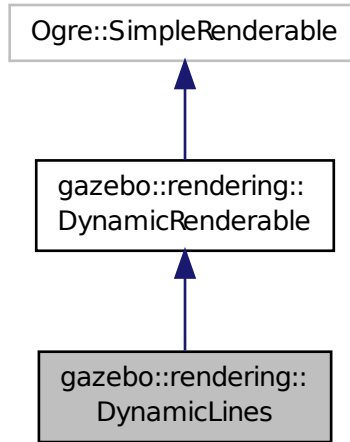
- **Diagnostics.hh**

## 10.38 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. 103)'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. 262), creates a simple vertex-only decl.*
- virtual void **FillHardwareBuffers** ()  
*Implementation **DynamicRenderable** (p. 262), pushes point list out to hardware memory.*

## Additional Inherited Members

### 10.38.1 Detailed Description

Class for drawing lines that can change.

### 10.38.2 Constructor & Destructor Documentation

10.38.2.1 gazebo::rendering::DynamicLines::DynamicLines ( RenderOpType *\_opType* = RENDERING\_LINE\_STRIP )

Constructor.

#### Parameters

<i>in</i>	<i>_opType</i>	The type of Line
-----------	----------------	------------------

10.38.2.2 virtual gazebo::rendering::DynamicLines::~DynamicLines ( ) [virtual]

Destructor.

### 10.38.3 Member Function Documentation

10.38.3.1 void gazebo::rendering::DynamicLines::AddPoint ( const math::Vector3 & *.pt* )

Add a point to the point list.

#### Parameters

<i>in</i>	<i>pt</i>	<b>math::Vector3</b> (p. 821) point
-----------	-----------	-------------------------------------

10.38.3.2 void gazebo::rendering::DynamicLines::AddPoint ( double *\_x*, double *\_y*, double *\_z* )

Add a point to the point list.



## Parameters

in	<code>_x</code>	X position.
in	<code>_y</code>	Y position.
in	<code>_z</code>	Z position.

## 10.38.3.3 void gazebo::rendering::DynamicLines::Clear ( )

Remove all points from the point list.

## 10.38.3.4 virtual void gazebo::rendering::DynamicLines::CreateVertexDeclaration ( ) [protected], [virtual]

Implementation **DynamicRenderable** (p. 262), creates a simple vertex-only decl.

Implements **gazebo::rendering::DynamicRenderable** (p. 264).

## 10.38.3.5 virtual void gazebo::rendering::DynamicLines::FillHardwareBuffers ( ) [protected], [virtual]

Implementation **DynamicRenderable** (p. 262), pushes point list out to hardware memory.

Implements **gazebo::rendering::DynamicRenderable** (p. 264).

## 10.38.3.6 static std::string gazebo::rendering::DynamicLines::GetMovableType ( ) [static]

Get type of movable.

## Returns

This returns "gazebo::dynamiclines"

## 10.38.3.7 virtual const Ogre::String&amp; gazebo::rendering::DynamicLines::getMovableType ( ) const [virtual]

Overridden function from **Ogre** (p. 103)'s base class.

## Returns

Returns "gazebo::ogredynamiclines"

10.38.3.8 const math::Vector3& gazebo::rendering::DynamicLines::GetPoint ( unsigned int *\_index* ) const

Return the location of an existing point in the point list.

## Parameters

in	<code>_index</code>	Number of the point to return
----	---------------------	-------------------------------

## Returns

**math::Vector3** (p. 821) value of the point

10.38.3.9 `unsigned int gazebo::rendering::DynamicLines::GetPointCount ( ) const`

Return the total number of points in the point list.

#### Returns

Number of points

10.38.3.10 `void gazebo::rendering::DynamicLines::SetPoint ( unsigned int _index, const math::Vector3 & _value )`

Change the location of an existing point in the point list.

#### Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the point to set
<code>in</code>	<code><i>_value</i></code>	<code>math::Vector3</code> (p. 821) value to set the point to

10.38.3.11 `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:

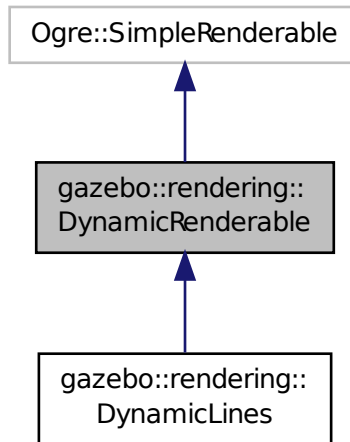
- `DynamicLines.hh`

## 10.39 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  $\sim$ **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.*

### 10.39.1 Detailed Description

Abstract base class providing mechanisms for dynamically growing hardware buffers.

### 10.39.2 Constructor & Destructor Documentation

#### 10.39.2.1 `gazebo::rendering::DynamicRenderable::DynamicRenderable ( )`

Constructor.

#### 10.39.2.2 `virtual gazebo::rendering::DynamicRenderable::~~DynamicRenderable ( ) [virtual]`

Virtual destructor.

### 10.39.3 Member Function Documentation

#### 10.39.3.1 `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. 261).

#### 10.39.3.2 `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. 261).

10.39.3.3 virtual `Ogre::Real gazebo::rendering::DynamicRenderable::getBoundingRadius ( ) const` [virtual]

Implementation of `Ogre::SimpleRenderable`.

#### Returns

The bounding radius

10.39.3.4 `RenderOpType gazebo::rendering::DynamicRenderable::GetOperationType ( ) const`

Get the render operation type.

#### Returns

The render operation type.

10.39.3.5 virtual `Ogre::Real gazebo::rendering::DynamicRenderable::getSquaredViewDepth ( const Ogre::Camera * _cam ) const` [virtual]

Implementation of `Ogre::SimpleRenderable`.

#### Parameters

in	_cam	Pointer to the <b>Ogre</b> (p. 103) camera that views the renderable.
----	------	---

#### Returns

The squared depth in the **Camera** (p. 157)'s view

10.39.3.6 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. 264).

#### Parameters

in	_opType	The type of render operation to perform.
in	_useIndices	Specifies whether to use indices to determine the vertices to use as input.

10.39.3.7 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.

**10.39.3.8 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.
-----------------	----------------------	--

**10.39.4 Member Data Documentation****10.39.4.1 size\_t gazebo::rendering::DynamicRenderable::indexBufferCapacity [protected]**

Maximum capacity of the currently allocated index buffer.

**10.39.4.2 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:

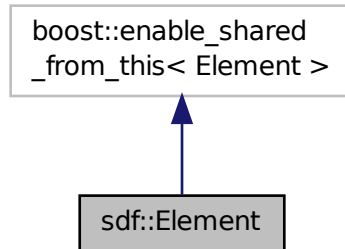
- **DynamicRenderable.hh**

**10.40 sdf::Element Class Reference**

**SDF** (p. 669) **Element** (p. 266) 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. 266).*
- **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. 669).*
- void **PrintDocRightPane** (std::string &\_html, int \_spacing, int &\_index)  
*Helper function for SDF::PrintDoc (p. 669).*
- 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** ()

### 10.40.1 Detailed Description

**SDF** (p. 669) **Element** (p. 266) class.

### 10.40.2 Constructor & Destructor Documentation

10.40.2.1 sdf::Element::Element ( )

10.40.2.2 virtual sdf::Element::~~Element ( ) [virtual]

### 10.40.3 Member Function Documentation

10.40.3.1 void sdf::Element::AddAttribute ( const std::string &\_key, const std::string &\_type, const std::string &\_defaultvalue, bool \_required, const std::string &\_description = " " )

10.40.3.2 ElementPtr sdf::Element::AddElement ( const std::string &\_name )

10.40.3.3 void sdf::Element::AddElementDescription ( ElementPtr \_elem )

Add a new element description.

10.40.3.4 void sdf::Element::AddValue ( const std::string &\_type, const std::string &\_defaultValue, bool \_required, const std::string &\_description = " " )

10.40.3.5 void sdf::Element::ClearElements ( )

Remove all child elements.

10.40.3.6 **Element\*** sdf::Element::Clone ( ) const

10.40.3.7 void sdf::Element::Copy ( const ElementPtr *\_elem* )

Copy values from an **Element** (p. 266).

10.40.3.8 **ParamPtr** sdf::Element::GetAttribute ( const std::string & *\_key* )

Get the param of an attribute.

#### Parameters

<i>_key</i>	the name of the attribute
-------------	---------------------------

10.40.3.9 **ParamPtr** sdf::Element::GetAttribute ( unsigned int *\_index* ) const

Get an attribute using an index.

10.40.3.10 unsigned int sdf::Element::GetAttributeCount ( ) const

Get the number of attributes.

10.40.3.11 bool sdf::Element::GetAttributeSet ( const std::string & *\_key* )

Return true if the attribute was set (i.e. not default value)

10.40.3.12 bool sdf::Element::GetCopyChildren ( ) const

10.40.3.13 std::string sdf::Element::GetDescription ( ) const

Get a text description of the element.

10.40.3.14 **ElementPtr** sdf::Element::GetElement ( const std::string & *\_name* ) const

Referenced by gazebo::physics::ScrewJoint< T >::Load(), and gazebo::physics::Hinge2Joint< T >::Load().

10.40.3.15 **ElementPtr** sdf::Element::GetElement ( const std::string & *\_name* )

10.40.3.16 **ElementPtr** sdf::Element::GetElementDescription ( unsigned int *\_index* ) const

Get an element description using an index.

10.40.3.17 **ElementPtr** sdf::Element::GetElementDescription ( const std::string & *\_key* ) const

Get an element descriptio using a key.

10.40.3.18 unsigned int sdf::Element::GetElementDescriptionCount ( ) const

Get the number of element descriptions.

10.40.3.19 ElementPtr sdf::Element::GetElementImpl ( const std::string & \_name ) const

10.40.3.20 ElementPtr sdf::Element::GetFirstElement ( ) const

10.40.3.21 std::string sdf::Element::GetInclude ( ) const

10.40.3.22 const std::string& sdf::Element::GetName ( ) const

10.40.3.23 ElementPtr sdf::Element::GetNextElement ( const std::string & \_name = " " ) const

10.40.3.24 ElementPtr sdf::Element::GetParent ( ) const

10.40.3.25 const std::string& sdf::Element::GetRequired ( ) const

10.40.3.26 ParamPtr sdf::Element::GetValue ( )

Get the param of the elements value.

10.40.3.27 bool sdf::Element::GetValueBool ( const std::string & \_key = " " )

10.40.3.28 char sdf::Element::GetValueChar ( const std::string & \_key = " " )

10.40.3.29 gazebo::common::Color sdf::Element::GetValueColor ( const std::string & \_key = " " )

10.40.3.30 double sdf::Element::GetValueDouble ( const std::string & \_key = " " )

Referenced by gazebo::physics::ScrewJoint< T >::Load().

10.40.3.31 float sdf::Element::GetValueFloat ( const std::string & \_key = " " )

10.40.3.32 int sdf::Element::GetValueInt ( const std::string & \_key = " " )

10.40.3.33 gazebo::math::Pose sdf::Element::GetValuePose ( const std::string & \_key = " " )

10.40.3.34 gazebo::math::Quaternion sdf::Element::GetValueQuaternion ( const std::string & \_key = " " )

10.40.3.35 std::string sdf::Element::GetValueString ( const std::string & \_key = " " )

10.40.3.36 gazebo::common::Time sdf::Element::GetValueTime ( const std::string & \_key = " " )

10.40.3.37 unsigned int sdf::Element::GetValueUInt ( const std::string & \_key = " " )

10.40.3.38 gazebo::math::Vector2d sdf::Element::GetValueVector2d ( const std::string & \_key = " " )

10.40.3.39 gazebo::math::Vector3 sdf::Element::GetValueVector3 ( const std::string & \_key = " " )

Referenced by gazebo::physics::ScrewJoint< T >::Load(), and gazebo::physics::Hinge2Joint< T >::Load().

10.40.3.40 `bool sdf::Element::HasAttribute ( const std::string & _key )`

10.40.3.41 `bool sdf::Element::HasElement ( const std::string & _name ) const`

Referenced by `gazebo::physics::ScrewJoint< T >::Load()`.

10.40.3.42 `bool sdf::Element::HasElementDescription ( const std::string & _name )`

Return true if an element description exists.

10.40.3.43 `void sdf::Element::InsertElement ( ElementPtr _elem )`

10.40.3.44 `void sdf::Element::PrintDescription ( std::string _prefix )`

10.40.3.45 `void sdf::Element::PrintDocLeftPane ( std::string & _html, int _spacing, int & _index )`

Helper function for **SDF::PrintDoc** (p. 669).

This generates the **SDF** (p. 669) html documentation.

#### Parameters

out	<code>_html</code>	Accumulated HTML for output.
in	<code>_spacing</code>	Amount of spacing for this element.
in	<code>_index</code>	Unique index for this element.

10.40.3.46 `void sdf::Element::PrintDocRightPane ( std::string & _html, int _spacing, int & _index )`

Helper function for **SDF::PrintDoc** (p. 669).

This generates the **SDF** (p. 669) html documentation.

#### Parameters

out	<code>_html</code>	Accumulated HTML for output
in	<code>_spacing</code>	Amount of spacing for this element.

10.40.3.47 `void sdf::Element::PrintValues ( std::string _prefix )`

10.40.3.48 `void sdf::Element::PrintWiki ( std::string _prefix )`

10.40.3.49 `void sdf::Element::RemoveChild ( ElementPtr _child )`

Remove a child element.

#### Parameters

in	<code>_child</code>	Pointer to the child to remove.
----	---------------------	---------------------------------

10.40.3.50 void sdf::Element::RemoveFromParent ( )

Remove this element from its parent.

10.40.3.51 void sdf::Element::Reset ( )

10.40.3.52 bool sdf::Element::Set ( const bool & *\_value* )

10.40.3.53 bool sdf::Element::Set ( const int & *\_value* )

10.40.3.54 bool sdf::Element::Set ( const unsigned int & *\_value* )

10.40.3.55 bool sdf::Element::Set ( const float & *\_value* )

10.40.3.56 bool sdf::Element::Set ( const double & *\_value* )

10.40.3.57 bool sdf::Element::Set ( const char & *\_value* )

10.40.3.58 bool sdf::Element::Set ( const std::string & *\_value* )

10.40.3.59 bool sdf::Element::Set ( const char \* *\_value* )

10.40.3.60 bool sdf::Element::Set ( const gazebo::math::Vector3 & *\_value* )

10.40.3.61 bool sdf::Element::Set ( const gazebo::math::Vector2i & *\_value* )

10.40.3.62 bool sdf::Element::Set ( const gazebo::math::Vector2d & *\_value* )

10.40.3.63 bool sdf::Element::Set ( const gazebo::math::Quaternion & *\_value* )

10.40.3.64 bool sdf::Element::Set ( const gazebo::math::Pose & *\_value* )

10.40.3.65 bool sdf::Element::Set ( const gazebo::common::Color & *\_value* )

10.40.3.66 bool sdf::Element::Set ( const gazebo::common::Time & *\_value* )

10.40.3.67 void sdf::Element::SetCopyChildren ( bool *\_value* )

10.40.3.68 void sdf::Element::SetDescription ( const std::string & *\_desc* )

Set a text description for the element.

10.40.3.69 void sdf::Element::SetInclude ( const std::string & *\_filename* )

10.40.3.70 void sdf::Element::SetName ( const std::string & *\_name* )

10.40.3.71 void sdf::Element::SetParent ( const ElementPtr *\_parent* )

10.40.3.72 void sdf::Element::SetRequired ( const std::string & *\_req* )

10.40.3.73 `std::string sdf::Element::ToString ( const std::string & _prefix ) const`

10.40.3.74 `void sdf::Element::Update ( )`

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

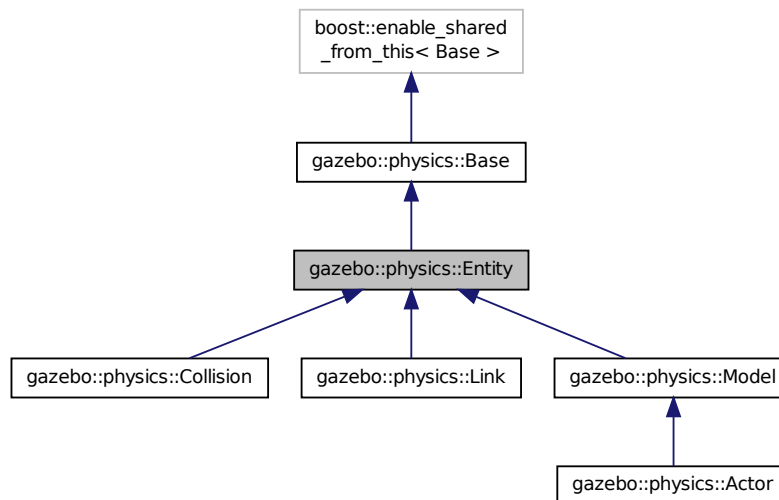
- **SDF.hh**

## 10.41 gazebo::physics::Entity Class Reference

**Base** (p. 133) 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. 285).*
  - **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. 107) 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. 274).*
- 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. 107) 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

### 10.41.1 Detailed Description

**Base** (p. 133) class for all physics objects in Gazebo.

### 10.41.2 Constructor & Destructor Documentation

#### 10.41.2.1 gazebo::physics::Entity::Entity ( BasePtr \_parent ) [explicit]

Constructor.

#### Parameters

in	<code>_parent</code>	Parent of the entity.
----	----------------------	-----------------------

#### 10.41.2.2 virtual gazebo::physics::Entity::~Entity ( ) [virtual]

Destructor.

### 10.41.3 Member Function Documentation

#### 10.41.3.1 virtual void gazebo::physics::Entity::Fini ( ) [virtual]

Finalize the entity.

Reimplemented from **gazebo::physics::Base** (p. 138).

Reimplemented in **gazebo::physics::Actor** (p. 110), **gazebo::physics::Model** (p. 473), **gazebo::physics::Link** (p. 411), and **gazebo::physics::Collision** (p. 193).

#### 10.41.3.2 virtual math::Box gazebo::physics::Entity::GetBoundingBox ( ) const [virtual]

Return the bounding box for the entity.

#### Returns

The bounding box.

Reimplemented in **gazebo::physics::Link** (p. 411), **gazebo::physics::Model** (p. 474), and **gazebo::physics::Collision** (p. 193).

#### 10.41.3.3 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. 190) object, or NULL if not found.

10.41.3.4 **LinkPtr** gazebo::physics::Entity::GetChildLink ( const std::string & \_name )

Get a child linked entity, if one exists.

## Parameters

in	_name	Name of the child <b>Link</b> (p. 404) object.
----	-------	--

## Returns

Pointer to the **Link** (p. 404) object, or NULL if not found.

10.41.3.5 **math::Box** gazebo::physics::Entity::GetCollisionBoundingBox ( ) const

Returns collision bounding box.

## Returns

Collision bounding box.

10.41.3.6 **const math::Pose&** gazebo::physics::Entity::GetDirtyPose ( ) const

Returns **Entity::dirtyPose** (p. 285).

The dirty pose is the pose set by the physics engine before its value is propagated to the rest of the simulator.

## Returns

The dirty pose of the entity.

10.41.3.7 **math::Pose** gazebo::physics::Entity::GetInitialRelativePose ( ) const

Get the initial relative pose.

## Returns

The initial relative pose.

10.41.3.8 void gazebo::physics::Entity::GetNearestEntityBelow ( double & *\_distBelow*, std::string & *\_entityName* )

Get the distance to the nearest entity below (along the Z-axis) this entity.

#### Parameters

out	<i>_distBelow</i>	The distance to the nearest entity below.
out	<i>_entityName</i>	The name of the nearest entity below.

10.41.3.9 ModelPtr gazebo::physics::Entity::GetParentModel ( )

Get the parent model, if one exists.

#### Returns

Pointer to a model, or NULL if no parent model exists.

10.41.3.10 virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularAccel ( ) const [inline],[virtual]

Get the angular acceleration of the entity.

#### Returns

**A** (p. 107) **math::Vector3** (p. 821) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 414), **gazebo::physics::Collision** (p. 194), and **gazebo::physics::Model** (p. 475).

10.41.3.11 virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularVel ( ) const [inline],[virtual]

Get the angular velocity of the entity.

#### Returns

**A** (p. 107) **math::Vector3** (p. 821) for the velocity.

Reimplemented in **gazebo::physics::Link** (p. 414), **gazebo::physics::Collision** (p. 194), and **gazebo::physics::Model** (p. 476).

10.41.3.12 virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearAccel ( ) const [inline],[virtual]

Get the linear acceleration of the entity.

#### Returns

**A** (p. 107) **math::Vector3** (p. 821) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 414), **gazebo::physics::Collision** (p. 195), and **gazebo::physics::Model** (p. 476).

10.41.3.13 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearVel ( ) const [inline],[virtual]`

Get the linear velocity of the entity.

#### Returns

**A** (p. 107) `math::Vector3` (p. 821) for the linear velocity.

Reimplemented in `gazebo::physics::Link` (p. 414), `gazebo::physics::Collision` (p. 195), and `gazebo::physics::Model` (p. 476).

10.41.3.14 `math::Pose gazebo::physics::Entity::GetRelativePose ( ) const`

Get the pose of the entity relative to its parent.

#### Returns

The pose of the entity relative to its parent.

10.41.3.15 `virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularAccel ( ) const [inline],[virtual]`

Get the angular acceleration of the entity in the world frame.

#### Returns

**A** (p. 107) `math::Vector3` (p. 821) for the acceleration.

Reimplemented in `gazebo::physics::Link` (p. 415), `gazebo::physics::Collision` (p. 196), and `gazebo::physics::Model` (p. 477).

10.41.3.16 `virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularVel ( ) const [inline],[virtual]`

Get the angular velocity of the entity in the world frame.

#### Returns

**A** (p. 107) `math::Vector3` (p. 821) for the velocity.

Reimplemented in `gazebo::physics::Collision` (p. 196), and `gazebo::physics::Model` (p. 477).

10.41.3.17 `virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearAccel ( ) const [inline],[virtual]`

Get the linear acceleration of the entity in the world frame.

#### Returns

**A** (p. 107) `math::Vector3` (p. 821) for the acceleration.

Reimplemented in `gazebo::physics::Link` (p. 416), `gazebo::physics::Collision` (p. 196), and `gazebo::physics::Model` (p. 477).

10.41.3.18 `virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearVel ( ) const [inline],[virtual]`

Get the linear velocity of the entity in the world frame.

#### Returns

**A** (p. 107) **math::Vector3** (p. 821) for the linear velocity.

Reimplemented in **gazebo::physics::Collision** (p. 196), and **gazebo::physics::Model** (p. 477).

10.41.3.19 `const math::Pose& gazebo::physics::Entity::GetWorldPose ( ) const [inline]`

Get the absolute pose of the entity.

#### Returns

The absolute pose of the entity.

Referenced by `gazebo::sensors::RFIDTag::GetTagPose()`.

10.41.3.20 `bool gazebo::physics::Entity::IsCanonicalLink ( ) const [inline]`

**A** (p. 107) helper function that checks if this is a canonical body.

#### Returns

True if the link is canonical.

10.41.3.21 `bool gazebo::physics::Entity::IsStatic ( ) const`

Return whether this entity is static.

#### Returns

True if static.

10.41.3.22 `virtual void gazebo::physics::Entity::Load ( sdf::ElementPtr _sdf ) [virtual]`

Load the entity.

#### Parameters

<code>in</code>	<code>_sdf</code>	Pointer to an SDF element.
-----------------	-------------------	----------------------------

Reimplemented from **gazebo::physics::Base** (p. 141).

Reimplemented in **gazebo::physics::Actor** (p. 111), **gazebo::physics::Model** (p. 477), **gazebo::physics::Link** (p. 416), and **gazebo::physics::Collision** (p. 197).

10.41.3.23 `virtual void gazebo::physics::Entity::OnPoseChange ( )` [protected],[pure virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implemented in `gazebo::physics::Link` (p. 417), and `gazebo::physics::Model` (p. 478).

10.41.3.24 `void gazebo::physics::Entity::PlaceOnEntity ( const std::string & _entityName )`

Move this entity to be ontop of another entity by name.

#### Parameters

<code>in</code>	<code>_entityName</code>	Name of the <b>Entity</b> (p. 274) this <b>Entity</b> (p. 274) should be ontop of.
-----------------	--------------------------	--

10.41.3.25 `void gazebo::physics::Entity::PlaceOnNearestEntityBelow ( )`

Move this entity to be ontop of the nearest entity below.

10.41.3.26 `virtual void gazebo::physics::Entity::Reset ( )` [virtual]

Reset the entity.

Reimplemented from `gazebo::physics::Base` (p. 142).

Reimplemented in `gazebo::physics::Model` (p. 478), and `gazebo::physics::Link` (p. 417).

10.41.3.27 `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.

10.41.3.28 `void gazebo::physics::Entity::SetAnimation ( common::PoseAnimationPtr _anim )`

Set an animation for this entity.

#### Parameters

<code>in</code>	<code>_anim</code>	Pose animation.
-----------------	--------------------	-----------------

10.41.3.29 `void gazebo::physics::Entity::SetCanonicalLink ( bool _value )`

Set to true if this entity is a canonical link for a model.

## Parameters

in	<i>_value</i>	True if the link is canonical.
----	---------------	--------------------------------

10.41.3.30 void gazebo::physics::Entity::SetInitialRelativePose ( const math::Pose & *\_pose* )

Set the initial pose.

## Parameters

in	<i>_pose</i>	The initial pose.
----	--------------	-------------------

10.41.3.31 virtual void gazebo::physics::Entity::SetName ( const std::string & *\_name* ) [virtual]

Set the name of the entity.

## Parameters

in	<i>_name</i>	The new name.
----	--------------	---------------

Reimplemented from **gazebo::physics::Base** (p. 143).

10.41.3.32 void gazebo::physics::Entity::SetRelativePose ( const math::Pose & *\_pose*, bool *\_notify* = true, bool *\_publish* = true )

Set the pose of the entity relative to its parent.

## Parameters

in	<i>_pose</i>	The new pose.
in	<i>_notify</i>	True = tell children of the pose change.
in	<i>_publish</i>	True to publish the pose.

10.41.3.33 void gazebo::physics::Entity::SetStatic ( const bool & *\_static* )

Set whether this entity is static: immovable.

## Parameters

in	<i>_static</i>	True = static.
----	----------------	----------------

10.41.3.34 void gazebo::physics::Entity::SetWorldPose ( const math::Pose & *\_pose*, bool *\_notify* = true, bool *\_publish* = true )

Set the world pose of the entity.

## Parameters

in	<i>_pose</i>	The new world pose.
in	<i>_notify</i>	True = tell children of the pose change.

in	<code>_publish</code>	True to publish the pose.
----	-----------------------	---------------------------

10.41.3.35 `void gazebo::physics::Entity::SetWorldTwist ( const math::Vector3 & _linear, const math::Vector3 & _angular, bool _updateChildren = true )`

Set angular and linear rates of an **physics::Entity** (p. 274).

#### Parameters

in	<code>_linear</code>	Linear twist.
in	<code>_angular</code>	Angular twist.
in	<code>_updateChildren</code>	True to pass this update to child entities.

10.41.3.36 `virtual void gazebo::physics::Entity::StopAnimation ( ) [virtual]`

Stop the current animation, if any.

Reimplemented in **gazebo::physics::Model** (p. 481).

10.41.3.37 `virtual void gazebo::physics::Entity::UpdateParameters ( sdf::ElementPtr _sdf ) [virtual]`

Update the parameters using new sdf values.

#### Parameters

in	<code>_sdf</code>	SDF to update from.
----	-------------------	---------------------

Reimplemented from **gazebo::physics::Base** (p. 144).

Reimplemented in **gazebo::physics::Actor** (p. 111), **gazebo::physics::Model** (p. 482), **gazebo::physics::Link** (p. 421), and **gazebo::physics::Collision** (p. 198).

## 10.41.4 Member Data Documentation

10.41.4.1 `common::PoseAnimationPtr gazebo::physics::Entity::animation [protected]`

Current pose animation.

10.41.4.2 `event::ConnectionPtr gazebo::physics::Entity::animationConnection [protected]`

Connection used to update an animation.

10.41.4.3 `math::Pose gazebo::physics::Entity::animationStartPose [protected]`

Start pose of an animation.



10.41.4.4 `std::vector<event::ConnectionPtr> gazebo::physics::Entity::connections` [protected]

All our event connections.

10.41.4.5 `math::Pose gazebo::physics::Entity::dirtyPose` [protected]

The pose set by a physics engine.

10.41.4.6 `transport::NodePtr gazebo::physics::Entity::node` [protected]

Communication node.

10.41.4.7 `EntityPtr gazebo::physics::Entity::parentEntity` [protected]

**A** (p. 107) helper that prevents numerous `dynamic_casts`.

10.41.4.8 `msgs::Pose* gazebo::physics::Entity::poseMsg` [protected]

Pose message container.

10.41.4.9 `common::Time gazebo::physics::Entity::prevAnimationTime` [protected]

Previous time an animation was updated.

10.41.4.10 `transport::PublisherPtr gazebo::physics::Entity::requestPub` [protected]

Request publisher.

10.41.4.11 `transport::PublisherPtr gazebo::physics::Entity::visPub` [protected]

Visual publisher.

10.41.4.12 `msgs::Visual* gazebo::physics::Entity::visualMsg` [protected]

Visual message container.

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

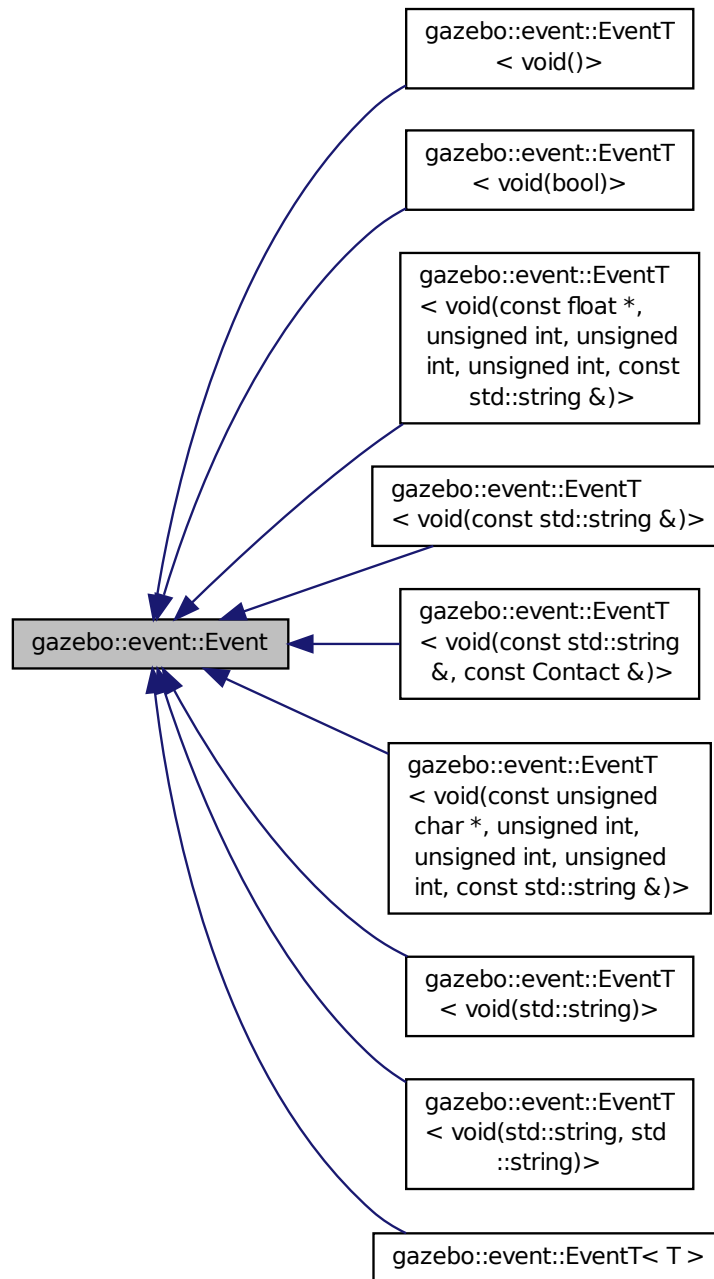
- **Entity.hh**

## 10.42 gazebo::event::Event Class Reference

Base class for all events.

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

Inheritance diagram for gazebo::event::Event:



## Public Member Functions

- virtual `~Event()`

*Constructor.*

- virtual void **Disconnect** (**ConnectionPtr** \_c)=0

*Disconnect.*

- virtual void **Disconnect** (int \_id)=0

*Disconnect.*

### 10.42.1 Detailed Description

Base class for all events.

### 10.42.2 Constructor & Destructor Documentation

10.42.2.1 virtual gazebo::event::Event::~Event ( ) [inline],[virtual]

Constructor.

### 10.42.3 Member Function Documentation

10.42.3.1 virtual void gazebo::event::Event::Disconnect ( **ConnectionPtr** \_c ) [pure virtual]

Disconnect.

Parameters

in	_c	<b>A</b> (p. 107) 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 float \*, 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).

10.42.3.2 virtual void gazebo::event::Event::Disconnect ( int \_id ) [pure virtual]

Disconnect.

Parameters

in	_id	Integer ID of a connection
----	-----	----------------------------

Implemented in **gazebo::event::EventT< T >** (p. 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 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**

## 10.43 gazebo::rendering::Events Class Reference

Base class for rendering events.

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

### Static Public Member Functions

- `template<typename T >`  
**static event::ConnectionPtr ConnectCreateScene** (T \_subscriber)  
*Connect to a scene created event.*
- `template<typename T >`  
**static event::ConnectionPtr ConnectRemoveScene** (T \_subscriber)  
*Connect to a scene removed event.*
- **static void DisconnectCreateScene** (event::ConnectionPtr \_connection)  
*Disconnect from a scene created event.*
- **static void DisconnectRemoveScene** (event::ConnectionPtr \_connection)  
*Disconnect from a scene removed event.*

### Static Public Attributes

- **static event::EventT** < void(const std::string &)> **createScene**  
*The event used to trigger a create scene event.*
- **static event::EventT** < void(const std::string &)> **removeScene**  
*The event used to trigger a remove scene event.*

#### 10.43.1 Detailed Description

Base class for rendering events.

#### 10.43.2 Member Function Documentation

10.43.2.1 `template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectCreateScene ( T _subscriber ) [inline],[static]`

Connect to a scene created event.

##### Parameters

<code>in</code>	<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 `createScene`.

10.43.2.2 `template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectRemoveScene ( T _subscriber ) [inline],[static]`

Connect to a scene removed event.

#### Parameters

in	_subscriber	Callback to trigger when event occurs.
----	-------------	--

#### Returns

Pointer the connection. This must stay in scope.

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

10.43.2.3 `static void gazebo::rendering::Events::DisconnectCreateScene ( event::ConnectionPtr _connection ) [inline],[static]`

Disconnect from a scene created event.

#### Parameters

in	_connection	The connection to disconnect.
----	-------------	-------------------------------

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

10.43.2.4 `static void gazebo::rendering::Events::DisconnectRemoveScene ( event::ConnectionPtr _connection ) [inline],[static]`

Disconnect from a scene removed event.

#### Parameters

in	_connection	The connection to disconnect.
----	-------------	-------------------------------

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

### 10.43.3 Member Data Documentation

10.43.3.1 `event::EventT<void (const std::string &> gazebo::rendering::Events::createScene [static]`

The event used to trigger a create scene event.

Referenced by ConnectCreateScene(), and DisconnectCreateScene().

10.43.3.2 `event::EventT<void (const std::string &> gazebo::rendering::Events::removeScene [static]`

The event used to trigger a remove scene event.

Referenced by ConnectRemoveScene(), and DisconnectRemoveScene().

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

- **RenderEvents.hh**

## 10.44 gazebo::event::Events Class Reference

An **Event** (p. 285) class to get notifications for simulator events.

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

### Static Public Member Functions

- `template<typename T >`  
**static ConnectionPtr ConnectAddEntity** (T \_subscriber)  
*Connect a boost::slot the the add entity signal.*
- `template<typename T >`  
**static ConnectionPtr ConnectCreateEntity** (T \_subscriber)  
*Connect a boost::slot the the add entity signal.*
- `template<typename T >`  
**static ConnectionPtr ConnectDeleteEntity** (T \_subscriber)  
*Connect a boost::slot the delete entity.*
- `template<typename T >`  
**static ConnectionPtr ConnectDiagTimerStart** (T \_subscriber)  
*Connect a boost::slot the diagnostic timer start signal.*
- `template<typename T >`  
**static ConnectionPtr ConnectDiagTimerStop** (T \_subscriber)  
*Connect a boost::slot the diagnostic timer stop signal.*
- `template<typename T >`  
**static ConnectionPtr ConnectPause** (T \_subscriber)  
*Connect a boost::slot the the pause signal.*
- `template<typename T >`  
**static ConnectionPtr ConnectPostRender** (T \_subscriber)  
*Connect a boost::slot the post render update signal.*
- `template<typename T >`  
**static ConnectionPtr ConnectPreRender** (T \_subscriber)  
*Render start signal.*
- `template<typename T >`  
**static ConnectionPtr ConnectRender** (T \_subscriber)  
*Connect a boost::slot the render update signal.*
- `template<typename T >`  
**static ConnectionPtr ConnectSetSelectedEntity** (T \_subscriber)  
*Connect a boost::slot the set selected entity.*
- `template<typename T >`  
**static ConnectionPtr 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 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 **DisconnectWorldUpdateEnd** (**ConnectionPtr** \_subscriber)  
*Disconnect a boost::slot the the world update end signal.*
  - static void **DisconnectWorldUpdateStart** (**ConnectionPtr** \_subscriber)  
*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. 107) world has been created.*
- static **EventT**< void()> **worldUpdateEnd**  
*World update has ended.*
- static **EventT**< void()> **worldUpdateStart**  
*World update has started.*

### 10.44.1 Detailed Description

An **Event** (p. 285) class to get notifications for simulator events.

### 10.44.2 Member Function Documentation

10.44.2.1 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectAddEntity ( T _subscriber )`  
[inline], [static]

Connect a boost::slot the the add entity signal.

#### Parameters

in	<code><i>_subscriber</i></code>	the subscriber to this event
----	---------------------------------	------------------------------

#### Returns

a connection

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



10.44.2.2 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectCreateEntity ( T _subscriber )`  
`[inline],[static]`

Connect a boost::slot the the add entity signal.

#### Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

#### Returns

a connection

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

10.44.2.3 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDeleteEntity ( T _subscriber )`  
`[inline],[static]`

Connect a boost::slot the delete entity.

#### Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

#### Returns

a connection

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

10.44.2.4 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStart ( T _subscriber )`  
`[inline],[static]`

Connect a boost::slot the diagnostic timer start signal.

#### Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

#### Returns

a connection

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

10.44.2.5 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStop ( T _subscriber )`  
`[inline],[static]`

Connect a boost::slot the diagnostic timer stop signal.

## Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

## Returns

a connection

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

10.44.2.6 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPause ( T _subscriber ) [inline], [static]`

Connect a boost::slot the the pause signal.

## Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

## Returns

a connection

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

10.44.2.7 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPostRender ( T _subscriber ) [inline], [static]`

Connect a boost::slot the post render update signal.

## Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

## Returns

a connection

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

10.44.2.8 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPreRender ( T _subscriber ) [inline], [static]`

Render start signal.

## Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

## Returns

a connection

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

**10.44.2.9** `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectRender ( T _subscriber )`  
`[inline], [static]`

Connect a boost::slot the render update signal.

#### Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

#### Returns

a connection

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

**10.44.2.10** `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectSetSelectedEntity ( T _subscriber )`  
`[inline], [static]`

Connect a boost::slot the set selected entity.

#### Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

#### Returns

a connection

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

**10.44.2.11** `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStep ( T _subscriber )` `[inline],`  
`[static]`

Connect a boost::slot the the step 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 step.

**10.44.2.12** `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStop ( T _subscriber )` `[inline],`  
`[static]`

Connect a boost::slot the the stop signal.

## Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

## Returns

a connection

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

**10.44.2.13** `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldCreated ( T _subscriber )`  
`[inline], [static]`

Connect a boost::slot the the world created signal.

## Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

## Returns

a connection

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

**10.44.2.14** `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.

**10.44.2.15** `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(), and worldUpdateStart.

10.44.2.16 `static void gazebo::event::Events::DisconnectAddEntity ( ConnectionPtr _subscriber ) [inline], [static]`

Disconnect a boost::slot the the add entity signal.

#### Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

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

10.44.2.17 `static void gazebo::event::Events::DisconnectCreateEntity ( ConnectionPtr _subscriber ) [inline], [static]`

Disconnect a boost::slot the the add entity signal.

#### Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

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

10.44.2.18 `static void gazebo::event::Events::DisconnectDeleteEntity ( ConnectionPtr _subscriber ) [inline], [static]`

Disconnect a boost::slot the delete entity.

#### Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

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

10.44.2.19 `static void gazebo::event::Events::DisconnectDiagTimerStart ( ConnectionPtr _subscriber ) [inline], [static]`

Disconnect a boost::slot the diagnostic timer start signal.

#### Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

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

10.44.2.20 `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()`.

**10.44.2.21** `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`.

**10.44.2.22** `static void gazebo::event::Events::DisconnectPostRender ( ConnectionPtr _subscriber ) [inline],[static]`

Disconnect a boost::slot the post render update signal.

## Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

**10.44.2.23** `static void gazebo::event::Events::DisconnectPreRender ( ConnectionPtr _subscriber ) [inline],[static]`

Disconnect a render start signal.

## Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

**10.44.2.24** `static void gazebo::event::Events::DisconnectRender ( ConnectionPtr _subscriber ) [inline],[static]`

Disconnect a boost::slot the render update signal.

## Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

**10.44.2.25** `static void gazebo::event::Events::DisconnectSetSelectedEntity ( ConnectionPtr _subscriber ) [inline],[static]`

Disconnect a boost::slot the set selected entity.

## Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

**10.44.2.26** static void gazebo::event::Events::DisconnectStep ( ConnectionPtr *\_subscriber* ) [inline],[static]

Disconnect a boost::slot the the step signal.

## Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

**10.44.2.27** static void gazebo::event::Events::DisconnectStop ( ConnectionPtr *\_subscriber* ) [inline],[static]

Disconnect a boost::slot the the stop signal.

## Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

**10.44.2.28** static void gazebo::event::Events::DisconnectWorldCreated ( ConnectionPtr *\_subscriber* ) [inline],[static]

Disconnect a boost::slot the the world created signal.

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

**10.44.2.29** 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.

**10.44.2.30** static void gazebo::event::Events::DisconnectWorldUpdateStart ( ConnectionPtr *\_subscriber* ) [inline],[static]

Disconnect a boost::slot the the world update start signal.

## Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

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

### 10.44.3 Member Data Documentation

#### 10.44.3.1 EventT<void (std::string)> gazebo::event::Events::addEntity [static]

An entity has been added.

Referenced by ConnectAddEntity(), and DisconnectAddEntity().

#### 10.44.3.2 EventT<void (std::string)> gazebo::event::Events::deleteEntity [static]

An entity has been deleted.

Referenced by ConnectDeleteEntity(), and DisconnectDeleteEntity().

#### 10.44.3.3 EventT<void (std::string)> gazebo::event::Events::diagTimerStart [static]

Diagnostic timer start.

Referenced by ConnectDiagTimerStart(), and DisconnectDiagTimerStart().

#### 10.44.3.4 EventT<void (std::string)> gazebo::event::Events::diagTimerStop [static]

Diagnostic timer stop.

Referenced by ConnectDiagTimerStop(), and DisconnectDiagTimerStop().

#### 10.44.3.5 EventT<void (std::string)> gazebo::event::Events::entityCreated [static]

An entity has been created.

Referenced by ConnectCreateEntity(), and DisconnectCreateEntity().

#### 10.44.3.6 EventT<void (bool)> gazebo::event::Events::pause [static]

Pause signal.

Referenced by ConnectPause(), and DisconnectPause().

#### 10.44.3.7 EventT<void ()> gazebo::event::Events::postRender [static]

Post-Render.

Referenced by ConnectPostRender(), and DisconnectPostRender().



**10.44.3.8** `EventT<void ()> gazebo::event::Events::preRender` [static]

Pre-render.

Referenced by `ConnectPreRender()`, and `DisconnectPreRender()`.

**10.44.3.9** `EventT<void ()> gazebo::event::Events::render` [static]

Render.

Referenced by `ConnectRender()`, and `DisconnectRender()`.

**10.44.3.10** `EventT<void (std::string, std::string)> gazebo::event::Events::setSelectedEntity` [static]

An entity has been selected.

Referenced by `ConnectSetSelectedEntity()`, and `DisconnectSetSelectedEntity()`.

**10.44.3.11** `EventT<void ()> gazebo::event::Events::step` [static]

Step the simulation once signal.

Referenced by `ConnectStep()`, and `DisconnectStep()`.

**10.44.3.12** `EventT<void ()> gazebo::event::Events::stop` [static]

Simulation stop signal.

Referenced by `ConnectStop()`, and `DisconnectStop()`.

**10.44.3.13** `EventT<void (std::string)> gazebo::event::Events::worldCreated` [static]

**A** (p. 107) world has been created.

Referenced by `ConnectWorldCreated()`, and `DisconnectWorldCreated()`.

**10.44.3.14** `EventT<void ()> gazebo::event::Events::worldUpdateEnd` [static]

World update has ended.

Referenced by `ConnectWorldUpdateEnd()`, and `DisconnectWorldUpdateEnd()`.

**10.44.3.15** `EventT<void ()> gazebo::event::Events::worldUpdateStart` [static]

World update has started.

Referenced by `ConnectWorldUpdateStart()`, and `DisconnectWorldUpdateStart()`.

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

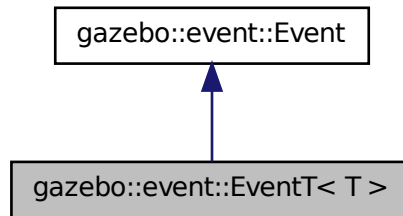
- **Events.hh**

## 10.45 gazebo::event::EventT< T > Class Template Reference

**A** (p. 107) class for event processing.

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

Inheritance diagram for gazebo::event::EventT< T >:



### Public Member Functions

- virtual `~EventT ()`  
*Destructor.*
- **ConnectionPtr Connect** (const boost::function< T > &\_subscriber)  
*Connect a callback to this event.*
- unsigned int **ConnectionCount** () const  
*Get the number of connections.*
- virtual void **Disconnect** (**ConnectionPtr** \_c)  
*Disconnect a callback to this event.*
- virtual void **Disconnect** (int \_id)  
*Disconnect a callback to this event.*
- void **operator()** ()  
*Access the signal.*
- template<typename P >  
void **operator()** (const P &\_p)  
*Signal the event with one parameter.*
- template<typename P1 , typename P2 >  
void **operator()** (const P1 &\_p1, const P2 &\_p2)  
*Signal the event with two parameters.*
- template<typename P1 , typename P2 , typename P3 >  
void **operator()** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3)  
*Signal the event with three parameters.*
- template<typename P1 , typename P2 , typename P3 , typename P4 >  
void **operator()** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3, const P4 &\_p4)  
*Signal the event with four parameters.*
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 >  
void **operator()** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3, const P4 &\_p4, const P5 &\_p5)

*Signal the event with five parameters.*

- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 >`  
void **operator()** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3, const P4 &\_p4, const P5 &\_p5, const P6 &\_p6)

*Signal the event with six parameters.*

- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 >`  
void **operator()** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3, const P4 &\_p4, const P5 &\_p5, const P6 &\_p6, const P7 &\_p7)

*Signal the event with seven parameters.*

- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 >`  
void **operator()** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3, const P4 &\_p4, const P5 &\_p5, const P6 &\_p6, const P7 &\_p7, const P8 &\_p8)

*Signal the event with eight parameters.*

- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 >`  
void **operator()** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3, const P4 &\_p4, const P5 &\_p5, const P6 &\_p6, const P7 &\_p7, const P8 &\_p8, const P9 &\_p9)

*Signal the event with nine parameters.*

- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 >`  
void **operator()** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3, const P4 &\_p4, const P5 &\_p5, const P6 &\_p6, const P7 &\_p7, const P8 &\_p8, const P9 &\_p9, const P10 &\_p10)

*Signal the event with ten parameters.*

- void **Signal** ()

*Signal the event for all subscribers.*

- `template<typename P >`  
void **Signal** (const P &\_p)

*Signal the event with one parameter.*

- `template<typename P1 , typename P2 >`  
void **Signal** (const P1 &\_p1, const P2 &\_p2)

*Signal the event with two parameter.*

- `template<typename P1 , typename P2 , typename P3 >`  
void **Signal** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3)

*Signal the event with three parameter.*

- `template<typename P1 , typename P2 , typename P3 , typename P4 >`  
void **Signal** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3, const P4 &\_p4)

*Signal the event with four parameter.*

- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 >`  
void **Signal** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3, const P4 &\_p4, const P5 &\_p5)

*Signal the event with five parameter.*

- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 >`  
void **Signal** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3, const P4 &\_p4, const P5 &\_p5, const P6 &\_p6)

*Signal the event with six parameter.*

- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 >`  
void **Signal** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3, const P4 &\_p4, const P5 &\_p5, const P6 &\_p6, const P7 &\_p7)

*Signal the event with seven parameter.*

- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 >`  
void **Signal** (const P1 &\_p1, const P2 &\_p2, const P3 &\_p3, const P4 &\_p4, const P5 &\_p5, const P6 &\_p6, const P7 &\_p7, const P8 &\_p8)

*Signal the event with eight parameter.*

- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 >`  
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9)`  
*Signal the event with nine parameter.*
- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 >`  
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9, const P10 &_p10)`  
*Signal the event with ten parameter.*

### 10.45.1 Detailed Description

`template<typename T>class gazebo::event::EventT< T >`

**A** (p. 107) class for event processing.

### 10.45.2 Member Function Documentation

10.45.2.1 `template<typename T> void gazebo::event::EventT< T >::operator()( ) [inline]`

Access the signal.

10.45.2.2 `template<typename T> template<typename P > void gazebo::event::EventT< T >::operator()( const P & _p ) [inline]`

Signal the event with one parameter.

#### Parameters

in	_p	the parameter
----	----	---------------

10.45.2.3 `template<typename T> template<typename P1 , typename P2 > void gazebo::event::EventT< T >::operator()( const P1 & _p1, const P2 & _p2 ) [inline]`

Signal the event with two parameters.

#### Parameters

in	_p1	the first parameter
in	_p2	the second parameter

10.45.2.4 `template<typename T> template<typename P1 , typename P2 , typename P3 > void gazebo::event::EventT< T >::operator()( const P1 & _p1, const P2 & _p2, const P3 & _p3 ) [inline]`

Signal the event with three parameters.

## Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter

10.45.2.5 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 > void gazebo::event::EventT< T >::operator() ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4 ) [inline]`

Signal the event with four parameters.

## Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter

10.45.2.6 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 > void gazebo::event::EventT< T >::operator() ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5 ) [inline]`

Signal the event with five parameters.

## Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter
in	<code>_p5</code>	the fift parameter

10.45.2.7 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 > void gazebo::event::EventT< T >::operator() ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6 ) [inline]`

Signal the event with six parameters.

## Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter
in	<code>_p5</code>	the fift parameter
in	<code>_p6</code>	the sixt parameter

10.45.2.8 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 > void gazebo::event::EventT< T >::operator() ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7 ) [inline]`

Signal the event with seven parameters.

#### Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter
in	<code>_p5</code>	the fifth parameter
in	<code>_p6</code>	the sixth parameter
in	<code>_p7</code>	the seventh parameter

10.45.2.9 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 > void gazebo::event::EventT< T >::operator() ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8 ) [inline]`

Signal the event with eight parameters.

#### Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter
in	<code>_p5</code>	the fifth parameter
in	<code>_p6</code>	the sixth parameter
in	<code>_p7</code>	the seventh parameter
in	<code>_p8</code>	the eighth parameter

10.45.2.10 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 > void gazebo::event::EventT< T >::operator() ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9 ) [inline]`

Signal the event with nine parameters.

#### Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter
in	<code>_p5</code>	the fifth parameter
in	<code>_p6</code>	the sixth parameter
in	<code>_p7</code>	the seventh parameter
in	<code>_p8</code>	the eighth parameter
in	<code>_p9</code>	the ninth parameter

10.45.2.11 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 > void gazebo::event::EventT< T >::operator() ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9, const P10 & _p10 ) [inline]`

Signal the event with ten parameters.

#### Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter
in	<code>_p5</code>	the fifth parameter
in	<code>_p6</code>	the sixth parameter
in	<code>_p7</code>	the seventh parameter
in	<code>_p8</code>	the eighth parameter
in	<code>_p9</code>	the ninth parameter
in	<code>_p10</code>	the tenth parameter

10.45.2.12 `template<typename T> void gazebo::event::EventT< T >::Signal ( ) [inline]`

Signal the event for all subscribers.

Referenced by `gazebo::event::EventT< void(bool)>::operator()()`.

10.45.2.13 `template<typename T> template<typename P > void gazebo::event::EventT< T >::Signal ( const P & _p ) [inline]`

Signal the event with one parameter.

#### Parameters

in	<code>_p</code>	parameter
----	-----------------	-----------

10.45.2.14 `template<typename T> template<typename P1 , typename P2 > void gazebo::event::EventT< T >::Signal ( const P1 & _p1, const P2 & _p2 ) [inline]`

Signal the event with two parameter.

#### Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter

10.45.2.15 `template<typename T> template<typename P1 , typename P2 , typename P3 > void gazebo::event::EventT< T >::Signal ( const P1 & _p1, const P2 & _p2, const P3 & _p3 ) [inline]`

Signal the event with three parameter.

## Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter

10.45.2.16 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 > void gazebo::event::EventT< T >::Signal ( const P1 & .p1, const P2 & .p2, const P3 & .p3, const P4 & .p4 ) [inline]`

Signal the event with four parameter.

## Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter

10.45.2.17 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 > void gazebo::event::EventT< T >::Signal ( const P1 & .p1, const P2 & .p2, const P3 & .p3, const P4 & .p4, const P5 & .p5 ) [inline]`

Signal the event with five parameter.

## Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter
in	<code>_p5</code>	the fifth parameter

10.45.2.18 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 > void gazebo::event::EventT< T >::Signal ( const P1 & .p1, const P2 & .p2, const P3 & .p3, const P4 & .p4, const P5 & .p5, const P6 & .p6 ) [inline]`

Signal the event with six parameter.

## Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter
in	<code>_p5</code>	the fifth parameter
in	<code>_p6</code>	the sixth parameter



10.45.2.19 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 > void gazebo::event::EventT< T >::Signal ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7 ) [inline]`

Signal the event with seven parameter.

#### Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter
in	<code>_p5</code>	the fifth parameter
in	<code>_p6</code>	the sixth parameter
in	<code>_p7</code>	the seventh parameter

10.45.2.20 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 > void gazebo::event::EventT< T >::Signal ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8 ) [inline]`

Signal the event with eight parameter.

#### Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter
in	<code>_p5</code>	the fifth parameter
in	<code>_p6</code>	the sixth parameter
in	<code>_p7</code>	the seventh parameter
in	<code>_p8</code>	the eighth parameter

10.45.2.21 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 > void gazebo::event::EventT< T >::Signal ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9 ) [inline]`

Signal the event with nine parameter.

#### Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter
in	<code>_p5</code>	the fifth parameter
in	<code>_p6</code>	the sixth parameter
in	<code>_p7</code>	the seventh parameter
in	<code>_p8</code>	the eighth parameter
in	<code>_p9</code>	the ninth parameter

10.45.2.22 `template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 > void gazebo::event::EventT< T >::Signal ( const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9, const P10 & _p10 ) [inline]`

Signal the event with ten parameter.

#### Parameters

in	<code>_p1</code>	the first parameter
in	<code>_p2</code>	the second parameter
in	<code>_p3</code>	the second parameter
in	<code>_p4</code>	the first parameter
in	<code>_p5</code>	the fifth parameter
in	<code>_p6</code>	the sixth parameter
in	<code>_p7</code>	the seventh parameter
in	<code>_p8</code>	the eighth parameter
in	<code>_p9</code>	the ninth parameter
in	<code>_p10</code>	the tenth parameter

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

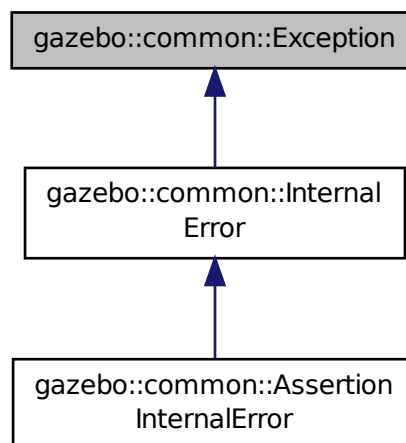
- **Event.hh**

## 10.46 gazebo::common::Exception Class Reference

Class for generating exceptions.

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

Inheritance diagram for gazebo::common::Exception:



## Public Member Functions

- **Exception** ()  
*Constructor.*
- **Exception** (const char \* \_file, int \_line, std::string \_msg)  
*Default constructor.*
- virtual **~Exception** ()  
*Destructor.*
- std::string **GetErrorFile** () const  
*Return the error function.*
- std::string **GetErrorStr** () const  
*Return the error string.*
- void **Print** () const  
*Print the exception to std out.*

## Friends

- std::ostream & **operator**<< (std::ostream &\_out, const gazebo::common::Exception &\_err)  
*stream insertion operator for Gazebo Error*

### 10.46.1 Detailed Description

Class for generating exceptions.

### 10.46.2 Constructor & Destructor Documentation

#### 10.46.2.1 gazebo::common::Exception::Exception ( )

Constructor.

#### 10.46.2.2 gazebo::common::Exception::Exception ( const char \* \_file, int \_line, std::string \_msg )

Default constructor.

#### Parameters

in	<i>_file</i>	File name
in	<i>_line</i>	Line number where the error occurred
in	<i>_msg</i>	Error message

#### 10.46.2.3 virtual gazebo::common::Exception::~~Exception ( ) [virtual]

Destructor.

### 10.46.3 Member Function Documentation

10.46.3.1 `std::string gazebo::common::Exception::GetErrorFile ( ) const`

Return the error function.

#### Returns

The error function name

10.46.3.2 `std::string gazebo::common::Exception::GetErrorStr ( ) const`

Return the error string.

#### Returns

The error string

10.46.3.3 `void gazebo::common::Exception::Print ( ) const`

Print the exception to std out.

## 10.46.4 Friends And Related Function Documentation

10.46.4.1 `std::ostream& operator<< ( std::ostream & _out, const gazebo::common::Exception & _err ) [friend]`

stream insertion operator for Gazebo Error

#### Parameters

<code>in</code>	<code>_out</code>	the output stream
<code>in</code>	<code>_err</code>	the exception

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

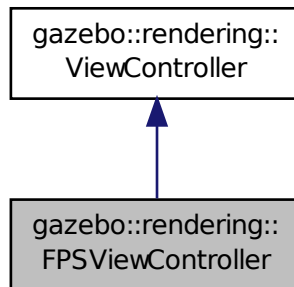
- **Exception.hh**

## 10.47 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

#### 10.47.1 Detailed Description

First Person Shooter style view controller.

## 10.47.2 Constructor & Destructor Documentation

### 10.47.2.1 gazebo::rendering::FPSViewController::FPSViewController ( UserCameraPtr *\_camera* )

Constructor.

#### Parameters

in	<b>Camera</b> (p. 157)	to controll
----	------------------------	-------------

### 10.47.2.2 virtual gazebo::rendering::FPSViewController::~~FPSViewController ( ) [virtual]

Destructor.

## 10.47.3 Member Function Documentation

### 10.47.3.1 static std::string gazebo::rendering::FPSViewController::GetTypeString ( ) [static]

Get the type name of this view controller.

#### Returns

The name of the controller type: "fps"

### 10.47.3.2 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. 848).

### 10.47.3.3 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. 849).

### 10.47.3.4 virtual void gazebo::rendering::FPSViewController::HandleMouseEvent ( const common::MouseEvent & *\_event* ) [virtual]

Handle a mouse event.

## Parameters

in	<code>_event</code>	The mouse position.
----	---------------------	---------------------

Implements `gazebo::rendering::ViewController` (p. 849).

10.47.3.5 `virtual void gazebo::rendering::FPSViewController::Init ( ) [virtual]`

Initialize the controller.

Implements `gazebo::rendering::ViewController` (p. 849).

10.47.3.6 `virtual void gazebo::rendering::FPSViewController::Update ( ) [virtual]`

Update the camera position.

Implements `gazebo::rendering::ViewController` (p. 850).

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

- `FPSViewController.hh`

## 10.48 urdf2gazebo::GazeboExtension Class Reference

```
#include <parser_urdf.hh>
```

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

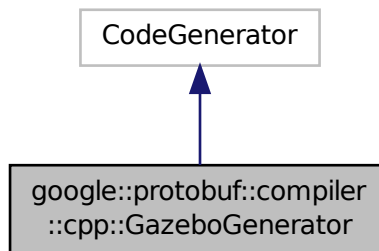
- `parser_urdf.hh`

## 10.49 google::protobuf::compiler::cpp::GazeboGenerator Class Reference

Google protobuf message generator for `gazebo::msgs` (p. 87).

```
#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 &parameter, OutputDirectory \*directory, string \*error) const

### 10.49.1 Detailed Description

Google protobuf message generator for **gazebo::msgs** (p. 87).

### 10.49.2 Constructor & Destructor Documentation

10.49.2.1 google::protobuf::compiler::cpp::GazeboGenerator::GazeboGenerator ( const std::string & *\_name* )

10.49.2.2 virtual google::protobuf::compiler::cpp::GazeboGenerator::~GazeboGenerator ( ) [virtual]

### 10.49.3 Member Function Documentation

10.49.3.1 virtual bool google::protobuf::compiler::cpp::GazeboGenerator::Generate ( const FileDescriptor \* *file*, const string & *parameter*, OutputDirectory \* *directory*, string \* *error* ) const [virtual]

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

- **GazeboGenerator.hh**

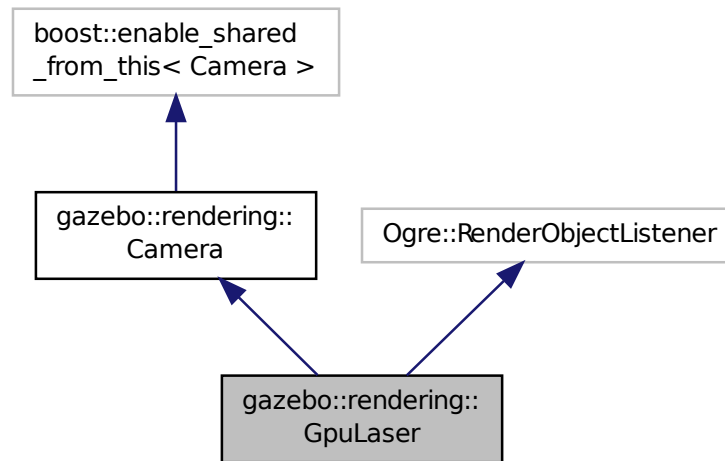
## 10.50 gazebo::rendering::GpuLaser Class Reference

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

### 10.50.1 Detailed Description

GPU based laser distance sensor.

### 10.50.2 Constructor & Destructor Documentation

10.50.2.1 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>	<b>Scene</b> (p. 651) that will contain the camera
in	<code>_autoRender</code>	Almost everyone should leave this as true.

10.50.2.2 virtual gazebo::rendering::GpuLaser::~GpuLaser ( ) [virtual]

Destructor.

### 10.50.3 Member Function Documentation

10.50.3.1 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. 107) pointer to the connection. This must be kept in scope.

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

10.50.3.2 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.
----	---------------------------	--------------------------

10.50.3.3 `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()`.

10.50.3.4 `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. 166).

10.50.3.5 `const float* gazebo::rendering::GpuLaser::GetLaserData ( )`

All things needed to get back z buffer for laser data.

## Returns

Array of laser data.

10.50.3.6 `virtual void gazebo::rendering::GpuLaser::Init ( ) [virtual]`

Initialize the camera.

Reimplemented from `gazebo::rendering::Camera` (p. 173).

10.50.3.7 `virtual void gazebo::rendering::GpuLaser::Load ( sdf::ElementPtr & _sdf ) [virtual]`

10.50.3.8 `virtual void gazebo::rendering::GpuLaser::Load ( ) [virtual]`

Load the camera with default parameters.

Reimplemented from `gazebo::rendering::Camera` (p. 174).

10.50.3.9 `virtual void gazebo::rendering::GpuLaser::notifyRenderSingleObject ( Ogre::Renderable * _rend, const Ogre::Pass * _p, const Ogre::AutoParamDataSource * _s, const Ogre::LightList * _ll, bool _supp ) [virtual]`

10.50.3.10 `virtual void gazebo::rendering::GpuLaser::PostRender ( ) [virtual]`

Post render.

Called after the render signal.

Reimplemented from `gazebo::rendering::Camera` (p. 175).

10.50.3.11 void gazebo::rendering::GpuLaser::SetParentSensor ( sensors::GpuRaySensor \* *\_parent* )

Set the parent sensor.

#### Parameters

in	<i>_parent</i>	Pointer to a <b>sensors::GpuRaySensor</b> (p. 320)
----	----------------	--

10.50.3.12 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

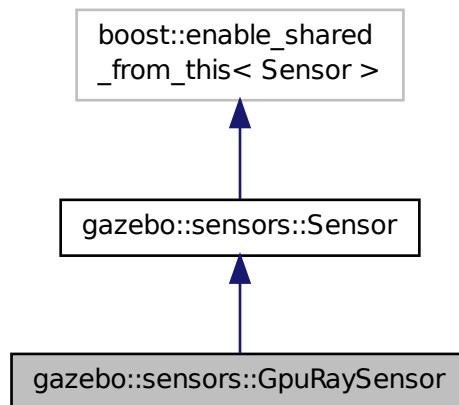
The documentation for this class was generated from the following file:

- **GpuLaser.hh**

## 10.51 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. 316).*

- 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.*

### 10.51.1 Constructor & Destructor Documentation

#### 10.51.1.1 gazebo::sensors::GpuRaySensor::GpuRaySensor ( )

Constructor.

#### 10.51.1.2 virtual gazebo::sensors::GpuRaySensor::~~GpuRaySensor ( ) [virtual]

Destructor.

## 10.51.2 Member Function Documentation

10.51.2.1 `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	<code>_subscriber</code>	Event callback.
----	--------------------------	-----------------

10.51.2.2 `void gazebo::sensors::GpuRaySensor::DisconnectNewLaserFrame ( event::ConnectionPtr & _conn )`

Disconnect Laser Frame.

### Parameters

in, out	<code>_conn</code>	Connection pointer to disconnect.
---------	--------------------	-----------------------------------

10.51.2.3 `virtual void gazebo::sensors::GpuRaySensor::Fini ( ) [protected],[virtual]`

Finalize the ray.

Reimplemented from `gazebo::sensors::Sensor` (p. 676).

10.51.2.4 `math::Angle gazebo::sensors::GpuRaySensor::GetAngleMax ( ) const`

Get the maximum angle.

### Returns

the maximum angle

10.51.2.5 `math::Angle gazebo::sensors::GpuRaySensor::GetAngleMin ( ) const`

Get the minimum angle.

### Returns

The minimum angle

10.51.2.6 `double gazebo::sensors::GpuRaySensor::GetAngleResolution ( ) const`

Get radians between each range.

10.51.2.7 `unsigned int gazebo::sensors::GpuRaySensor::GetCameraCount ( ) const`

Gets the camera count.



**Returns**

Number of cameras

**10.51.2.8 double gazebo::sensors::GpuRaySensor::GetCosHorzFOV ( ) const**

Get Cos Horz field-of-view.

**Returns**

$2 * \text{atan}(\tan(\text{this->hfov}/2) / \cos(\text{this->vfov}/2))$

**10.51.2.9 double gazebo::sensors::GpuRaySensor::GetCosVertFOV ( ) const**

Get Cos Vert field-of-view.

**Returns**

$2 * \text{atan}(\tan(\text{this->vfov}/2) / \cos(\text{this->hfov}/2))$

**10.51.2.10 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

**10.51.2.11 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.

**10.51.2.12 double gazebo::sensors::GpuRaySensor::GetHorzHalfAngle ( ) const**

Get  $(\text{horizontal\_max\_angle} + \text{horizontal\_min\_angle}) * 0.5$ .

**Returns**

$(\text{horizontal\_max\_angle} + \text{horizontal\_min\_angle}) * 0.5$

### 10.51.2.13 `rendering::GpuLaserPtr gazebo::sensors::GpuRaySensor::GetLaserCamera ( ) const` [inline]

Returns a pointer to the internally kept `rendering::GpuLaser` (p. 316).

**Returns**

Pointer to GpuLaser

### 10.51.2.14 `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.

### 10.51.2.15 `int gazebo::sensors::GpuRaySensor::GetRangeCount ( ) const`

Get the range count.

**Returns**

The number of ranges

### 10.51.2.16 `double gazebo::sensors::GpuRaySensor::GetRangeCountRatio ( ) const`

Return the ratio of horizontal range count to vertical range count.

**A** (p. 107) 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.

### 10.51.2.17 `double gazebo::sensors::GpuRaySensor::GetRangeMax ( ) const`

Get the maximum range.

**Returns**

The maximum range

10.51.2.18 `double gazebo::sensors::GpuRaySensor::GetRangeMin ( ) const`

Get the minimum range.

**Returns**

The minimum range

10.51.2.19 `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

10.51.2.20 `void gazebo::sensors::GpuRaySensor::GetRanges ( std::vector< double > & _ranges ) const`

Get all the ranges.

**Parameters**

out	<i>_range</i>	<b>A</b> (p. 107) vector that will contain all the range data
-----	---------------	---

10.51.2.21 `int gazebo::sensors::GpuRaySensor::GetRayCount ( ) const`

Get the ray count.

**Returns**

The number of rays

10.51.2.22 `double gazebo::sensors::GpuRaySensor::GetRayCountRatio ( ) const`

Return the ratio of horizontal ray count to vertical ray count.

**A** (p. 107) 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.

10.51.2.23 `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

**10.51.2.24** `double gazebo::sensors::GpuRaySensor::GetVertFOV ( ) const`

Get the vertical field-of-view.

**10.51.2.25** `double gazebo::sensors::GpuRaySensor::GetVertHalfAngle ( ) const`

Get  $(\text{vertical\_max\_angle} + \text{vertical\_min\_angle}) * 0.5$ .

### Returns

$(\text{vertical\_max\_angle} + \text{vertical\_min\_angle}) * 0.5$

**10.51.2.26** `math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMax ( ) const`

Get the vertical scan line top angle.

### Returns

The Maximum angle of the scan block

**10.51.2.27** `math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMin ( ) const`

Get the vertical scan bottom angle.

### Returns

The minimum angle of the scan block

**10.51.2.28** `int gazebo::sensors::GpuRaySensor::GetVerticalRangeCount ( ) const`

Get the vertical scan line count.

### Returns

The number of scan lines vertically

10.51.2.29 `int gazebo::sensors::GpuRaySensor::GetVerticalRayCount ( ) const`

Get the vertical scan line count.

#### Returns

The number of scan lines vertically

10.51.2.30 `virtual void gazebo::sensors::GpuRaySensor::Init ( ) [virtual]`

Initialize the ray.

Reimplemented from **`gazebo::sensors::Sensor`** (p. 678).

10.51.2.31 `bool gazebo::sensors::GpuRaySensor::IsHorizontal ( ) const`

Gets if sensor is horizontal.

#### Returns

True if horizontal, false if not

10.51.2.32 `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 <b><code>Sensor</code></b> (p. 672) parameters
<code>in</code>	<code>_worldName</code>	Name of world to load from

10.51.2.33 `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. 678).

10.51.2.34 `void gazebo::sensors::GpuRaySensor::SetAngleMax ( double _angle )`

Set the scan maximum angle.

#### Parameters

<code>in</code>	<code>_angle</code>	The maximum angle
-----------------	---------------------	-------------------

10.51.2.35 void gazebo::sensors::GpuRaySensor::SetAngleMin ( double *\_angle* )

Set the scan minimum angle.

Parameters

in	<i>_angle</i>	The minimum angle
----	---------------	-------------------

10.51.2.36 void gazebo::sensors::GpuRaySensor::SetVerticalAngleMax ( double *\_angle* )

Set the vertical scan line top angle.

Parameters

in	<i>_angle</i>	The Maximum angle of the scan block
----	---------------	-------------------------------------

10.51.2.37 void gazebo::sensors::GpuRaySensor::SetVerticalAngleMin ( double *\_angle* )

Set the vertical scan bottom angle.

Parameters

in	<i>_angle</i>	The minimum angle of the scan block
----	---------------	-------------------------------------

10.51.2.38 virtual void gazebo::sensors::GpuRaySensor::UpdateImpl ( bool *\_force* ) [protected],[virtual]

Update the sensor information.

Parameters

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

Reimplemented from **gazebo::sensors::Sensor** (p. 679).

### 10.51.3 Member Data Documentation

10.51.3.1 unsigned int gazebo::sensors::GpuRaySensor::cameraCount [protected]

Number of cameras.

10.51.3.2 sdf::ElementPtr gazebo::sensors::GpuRaySensor::cameraElem [protected]

Camera SDF element.

10.51.3.3 double gazebo::sensors::GpuRaySensor::chfov [protected]

Cos horizontal field-of-view.

10.51.3.4 `double gazebo::sensors::GpuRaySensor::cvfov` [protected]

Cos vertical field-of-view.

10.51.3.5 `double gazebo::sensors::GpuRaySensor::far` [protected]

Far clip plane.

10.51.3.6 `double gazebo::sensors::GpuRaySensor::hfov` [protected]

Horizontal field-of-view.

10.51.3.7 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::horzElem` [protected]

Horizontal SDF element.

10.51.3.8 `double gazebo::sensors::GpuRaySensor::horzHalfAngle` [protected]

Horizontal half angle.

10.51.3.9 `unsigned int gazebo::sensors::GpuRaySensor::horzRangeCount` [protected]

Horizontal range count.

10.51.3.10 `unsigned int gazebo::sensors::GpuRaySensor::horzRayCount` [protected]

Horizontal ray count.

10.51.3.11 `bool gazebo::sensors::GpuRaySensor::isHorizontal` [protected]

True if the sensor is horizontal only.

10.51.3.12 `double gazebo::sensors::GpuRaySensor::near` [protected]

Near clip plane.

10.51.3.13 `double gazebo::sensors::GpuRaySensor::rangeCountRatio` [protected]

Range count ratio.

10.51.3.14 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::rangeElem` [protected]

Range SDF element.

10.51.3.15 `double gazebo::sensors::GpuRaySensor::rayCountRatio` [protected]

Ray count ratio.

10.51.3.16 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::scanElem` [protected]

Scan SDF elementz.

10.51.3.17 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::vertElem` [protected]

Vertical SDF element.

10.51.3.18 `double gazebo::sensors::GpuRaySensor::vertHalfAngle` [protected]

Vertical half angle.

10.51.3.19 `unsigned int gazebo::sensors::GpuRaySensor::vertRangeCount` [protected]

Vertical range count.

10.51.3.20 `unsigned int gazebo::sensors::GpuRaySensor::vertRayCount` [protected]

Vertical ray count.

10.51.3.21 `double gazebo::sensors::GpuRaySensor::vfov` [protected]

Vertical field-of-view.

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

- **GpuRaySensor.hh**

## 10.52 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. 103) 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.*

### 10.52.1 Detailed Description

Displays a grid of cells, drawn with lines.

Displays a grid of cells, drawn with lines. **A** (p. 107) grid with an identity orientation is drawn along the XY plane.

### 10.52.2 Constructor & Destructor Documentation

10.52.2.1 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

### 10.52.2.2 gazebo::rendering::Grid::~~Grid ( )

Destructor.

## 10.52.3 Member Function Documentation

### 10.52.3.1 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.
-----------	----------------	--

### 10.52.3.2 uint32\_t gazebo::rendering::Grid::GetCellCount ( ) const [inline]

Get the number of cells.

### 10.52.3.3 float gazebo::rendering::Grid::GetCellLength ( ) const [inline]

Get the cell length.

#### Returns

The cell length

### 10.52.3.4 common::Color gazebo::rendering::Grid::GetColor ( ) const [inline]

Return the grid color.

#### Returns

The grid color

### 10.52.3.5 uint32\_t gazebo::rendering::Grid::GetHeight ( ) const [inline]

Get the height of the grid.

#### Returns

The height

### 10.52.3.6 float gazebo::rendering::Grid::GetLineWidth ( ) const [inline]

Get the width of the grid line.

#### Returns

The line width

10.52.3.7 `Ogre::SceneNode* gazebo::rendering::Grid::GetSceneNode ( ) [inline]`

Get the **Ogre** (p. 103) scene node associated with this grid.

Returns

The **Ogre** (p. 103) scene node associated with this grid

10.52.3.8 `void gazebo::rendering::Grid::Init ( )`

Initialize the grid.

10.52.3.9 `void gazebo::rendering::Grid::SetCellCount ( uint32_t _count )`

Set the number of cells.

Parameters

in	_count	The number of cells
----	--------	---------------------

10.52.3.10 `void gazebo::rendering::Grid::SetCellLength ( float _len )`

Set the cell length.

Parameters

in	_len	The cell length
----	------	-----------------

10.52.3.11 `void gazebo::rendering::Grid::SetColor ( const common::Color & _color )`

Sets the color of the grid.

Parameters

in	_color	The grid color
----	--------	----------------

10.52.3.12 `void gazebo::rendering::Grid::SetHeight ( uint32_t _count )`

Set the height of the grid.

Parameters

in	_count	<b>Grid</b> (p. 332) height
----	--------	-----------------------------

10.52.3.13 `void gazebo::rendering::Grid::SetLineWidth ( float _width )`

Set the line width.

## Parameters

<code>in</code>	<code>_width</code>	The width of the grid
-----------------	---------------------	-----------------------

10.52.3.14 `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**

## 10.53 gazebo::physics::Gripper Class Reference

**A** (p. 107) 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.*

#### 10.53.1 Detailed Description

**A** (p. 107) gripper abstraction.

**A** (p. 107) 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.

#### 10.53.2 Constructor & Destructor Documentation

10.53.2.1 `gazebo::physics::Gripper::Gripper ( ModelPtr _model ) [explicit]`

Constructor.

## Parameters

<code>in</code>	<code>_model</code>	The model which contains the <b>Gripper</b> (p. 336).
-----------------	---------------------	---

10.53.2.2 virtual gazebo::physics::Gripper::~~Gripper ( ) [virtual]

Destructor.

### 10.53.3 Member Function Documentation

10.53.3.1 virtual void gazebo::physics::Gripper::Init ( ) [virtual]

Initialize.

10.53.3.2 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**

## 10.54 gazebo::rendering::GUIOverlay Class Reference

**A** (p. 107) 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. 157) to and overlay window.*
- bool **AttachCameraToImage** (**DepthCameraPtr** &\_camera, const std::string &\_windowName)  
*Use this function to draw the output from a **rendering::DepthCamera** (p. 246) 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.*

### 10.54.1 Detailed Description

**A** (p. 107) class that creates a CEGUI overlay on a render window.

### 10.54.2 Constructor & Destructor Documentation

#### 10.54.2.1 gazebo::rendering::GUIOverlay::GUIOverlay ( )

Constructor.

#### 10.54.2.2 virtual gazebo::rendering::GUIOverlay::~GUIOverlay ( ) [virtual]

Destructor.

### 10.54.3 Member Function Documentation

#### 10.54.3.1 bool gazebo::rendering::GUIOverlay::AttachCameraToImage ( CameraPtr &\_camera, const std::string &\_windowName )

Use this function to draw the output from a **rendering::Camera** (p. 157) 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

10.54.3.2 `bool gazebo::rendering::GUIOverlay::AttachCameraToImage ( DepthCameraPtr & _camera, const std::string & _windowName )`

Use this function to draw the output from a **rendering::DepthCamera** (p. 246) 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

10.54.3.3 `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> .

10.54.3.4 `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.

10.54.3.5 `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.

### 10.54.3.6 `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.

### 10.54.3.7 `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.

### 10.54.3.8 `void gazebo::rendering::GUIOverlay::Hide ( )`

Make the overlay invisible.

### 10.54.3.9 `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.
-----------------	----------------------------	--

### 10.54.3.10 `bool gazebo::rendering::GUIOverlay::IsInitialized ( )`

Return true if the overlay has been initialized.

**Returns**

True if initialized



10.54.3.11 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.
-----------	------------------	--------------------------

10.54.3.12 void gazebo::rendering::GUIOverlay::Resize ( unsigned int *\_width*, unsigned int *\_height* )

Resize the window.

10.54.3.13 void gazebo::rendering::GUIOverlay::Show ( )

Make the overlay visible.

10.54.3.14 void gazebo::rendering::GUIOverlay::Update ( )

Update the overlay's objects.

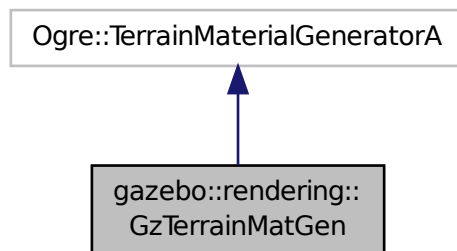
The documentation for this class was generated from the following file:

- **GUIOverlay.hh**

## 10.55 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.*

### 10.55.1 Constructor & Destructor Documentation

#### 10.55.1.1 gazebo::rendering::GzTerrainMatGen::GzTerrainMatGen ( )

Constructor.

#### 10.55.1.2 virtual gazebo::rendering::GzTerrainMatGen::~~GzTerrainMatGen ( ) [virtual]

Destructor.

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

- **Heightmap.hh**

## 10.56 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.*

### 10.56.1 Detailed Description

Rendering a terrain using heightmap information.

## 10.56.2 Constructor & Destructor Documentation

### 10.56.2.1 gazebo::rendering::Heightmap::Heightmap ( ScenePtr *\_scene* )

Constructor.

#### Parameters

in	<i>_scene</i>	Pointer to the scene that will contain the heightmap
----	---------------	--

### 10.56.2.2 virtual gazebo::rendering::Heightmap::~Heightmap ( ) [virtual]

Destructor.

## 10.56.3 Member Function Documentation

### 10.56.3.1 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

### 10.56.3.2 Ogre::TerrainGroup\* gazebo::rendering::Heightmap::GetOgreTerrain ( ) const

Get a pointer to the OGRE terrain group object.

#### Returns

Pointer to the OGRE terrain.

### 10.56.3.3 void gazebo::rendering::Heightmap::Load ( )

Load the heightmap.

### 10.56.3.4 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:

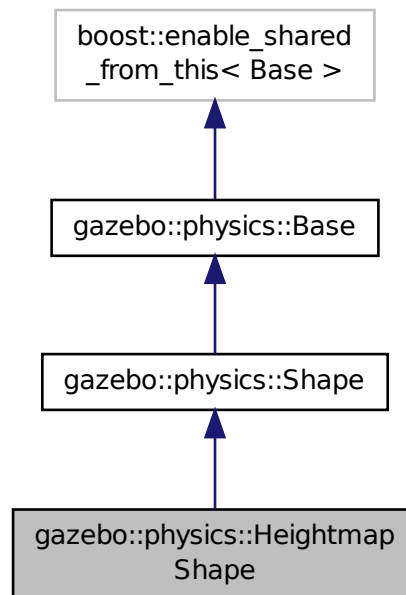
- **Heightmap.hh**

## 10.57 gazebo::physics::HeightmapShape Class Reference

**HeightmapShape** (p. 344) 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

#### 10.57.1 Detailed Description

**HeightmapShape** (p. 344) 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.

#### 10.57.2 Constructor & Destructor Documentation

##### 10.57.2.1 gazebo::physics::HeightmapShape::HeightmapShape ( CollisionPtr \_parent ) [explicit]

Constructor.

##### Parameters

in	<code>_parent</code>	Parent <b>Collision</b> (p. 190) object.
----	----------------------	--

10.57.2.2 `virtual gazebo::physics::HeightmapShape::~~HeightmapShape ( ) [virtual]`

Destructor.

### 10.57.3 Member Function Documentation

10.57.3.1 `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. 695).

10.57.3.2 `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.

10.57.3.3 `float gazebo::physics::HeightmapShape::GetMaxHeight ( ) const`

Get the maximum height.

#### Returns

The maximum height.

10.57.3.4 `float gazebo::physics::HeightmapShape::GetMinHeight ( ) const`

Get the minimum height.

#### Returns

The minimum height.

10.57.3.5 `math::Vector3 gazebo::physics::HeightmapShape::GetPos ( ) const`

Get the origin in world coordinate frame.

**Returns**

The origin in world coordinate frame.

10.57.3.6 `math::Vector3 gazebo::physics::HeightmapShape::GetSize ( ) const`

Get the size in meters.

**Returns**

The size in meters.

10.57.3.7 `int gazebo::physics::HeightmapShape::GetSubSampling ( ) const`

Get the amount of subsampling.

**Returns**

Amount of subsampling.

10.57.3.8 `std::string gazebo::physics::HeightmapShape::GetURI ( ) const`

Get the URI of the heightmap image.

**Returns**

The heightmap image URI.

10.57.3.9 `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. 812), result.x = width, result.y = length/height.

10.57.3.10 `virtual void gazebo::physics::HeightmapShape::Init ( ) [virtual]`

Initialize the heightmap.

Implements **gazebo::physics::Shape** (p. 695).

10.57.3.11 `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. 141).

10.57.3.12 `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. 695).

### 10.57.4 Member Data Documentation

10.57.4.1 `std::vector<float> gazebo::physics::HeightmapShape::heights [protected]`

Lookup table of heights.

10.57.4.2 `common::Image gazebo::physics::HeightmapShape::img [protected]`

Image used to generate the heights.

10.57.4.3 `math::Vector3 gazebo::physics::HeightmapShape::scale [protected]`

Scaling factor.

10.57.4.4 `int gazebo::physics::HeightmapShape::subSampling [protected]`

Level of subsampling.

10.57.4.5 `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**

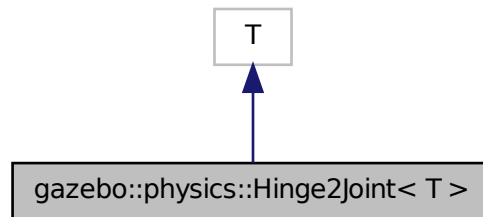
## 10.58 `gazebo::physics::Hinge2Joint< T >` Class Template Reference

**A** (p. 107) 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.*

### 10.58.1 Detailed Description

```
template<class T>class gazebo::physics::Hinge2Joint< T >
```

**A** (p. 107) two axis hinge joint.

### 10.58.2 Constructor & Destructor Documentation

10.58.2.1 `template<class T > gazebo::physics::Hinge2Joint< T >::Hinge2Joint ( BasePtr _parent ) [inline], [explicit]`

Constructor.

#### Parameters

in	_parent	Parent link.
----	---------	--------------

References gazebo::physics::Base::HINGE2\_JOINT.

10.58.2.2 `template<class T> virtual gazebo::physics::Hinge2Joint< T >::~~Hinge2Joint ( ) [inline], [virtual]`

Destructor.

### 10.58.3 Member Function Documentation

10.58.3.1 `template<class T> virtual unsigned int gazebo::physics::Hinge2Joint< T >::GetAngleCount ( ) const [inline], [virtual]`

10.58.3.2 `template<class T> virtual void gazebo::physics::Hinge2Joint< T >::Load ( sdf::ElementPtr _sdf ) [inline], [virtual]`

Load the joint.

#### Parameters

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

References `sdf::Element::GetElement()`, and `sdf::Element::GetValueVector3()`.

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

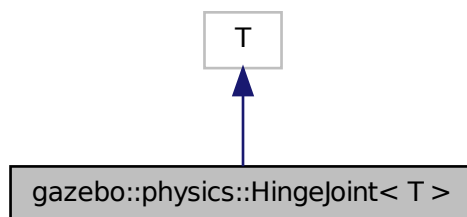
- **Hinge2Joint.hh**

## 10.59 gazebo::physics::HingeJoint< T > Class Template Reference

**A** (p. 107) 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.*

### 10.59.1 Detailed Description

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

**A** (p. 107) single axis hinge joint.

### 10.59.2 Constructor & Destructor Documentation

10.59.2.1 `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.

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

Destructor.

### 10.59.3 Member Function Documentation

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

10.59.3.2 `template<class T > virtual void gazebo::physics::HingeJoint< T >::Init ( )` [inline], [protected],  
[virtual]

Initialize joint.

References gazebo::msgs::Init().

10.59.3.3 `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**

## 10.60 gazebo::common::Image Class Reference

Encapsulates an image.

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

### Public Types

- enum **PixelFormat** {  
**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 }**  
*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. 353).*

### 10.60.1 Detailed Description

Encapsulates an image.

### 10.60.2 Member Enumeration Documentation

#### 10.60.2.1 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***

### 10.60.3 Constructor & Destructor Documentation

10.60.3.1 `gazebo::common::Image::Image ( const std::string & _filename = " " ) [explicit]`

Constructor.

#### Parameters

<code>in</code>	<code><i>_filename</i></code>	the path to the image
-----------------	-------------------------------	-----------------------

10.60.3.2 `virtual gazebo::common::Image::~Image ( ) [virtual]`

Destructor.

### 10.60.4 Member Function Documentation

10.60.4.1 `static Image::PixelFormat gazebo::common::Image::ConvertPixelFormat ( const std::string & _format ) [static]`

Convert a string to a **Image::PixelFormat** (p. 353).

#### Parameters

<code>in</code>	<code><i>_format</i></code>	Pixel format string.
-----------------	-----------------------------	----------------------

#### See Also

`Image::PixelFormatNames`

#### Returns

**Image::PixelFormat** (p. 353)

10.60.4.2 `Color gazebo::common::Image::GetAvgColor ( )`

Get the average color.

**Returns**

The average color

**10.60.4.3 unsigned int gazebo::common::Image::GetBPP ( ) const**

Get the size of one pixel in bits.

**Returns**

The BPP of the image

**10.60.4.4 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

**10.60.4.5 std::string gazebo::common::Image::GetFilename ( ) const**

Get the full filename of the image.

**Returns**

The filename used to load the image

**10.60.4.6 unsigned int gazebo::common::Image::GetHeight ( ) const**

Get the height.

**Returns**

The image height

**10.60.4.7 Color gazebo::common::Image::GetMaxColor ( )**

Get the max color.

**Returns**

The max color

**10.60.4.8 int gazebo::common::Image::GetPitch ( ) const****Returns**

The pitch of the image

#### 10.60.4.9 `Color gazebo::common::Image::GetPixel ( unsigned int _x, unsigned int _y )`

Get a pixel color value.

##### Parameters

in	<code>_x</code>	Column location in the image
in	<code>_y</code>	Row location in the image

#### 10.60.4.10 `PixelFormat gazebo::common::Image::GetPixelFormat ( ) const`

Get the pixel format.

##### Returns

PixelFormat

#### 10.60.4.11 `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	<code>_data</code>	Pointer to a NULL array of char.
out	<code>_count</code>	The resulting data array size

#### 10.60.4.12 `unsigned int gazebo::common::Image::GetWidth ( ) const`

Get the width.

##### Returns

The image width

#### 10.60.4.13 `int gazebo::common::Image::Load ( const std::string & _filename )`

Load an image.

Return 0 on success

##### Parameters

in	<code>_filename</code>	the path to the image file
----	------------------------	----------------------------

#### 10.60.4.14 `void gazebo::common::Image::Rescale ( int _width, int _height )`

Rescale the image.



## Parameters

<code>in</code>	<code>_width</code>	New image width
<code>in</code>	<code>_height</code>	New image height

10.60.4.15 `void gazebo::common::Image::SavePNG ( const std::string & _filename )`

Save the image in PNG format.

## Parameters

<code>in</code>	<code>_filename</code>	The name of the saved image
-----------------	------------------------	-----------------------------

10.60.4.16 `void gazebo::common::Image::SetFromData ( const unsigned char * _data, unsigned int _width, unsigned int _height, Image::PixelFormat _format )`

Set the image from raw data.

## Parameters

<code>in</code>	<code>_data</code>	Pointer to the raw image data
<code>in</code>	<code>_width</code>	Width in pixels
<code>in</code>	<code>_height</code>	Height in pixels
<code>in</code>	<code>_format</code>	Pixel format of the provided data

10.60.4.17 `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:

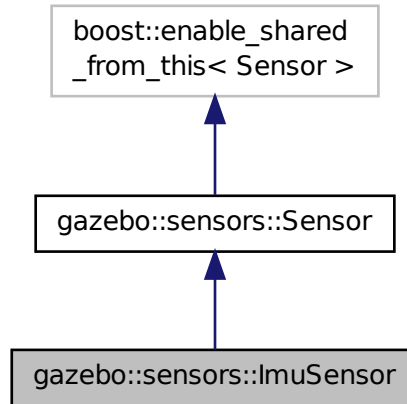
- **Image.hh**

## 10.61 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 linear acceleration.*

## 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

### 10.61.1 Detailed Description

An IMU sensor.

### 10.61.2 Constructor & Destructor Documentation

#### 10.61.2.1 gazebo::sensors::ImuSensor::ImuSensor ( )

Constructor.

#### 10.61.2.2 virtual gazebo::sensors::ImuSensor::~ImuSensor ( ) [virtual]

Destructor.

### 10.61.3 Member Function Documentation

#### 10.61.3.1 virtual void gazebo::sensors::ImuSensor::Fini ( ) [protected],[virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 676).

#### 10.61.3.2 math::Vector3 gazebo::sensors::ImuSensor::GetAngularVelocity ( ) const

Returns the angular velocity.

#### Returns

Angular velocity.

#### 10.61.3.3 math::Vector3 gazebo::sensors::ImuSensor::GetLinearAcceleration ( ) const

Returns the linear acceleration.

#### Returns

Linear acceleration.

#### 10.61.3.4 virtual void gazebo::sensors::ImuSensor::Init ( ) [protected],[virtual]

Initialize the IMU.

Reimplemented from **gazebo::sensors::Sensor** (p. 678).

10.61.3.5 `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 <b>Sensor</b> (p. 672) parameters.
in	<code>_worldName</code>	Name of world to load from.

Reimplemented from `gazebo::sensors::Sensor` (p. 678).

10.61.3.6 `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. 678).

10.61.3.7 `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. 679).

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

- **ImuSensor.hh**

## 10.62 gazebo::physics::Inertial Class Reference

**A** (p. 107) 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. 404) 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.*

### 10.62.1 Detailed Description

**A** (p. 107) class for inertial information about a link.

### 10.62.2 Constructor & Destructor Documentation

#### 10.62.2.1 gazebo::physics::Inertial::Inertial ( )

Default Constructor.

#### 10.62.2.2 gazebo::physics::Inertial::Inertial ( double \_mass ) [explicit]

Constructor.

##### Parameters

<code>in</code>	<code>_mass</code>	Mass value in kg if using metric.
-----------------	--------------------	-----------------------------------

#### 10.62.2.3 gazebo::physics::Inertial::Inertial ( const Inertial & \_inertial )

Copy constructor.

##### Parameters

<code>in</code>	<code>_inertial</code>	<b>Inertial</b> (p. 360) element to copy
-----------------	------------------------	--

10.62.2.4 `virtual gazebo::physics::Inertial::~~Inertial ( ) [virtual]`

Destructor.

### 10.62.3 Member Function Documentation

10.62.3.1 `const math::Vector3& gazebo::physics::Inertial::GetCoG ( ) const [inline]`

Get the center of gravity.

Returns

The center of gravity.

10.62.3.2 `double gazebo::physics::Inertial::GetIXX ( ) const`

Get IXX.

Returns

IXX value

10.62.3.3 `double gazebo::physics::Inertial::GetIXY ( ) const`

Get IXY.

Returns

IXY value

10.62.3.4 `double gazebo::physics::Inertial::GetIXZ ( ) const`

Get IXZ.

Returns

IXZ value

10.62.3.5 `double gazebo::physics::Inertial::GetIYY ( ) const`

Get IYY.

Returns

IYY value

10.62.3.6 `double gazebo::physics::Inertial::GetYZ ( ) const`

Get IYZ.

Returns

IYZ value

10.62.3.7 `double gazebo::physics::Inertial::GetIZZ ( ) const`

Get IZZ.

Returns

IZZ value

10.62.3.8 `double gazebo::physics::Inertial::GetMass ( ) const`

Get the mass.

10.62.3.9 `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. 404) frame.

Returns

The inertial pose.

10.62.3.10 `math::Vector3 gazebo::physics::Inertial::GetPrincipalMoments ( ) const`

Get the principal moments of inertia (Ixx, Iyy, Izz).

Returns

The principal moments.

10.62.3.11 `math::Vector3 gazebo::physics::Inertial::GetProductsofInertia ( ) const`

Get the products of inertia (Ixy, Ixy, Iyz).

Returns

The products of inertia.

10.62.3.12 `void gazebo::physics::Inertial::Load ( sdf::ElementPtr _sdf )`

Load from SDF values.

Parameters



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

### 10.62.3.13 Inertial gazebo::physics::Inertial::operator+ ( const Inertial & *\_inertial* ) const

Addition operator.

#### Parameters

<code>in</code>	<code>_inertial</code>	<b>Inertial</b> (p. 360) to add.
-----------------	------------------------	----------------------------------

#### Returns

The result of the addition.

### 10.62.3.14 const Inertial& gazebo::physics::Inertial::operator+= ( const Inertial & *\_inertial* )

Addition equal operator.

#### Parameters

<code>in</code>	<code>_inertial</code>	<b>Inertial</b> (p. 360) to add.
-----------------	------------------------	----------------------------------

#### Returns

Reference to this object.

### 10.62.3.15 Inertial& gazebo::physics::Inertial::operator= ( const Inertial & *\_inertial* )

Equal operator.

#### Parameters

<code>in</code>	<code>_inertial</code>	<b>Inertial</b> (p. 360) to copy.
-----------------	------------------------	-----------------------------------

#### Returns

Reference to this object.

### 10.62.3.16 void gazebo::physics::Inertial::ProcessMsg ( const msgs::Inertial & *\_msg* )

Update parameters from a message.

#### Parameters

<code>in</code>	<code>_msg</code>	Message to read
-----------------	-------------------	-----------------

10.62.3.17 void gazebo::physics::Inertial::Reset ( )

Reset all the mass properties.

10.62.3.18 void gazebo::physics::Inertial::Rotate ( const math::Quaternion & \_rot )

Rotate this mass.

Parameters

in	_rot	Rotation amount.
----	------	------------------

10.62.3.19 void gazebo::physics::Inertial::SetCoG ( double \_cx, double \_cy, double \_cz )

Set the center of gravity.

Parameters

in	_cx	X position.
in	_cy	Y position.
in	_cz	Z position.

10.62.3.20 void gazebo::physics::Inertial::SetCoG ( const math::Vector3 & \_center )

Set the center of gravity.

Parameters

in	_center	Center of the gravity.
----	---------	------------------------

10.62.3.21 void gazebo::physics::Inertial::SetInertiaMatrix ( double \_ixx, double \_iyy, double \_izz, double \_ixy, double \_ixz, double iyz )

Set the mass matrix.

Parameters

in	_ixx	X second moment of inertia about x axis.
in	_iyy	Y second moment of inertia about y axis.
in	_izz	Z second moment of inertia about z axis.
in	_ixy	XY inertia.
in	_ixz	XZ inertia.
in	_iyz	YZ inertia.

10.62.3.22 void gazebo::physics::Inertial::SetIXX ( double \_v )

Set IXX.

## Parameters

in	_v	IXX value
----	----	-----------

10.62.3.23 void gazebo::physics::Inertial::SetIXY ( double \_v )

Set IXY.

## Parameters

in	_v	IXY value
----	----	-----------

10.62.3.24 void gazebo::physics::Inertial::SetIXZ ( double \_v )

Set IXZ.

## Parameters

in	_v	IXZ value
----	----	-----------

10.62.3.25 void gazebo::physics::Inertial::SetIYY ( double \_v )

Set IYY.

## Parameters

in	_v	IYY value
----	----	-----------

10.62.3.26 void gazebo::physics::Inertial::SetIYZ ( double \_v )

Set IYZ.

## Parameters

in	_v	IXX value
----	----	-----------

10.62.3.27 void gazebo::physics::Inertial::SetIZZ ( double \_v )

Set IZZ.

## Parameters

in	_v	IZZ value
----	----	-----------

10.62.3.28 void gazebo::physics::Inertial::SetMass ( double m )

Set the mass.

10.62.3.29 void gazebo::physics::Inertial::UpdateParameters ( sdf::ElementPtr \_sdf )

update the parameters using new sdf values.

#### Parameters

in	<i>_sdf</i>	Update values from.
----	-------------	---------------------

## 10.62.4 Friends And Related Function Documentation

10.62.4.1 std::ostream& operator<< ( std::ostream & \_out, const gazebo::physics::Inertial & \_inertial ) [friend]

Output operator.

#### Parameters

in	<i>_out</i>	Output stream.
in	<i>_inertial</i>	<b>Inertial</b> (p. 360) object to output.

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

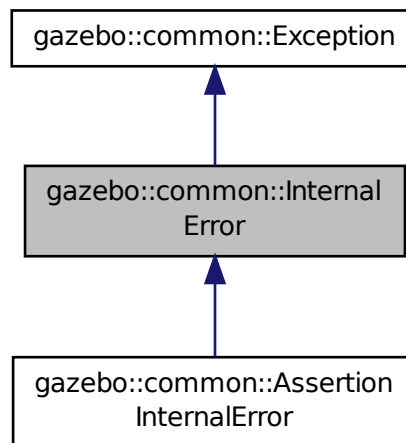
- **Inertial.hh**

## 10.63 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 **~InternalError** ()  
*Destructor.*

### 10.63.1 Detailed Description

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

### 10.63.2 Constructor & Destructor Documentation

#### 10.63.2.1 gazebo::common::InternalError::InternalError ( )

Constructor.

#### 10.63.2.2 gazebo::common::InternalError::InternalError ( const char \* \_file, int \_line, const 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

#### 10.63.2.3 virtual gazebo::common::InternalError::~~InternalError ( ) [virtual]

Destructor.

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

- **Exception.hh**

## 10.64 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.*

### 10.64.1 Detailed Description

Manages boost::asio IO.

### 10.64.2 Constructor & Destructor Documentation

#### 10.64.2.1 gazebo::transport::IOManager::IOManager ( )

Constructor.

#### 10.64.2.2 gazebo::transport::IOManager::~~IOManager ( )

Destructor.

### 10.64.3 Member Function Documentation

#### 10.64.3.1 void gazebo::transport::IOManager::DecCount ( )

Decrement the event count by 1.

#### 10.64.3.2 unsigned int gazebo::transport::IOManager::GetCount ( ) const

Get the event count.

#### Returns

The event count

#### 10.64.3.3 boost::asio::io\_service& gazebo::transport::IOManager::GetIO ( )

Get handle to boost::asio IO service.

#### Returns

Handle to boost::asio IO service

## 10.64.3.4 void gazebo::transport::IOManager::IncCount ( )

Increment the event count by 1.

## 10.64.3.5 void gazebo::transport::IOManager::Stop ( )

Stop the IO service.

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

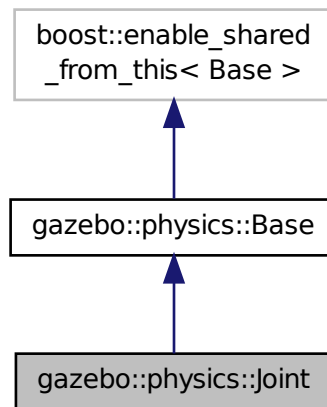
- **IOManager.hh**

## 10.65 gazebo::physics::Joint Class Reference

**Base** (p. 133) 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. 371) attribute types.

## Public Member Functions

- **Joint** (**BasePtr** \_parent)

- Constructor.*

  - virtual  $\sim$ **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.*
- **LinkPtr GetChild** () const

*Get the child link.*
- virtual double **GetForce** (int \_index)
- virtual **JointWrench GetForceTorque** (int \_index)=0

*get force torque values at a joint*
- 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).*
- virtual **LinkPtr GetJointLink** (int \_index) const =0

*Get the link to which the joint is attached according to 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. 404) due to the existence of this **Joint** (p. 371).*
- virtual **math::Vector3 GetLinkTorque** (unsigned int \_index) const =0

*Get the torque applied to the center of mass of a **physics::Link** (p. 404) due to the existence of this **Joint** (p. 371).*
- **math::Vector3 GetLocalAxis** (int \_index) const

*Get the axis of rotation.*
- 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.*



- virtual double **GetVelocity** (int \_index) const =0  
*Get the rotation rate of an 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. 371).*
- void **Load** (LinkPtr \_parent, LinkPtr \_child, const math::Vector3 &\_pos) **GAZEBO\_DEPRECATED**  
*Set parent and child links of a **physics::Joint** (p. 371) and its anchor offset position.*
- virtual void **Load** (sdf::ElementPtr \_sdf)  
*Load **physics::Joint** (p. 371) from a SDF **sdf::Element** (p. 266).*
- 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. 371) is static, Gazebo stores the state of this **Joint** (p. 371) as a scalar inside the **Joint** (p. 371) class, so this call will NOT move the joint dynamically for a static **Model** (p. 469).*
- 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. 371).*
- virtual void **SetHighStop** (int \_index, const math::Angle &\_angle)=0  
*Set the high stop of an axis(index).*
- virtual void **SetLowStop** (int \_index, const math::Angle &\_angle)=0  
*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.*
- **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 **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.*
- **ModelPtr model**  
*Pointer to the parent model.*
- **LinkPtr parentLink**  
*The second link this joint connects to.*

### 10.65.1 Detailed Description

**Base** (p. 133) class for all joints.

### 10.65.2 Member Enumeration Documentation

#### 10.65.2.1 enum gazebo::physics::Joint::Attribute

**Joint** (p. 371) 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.

### 10.65.3 Constructor & Destructor Documentation

10.65.3.1 gazebo::physics::Joint::Joint ( BasePtr *\_parent* ) [explicit]

Constructor.

#### Parameters

in	<b>Joint</b> (p. 371)	parent
----	-----------------------	--------

10.65.3.2 virtual gazebo::physics::Joint::~~Joint ( ) [virtual]

Destructor.

### 10.65.4 Member Function Documentation

10.65.4.1 virtual void gazebo::physics::Joint::ApplyDamping ( ) [virtual]

Callback to apply damping force to joint.

10.65.4.2 virtual bool gazebo::physics::Joint::AreConnected ( LinkPtr *\_one*, LinkPtr *\_two* ) const [pure virtual]

Determines of the two bodies are connected by a joint.

#### Parameters

in	<i>_one</i>	First link.
in	<i>_two</i>	Second link.

#### Returns

True if the two links are connected by a joint.

10.65.4.3 virtual void gazebo::physics::Joint::Attach ( LinkPtr *\_parent*, LinkPtr *\_child* ) [virtual]

Attach the two bodies with this joint.

#### Parameters

in	<i>_parent</i>	Parent link.
in	<i>_child</i>	Child link.

10.65.4.4 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().

10.65.4.5 `virtual void gazebo::physics::Joint::Detach ( ) [virtual]`

Detach this joint from all links.

10.65.4.6 `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().

10.65.4.7 `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.
-----	-------------------	---

10.65.4.8 `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.

10.65.4.9 `math::Angle gazebo::physics::Joint::GetAngle ( int _index ) const`

Get the angle of rotation of an axis(index)

## Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

## Returns

Angle of the axis.

10.65.4.10 `virtual unsigned int gazebo::physics::Joint::GetAngleCount ( ) const` [pure virtual]

Get the angle count.

## Returns

The number of DOF for the joint.

10.65.4.11 `virtual math::Angle gazebo::physics::Joint::GetAngleImpl ( int _index ) const` [protected], [pure virtual]

Get the angle of an axis helper function.

## Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

## Returns

Angle of the axis.

10.65.4.12 `LinkPtr gazebo::physics::Joint::GetChild ( ) const`

Get the child link.

## Returns

Pointer to the child link.

10.65.4.13 `virtual double gazebo::physics::Joint::GetForce ( int _index )` [virtual]

**Todo** : not yet implemented. Get the internal forces at a this **Joint** (p.371). 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

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

**Returns**

The force applied to an axis.

10.65.4.14 `virtual JointWrench gazebo::physics::Joint::GetForceTorque ( int _index ) [pure virtual]`

get force torque values at a joint

**Parameters**

<code>in</code>	<code><i>_index</i></code>	Force and torque on child link if <code>_index = 0</code> and on parent link of <code>_index = 1</code>
-----------------	----------------------------	---

**Returns**

The force and torque at the joint

10.65.4.15 `virtual math::Vector3 gazebo::physics::Joint::GetGlobalAxis ( int _index ) const [pure virtual]`

Get the axis of rotation in global coordinate frame.

**Parameters**

<code>in</code>	<code><i>_index</i></code>	Index of the axis to get.
-----------------	----------------------------	---------------------------

**Returns**

Axis value for the provided index.

10.65.4.16 `virtual math::Angle gazebo::physics::Joint::GetHighStop ( int _index ) [pure virtual]`

Get the high stop of an axis(index).

**Parameters**

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

**Returns**

Angle of the high stop value.

10.65.4.17 `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.

10.65.4.18 `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. 404) due to the existence of this **Joint** (p. 371).

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.

10.65.4.19 `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. 404) due to the existence of this **Joint** (p. 371).

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><i>index</i></code>	The index of the link(0 or 1)
-----------------	---------------------------	-------------------------------

**Returns**

Torque applied to the link.

10.65.4.20 `math::Vector3 gazebo::physics::Joint::GetLocalAxis ( int _index ) const`

Get the axis of rotation.

**Parameters**

<code>in</code>	<code><i>_index</i></code>	Index of the axis to get.
-----------------	----------------------------	---------------------------

**Returns**

Axis value for the provided index.

10.65.4.21 `virtual math::Angle gazebo::physics::Joint::GetLowStop ( int _index )` [pure virtual]

Get the low stop of an axis(index).

## Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

## Returns

Angle of the low stop value.

**10.65.4.22** `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.

**10.65.4.23** `LinkPtr gazebo::physics::Joint::GetParent ( ) const`

Get the parent link.

## Returns

Pointer to the parent link.

**10.65.4.24** `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.

**10.65.4.25** `virtual void gazebo::physics::Joint::Init ( ) [virtual]`

Initialize a joint.

Reimplemented from `gazebo::physics::Base` (p. 141).



10.65.4.26 void gazebo::physics::Joint::Load ( LinkPtr *\_parent*, LinkPtr *\_child*, const math::Pose & *\_pose* )

Set pose, parent and child links of a **physics::Joint** (p. 371).

#### Parameters

in	<i>_parent</i>	Parent link.
in	<i>_child</i>	Child link.
in	<i>_pose</i>	Pose containing <b>Joint</b> (p. 371) Anchor offset from child link.

10.65.4.27 void gazebo::physics::Joint::Load ( LinkPtr *\_parent*, LinkPtr *\_child*, const math::Vector3 & *\_pos* )

Set parent and child links of a **physics::Joint** (p. 371) and its anchor offset position.

This function is deprecated, use **Load(LinkPtr *\_parent*, LinkPtr *\_child*, const math::Pose & *\_pose*)** (p. 381)

#### Parameters

in	<i>_parent</i>	Parent link.
in	<i>_child</i>	Child link.
in	<i>_pos</i>	<b>Joint</b> (p. 371) Anchor offset from child link.

10.65.4.28 virtual void gazebo::physics::Joint::Load ( sdf::ElementPtr *sdf* ) [virtual]

Load **physics::Joint** (p. 371) from a SDF **sdf::Element** (p. 266).

#### Parameters

in	<i>_sdf</i>	SDF values to load from.
----	-------------	--------------------------

Reimplemented from **gazebo::physics::Base** (p. 141).

10.65.4.29 virtual void gazebo::physics::Joint::Reset ( ) [virtual]

Reset the joint.

Reimplemented from **gazebo::physics::Base** (p. 142).

10.65.4.30 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.

10.65.4.31 `void gazebo::physics::Joint::SetAngle ( int _index, math::Angle _angle )`

If the **Joint** (p. 371) is static, Gazebo stores the state of this **Joint** (p. 371) as a scalar inside the **Joint** (p. 371) class, so this call will NOT move the joint dynamically for a static **Model** (p. 469).

But if this **Model** (p. 469) is not static, then it is updated dynamically, all the connected children **Link** (p. 404)'s are moved as a result of the **Joint** (p. 371) angle setting. Dynamic **Joint** (p. 371) angle update is accomplished by calling **JointController::SetJointPosition** (p. 386).

#### Parameters

in	<code><i>_index</i></code>	Index of the axis.
in	<code><i>_angle</i></code>	Angle to set the joint to.

10.65.4.32 `virtual void gazebo::physics::Joint::SetAttribute ( const std::string & _key, int _index, const boost::any & _value )`  
`[pure virtual]`

Set a non-generic parameter for the joint.

replaces SetAttribute(Attribute, int, double)

#### Parameters

in	<code><i>_key</i></code>	String key.
in	<code><i>_index</i></code>	Index of the axis.
in	<code><i>_value</i></code>	Value of the attribute.

10.65.4.33 `virtual void gazebo::physics::Joint::SetAxis ( int _index, const math::Vector3 & _axis )` `[pure virtual]`

Set the axis of rotation.

#### Parameters

in	<code><i>_index</i></code>	Index of the axis to set.
in	<code><i>_axis</i></code>	Axis value.

10.65.4.34 `virtual void gazebo::physics::Joint::SetDamping ( int _index, double _damping )` `[pure virtual]`

Set the joint damping.

#### Parameters

in	<code><i>_index</i></code>	Index of the axis to set.
in	<code><i>_damping</i></code>	Damping value for the axis.

10.65.4.35 `virtual void gazebo::physics::Joint::SetForce ( int _index, double _force )` `[virtual]`

Set the force applied to this **physics::Joint** (p. 371).

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

in	<i>_index</i>	Index of the axis.
in	<i>_force</i>	Force value.

10.65.4.36 virtual void gazebo::physics::Joint::SetHighStop ( int *\_index*, const math::Angle & *\_angle* ) [pure virtual]

Set the high stop of an axis(index).

## Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_angle</i>	High stop angle.

10.65.4.37 virtual void gazebo::physics::Joint::SetLowStop ( int *\_index*, const math::Angle & *\_angle* ) [pure virtual]

Set the low stop of an axis(index).

## Parameters

in	<i>_index</i>	Index of the axis.
in	<i>_angle</i>	Low stop angle.

10.65.4.38 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

in	<i>_index</i>	Index of the axis.
in	<i>_force</i>	Maximum force that can be applied to the axis.

10.65.4.39 void gazebo::physics::Joint::SetModel ( ModelPtr *\_model* )

Set the model this joint belongs too.

## Parameters

in	<i>_model</i>	Pointer to a model.
----	---------------	---------------------

10.65.4.40 void gazebo::physics::Joint::SetState ( const JointState & *\_state* )

Set the joint state.

## Parameters

in	<code>_state</code>	<b>Joint</b> (p. 371) state
----	---------------------	-----------------------------

10.65.4.41 `virtual void gazebo::physics::Joint::SetVelocity ( int _index, double _vel )` [pure virtual]

Set the velocity of an axis(index).

## Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_vel</code>	Velocity.

10.65.4.42 `void gazebo::physics::Joint::Update ( )` [virtual]

Update the joint.

Reimplemented from **gazebo::physics::Base** (p. 143).

10.65.4.43 `virtual void gazebo::physics::Joint::UpdateParameters ( sdf::ElementPtr _sdf )` [virtual]

Update the parameters using new sdf values.

## Parameters

in	<code>_sdf</code>	SDF values to update from.
----	-------------------	----------------------------

Reimplemented from **gazebo::physics::Base** (p. 144).

## 10.65.5 Member Data Documentation

10.65.5.1 `LinkPtr gazebo::physics::Joint::anchorLink` [protected]

Anchor link.

10.65.5.2 `math::Vector3 gazebo::physics::Joint::anchorPos` [protected]

Anchor pose.

10.65.5.3 `gazebo::event::ConnectionPtr gazebo::physics::Joint::applyDamping` [protected]

apply damping for adding viscous damping forces on updates

10.65.5.4 `LinkPtr gazebo::physics::Joint::childLink` [protected]

The first link this joint connects to.

10.65.5.5 double gazebo::physics::Joint::dampingCoefficient [protected]

joint dampingCoefficient

10.65.5.6 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.

10.65.5.7 ModelPtr gazebo::physics::Joint::model [protected]

Pointer to the parent model.

10.65.5.8 LinkPtr gazebo::physics::Joint::parentLink [protected]

The second link this joint connects to.

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

- **Joint.hh**

## 10.66 gazebo::physics::JointController Class Reference

**A** (p. 107) class for manipulating **physics::Joint** (p. 371).

```
#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. 371) by name.*
- void **SetJointPosition** (**JointPtr** \_joint, double \_position)  
*Set the positions of a **Joint** (p. 371) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 718) and radians for **HingeJoint** (p. 350), etc.*
- void **SetJointPositions** (const std::map< std::string, double > &\_jointPositions)  
*Set the positions of a set of **Joint** (p. 371)'s.*
- void **Update** ()  
*Update the joint control.*

### 10.66.1 Detailed Description

**A** (p. 107) class for manipulating **physics::Joint** (p. 371).

## 10.66.2 Constructor & Destructor Documentation

### 10.66.2.1 gazebo::physics::JointController::JointController ( **ModelPtr** *\_model* ) [explicit]

Constructor.

#### Parameters

in	<i>_model</i>	<b>Model</b> (p. 469) that uses this joint controller.
----	---------------	--

## 10.66.3 Member Function Documentation

### 10.66.3.1 void gazebo::physics::JointController::AddJoint ( **JointPtr** *\_joint* )

Add a joint to control.

#### Parameters

in	<i>_joint</i>	<b>Joint</b> (p. 371) to control.
----	---------------	-----------------------------------

### 10.66.3.2 void gazebo::physics::JointController::Reset ( )

Reset all commands.

### 10.66.3.3 void gazebo::physics::JointController::SetJointPosition ( const std::string & *\_name*, double *\_position* )

Set the positions of a **Joint** (p. 371) by name.

#### See Also

**JointController::SetJointPosition(JointPtr, double)** (p. 386)

### 10.66.3.4 void gazebo::physics::JointController::SetJointPosition ( **JointPtr** *\_joint*, double *\_position* )

Set the positions of a **Joint** (p. 371) by name. The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 718) and radians for **HingeJoint** (p. 350), etc.

Implementation: In order to change the position of a **Joint** (p. 371) inside a **Model** (p. 469), this call must recursively crawl through all the connected children **Link** (p. 404)'s in this **Model** (p. 469), and update each **Link** (p. 404) Pose affected by this **Joint** (p. 371) 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>	<b>Joint</b> (p. 371) to set.
in	<i>_position</i>	Position of the joint.

### 10.66.3.5 void gazebo::physics::JointController::SetJointPositions ( const std::map< std::string, double > & *\_jointPositions* )

Set the positions of a set of **Joint** (p. 371)'s.

## See Also

**JointController::SetJointPosition(JointPtr, double)** (p. 386)

10.66.3.6 void gazebo::physics::JointController::Update ( )

Update the joint control.

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

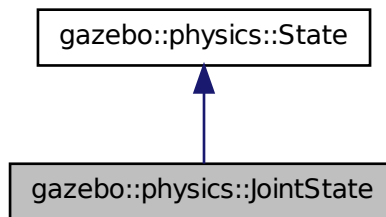
- **JointController.hh**

## 10.67 gazebo::physics::JointState Class Reference

keeps track of state of a **physics::Joint** (p. 371)

```
#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

### 10.67.1 Detailed Description

keeps track of state of a **physics::Joint** (p. 371)

### 10.67.2 Constructor & Destructor Documentation

#### 10.67.2.1 gazebo::physics::JointState::JointState ( )

Default constructor.

#### 10.67.2.2 gazebo::physics::JointState::JointState ( **JointPtr** *joint* ) [explicit]

Constructor.

#### Parameters

<code>in</code>	<code>_joint</code>	<b>Joint</b> (p. 371) to get the state of.
-----------------	---------------------	--

#### 10.67.2.3 gazebo::physics::JointState::JointState ( const **sdf::ElementPtr** *sdf* ) [explicit]

Constructor.

Build a **JointState** (p. 387) from SDF data



## Parameters

in	_sdf	SDF data to load a joint state from.
----	------	--------------------------------------

10.67.2.4 virtual gazebo::physics::JointState::~~JointState ( ) [virtual]

Destructor.

### 10.67.3 Member Function Documentation

10.67.3.1 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.
-----	------	--------------------------

10.67.3.2 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

<b>common::Exception</b> (p. 310)	When _axis is invalid.
--------------------------------------	------------------------

10.67.3.3 unsigned int gazebo::physics::JointState::GetAngleCount ( ) const

Get the number of angles.

## Returns

The number of angles.

10.67.3.4 const std::vector<math::Angle>& gazebo::physics::JointState::GetAngles ( ) const

Get the angles.

**Returns**

Vector of angles.

### 10.67.3.5 `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.

### 10.67.3.6 `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. 732).

### 10.67.3.7 `JointState gazebo::physics::JointState::operator+ ( const JointState & _state ) const`

Addition operator.

**Parameters**

<code>in</code>	<code>_pt</code>	<b>A</b> (p. 107) state to add.
-----------------	------------------	---------------------------------

**Returns**

The resulting state.

### 10.67.3.8 `JointState gazebo::physics::JointState::operator- ( const JointState & _state ) const`

Subtraction operator.

**Parameters**

<code>in</code>	<code>_pt</code>	<b>A</b> (p. 107) state to subtract.
-----------------	------------------	--------------------------------------

**Returns**

The resulting state.

### 10.67.3.9 `JointState& gazebo::physics::JointState::operator= ( const JointState & _state )`

Assignment operator.

## Parameters

in	<code>_state</code>	<b>State</b> (p. 729) value
----	---------------------	-----------------------------

## Returns

this

## 10.67.4 Friends And Related Function Documentation

10.67.4.1 `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>	<b>Joint</b> (p. 371) state to output.

## Returns

The stream.

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

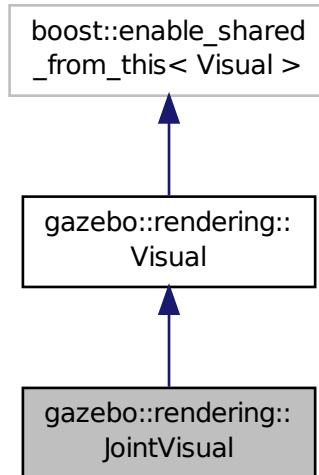
- [JointState.hh](#)

## 10.68 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

### 10.68.1 Detailed Description

Visualization for joints.

### 10.68.2 Constructor & Destructor Documentation

#### 10.68.2.1 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

10.68.2.2 virtual gazebo::rendering::JointVisual::~~JointVisual( ) [virtual]

Destructor.

### 10.68.3 Member Function Documentation

10.68.3.1 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**

## 10.69 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.*

### 10.69.1 Detailed Description

Wrench information from a joint.

These are forces and torques on parent and child Links, relative to the **Link** (p. 404)'s center of mass.

## 10.69.2 Member Function Documentation

### 10.69.2.1 `JointWrench& gazebo::physics::JointWrench::operator= ( const JointWrench & _wrench ) [inline]`

Operator =.

#### Parameters

<code>in</code>	<code>_wrench</code>	<b>Joint</b> (p. 371) wrench to set from.
-----------------	----------------------	---

#### Returns

`*this`

References `body1Force`, `body1Torque`, `body2Force`, and `body2Torque`.

## 10.69.3 Member Data Documentation

### 10.69.3.1 `math::Vector3 gazebo::physics::JointWrench::body1Force`

Force on the first link.

Referenced by `operator=()`.

### 10.69.3.2 `math::Vector3 gazebo::physics::JointWrench::body1Torque`

Torque on the first link.

Referenced by `operator=()`.

### 10.69.3.3 `math::Vector3 gazebo::physics::JointWrench::body2Force`

Force on the second link.

Referenced by `operator=()`.

### 10.69.3.4 `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:

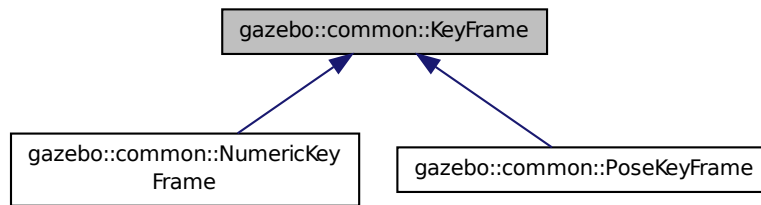
- **JointWrench.hh**

## 10.70 `gazebo::common::KeyFrame` Class Reference

**A** (p. 107) 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*

#### 10.70.1 Detailed Description

**A** (p. 107) key frame in an animation.

#### 10.70.2 Constructor & Destructor Documentation

##### 10.70.2.1 gazebo::common::KeyFrame::KeyFrame ( double *\_time* )

Constructor.

##### Parameters

in	<i>_time</i>	<b>Time</b> (p. 760) of the keyframe in seconds
----	--------------	---

##### 10.70.2.2 virtual gazebo::common::KeyFrame::~~KeyFrame ( ) [virtual]

Destructor.

### 10.70.3 Member Function Documentation

#### 10.70.3.1 `double gazebo::common::KeyFrame::GetTime ( ) const`

Get the time of the keyframe.

#### Returns

the time

### 10.70.4 Member Data Documentation

#### 10.70.4.1 `double gazebo::common::KeyFrame::time [protected]`

time of key frame

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

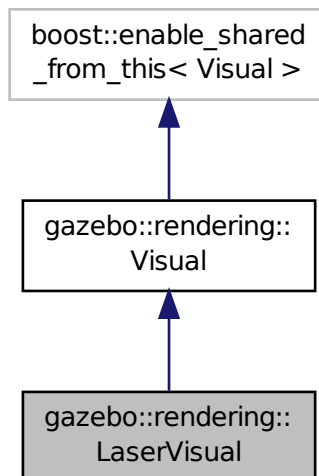
- **KeyFrame.hh**

## 10.71 `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

### 10.71.1 Detailed Description

Visualization for laser data.

### 10.71.2 Constructor & Destructor Documentation

10.71.2.1 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 <b>Visual</b> (p. 850).
in	<code>_topicName</code>	Name of the topic that has laser data.

10.71.2.2 virtual gazebo::rendering::LaserVisual::~~LaserVisual ( ) [virtual]

Destructor.

### 10.71.3 Member Function Documentation

10.71.3.1 virtual void gazebo::rendering::LaserVisual::SetEmissive ( const **common::Color** & \_color ) [virtual]

Documentation inherited from parent.

Reimplemented from **gazebo::rendering::Visual** (p. 865).

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

- **LaserVisual.hh**

## 10.72 gazebo::rendering::Light Class Reference

**A** (p. 107) 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.*

### 10.72.1 Detailed Description

**A** (p. 107) 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.

### 10.72.2 Constructor & Destructor Documentation

#### 10.72.2.1 gazebo::rendering::Light::Light ( ScenePtr \_scene )

Constructor.

##### Parameters

in	_scene	Pointer to the scene that contains the <b>Light</b> (p. 397).
----	--------	---

#### 10.72.2.2 virtual gazebo::rendering::Light::~~Light ( ) [virtual]

Destructor.

### 10.72.3 Member Function Documentation

#### 10.72.3.1 void gazebo::rendering::Light::FillMsg ( msgs::Light & \_msg ) const

Fill the contents of a light message.

##### Parameters

out	_msg	Message to fill.
-----	------	------------------

**10.72.3.2 common::Color gazebo::rendering::Light::GetDiffuseColor ( ) const**

Get the diffuse color.

**Returns**

The light's diffuse color.

**10.72.3.3 math::Vector3 gazebo::rendering::Light::GetDirection ( ) const**

Get the direction.

**Returns**

The light's direction.

**10.72.3.4 std::string gazebo::rendering::Light::GetName ( ) const**

Get the name of the visual.

**Returns**

The light's name.

**10.72.3.5 math::Vector3 gazebo::rendering::Light::GetPosition ( ) const**

Get the position of the light.

**Returns**

The position of the light

**10.72.3.6 common::Color gazebo::rendering::Light::GetSpecularColor ( ) const**

Get the specular color.

**Returns**

The specular color

**10.72.3.7 std::string gazebo::rendering::Light::GetType ( ) const**

Get the type of the light.

**Returns**

The light type: "point", "spot", "directional".

10.72.3.8 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 <b>Light</b> (p. 397) description.
----	-------------	--

## 10.72.3.9 void gazebo::rendering::Light::Load ( )

Load the light using default parameters.

10.72.3.10 void gazebo::rendering::Light::LoadFromMsg ( ConstLightPtr & *\_msg* )

Load from a light message.

## Parameters

in	<i>_msg</i>	Containing the light information.
----	-------------	-----------------------------------

## 10.72.3.11 virtual void gazebo::rendering::Light::OnPoseChange ( ) [inline],[protected],[virtual]

On pose change callback.

10.72.3.12 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

10.72.3.13 void gazebo::rendering::Light::SetCastShadows ( const bool & *\_cast* )

Set cast shadows.

## Parameters

in	<i>_cast</i>	Set to true to cast shadows.
----	--------------	------------------------------

10.72.3.14 void gazebo::rendering::Light::SetDiffuseColor ( const common::Color & *\_color* )

Set the diffuse color.

## Parameters

in	<code>_color</code>	<b>Light</b> (p. 397) diffuse color.
----	---------------------	--------------------------------------

10.72.3.15 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.
----	-------------------	--

10.72.3.16 void gazebo::rendering::Light::SetLightType ( const std::string & *\_type* )

Set the light type.

## Parameters

in	<code>_type</code>	The light type: "point", "spot", "directional"
----	--------------------	--

10.72.3.17 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.
----	--------------------	---------------------------

10.72.3.18 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
----	-----------------	----------------------------

10.72.3.19 void gazebo::rendering::Light::SetRange ( const double & *\_range* )

Set the range.

## Parameters

in	<code>_range</code>	Range of the light in meters.
----	---------------------	-------------------------------

10.72.3.20 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.
----	----	---

10.72.3.21 void gazebo::rendering::Light::SetSpecularColor ( const common::Color & *\_color* )

Set the specular color.

## Parameters

in	_color	The specular color
----	--------	--------------------

10.72.3.22 void gazebo::rendering::Light::SetSpotFalloff ( const double & *\_value* )

Set the spot light falloff.

## Parameters

in	_value	Falloff value
----	--------	---------------

10.72.3.23 void gazebo::rendering::Light::SetSpotInnerAngle ( const double & *\_angle* )

Set the spot light inner angle.

## Parameters

in	_angle	Inner angle in radians
----	--------	------------------------

10.72.3.24 void gazebo::rendering::Light::SetSpotOuterAngle ( const double & *\_angle* )

Set the spot light outer angle.

## Parameters

in	_angle	Outer angle in radians
----	--------	------------------------

10.72.3.25 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.
----	----	--

10.72.3.26 void gazebo::rendering::Light::ToggleShowVisual ( )

10.72.3.27 void gazebo::rendering::Light::UpdateFromMsg ( ConstLightPtr & \_msg )

Update a light source from a message.

#### Parameters

in	_msg	<b>Light</b> (p. 397) message to update from
----	------	--

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

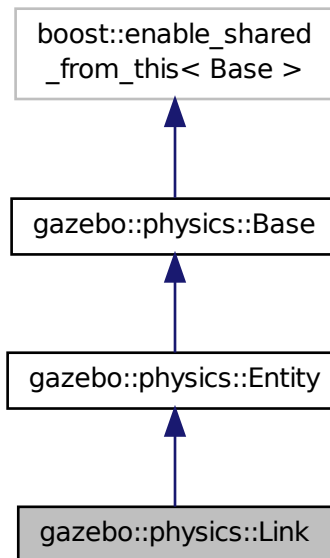
- **Light.hh**

## 10.73 gazebo::physics::Link Class Reference

**Link** (p. 404) 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. 404) as a parent **Link** (p. 404).*
- 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. 404) as a child **Link** (p. 404).*
- 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
  - Get a collision by id.*
- **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** ()=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 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 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. 404) as a parent **Link** (p. 404).*
- void **RemoveParentJoint** (**JointPtr** \_joint)  
*Remove Joints that have this **Link** (p. 404) as a child **Link** (p. 404).*
- void **Reset** ()  
*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. 360) properties.*
- **math::Vector3 linearAccel**  
*Linear acceleration.*
- **std::vector< std::string > visuals**  
*Link (p. 404) visual elements.*

## Additional Inherited Members

### 10.73.1 Detailed Description

**Link** (p. 404) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

### 10.73.2 Constructor & Destructor Documentation

#### 10.73.2.1 gazebo::physics::Link::Link ( EntityPtr \_parent ) [explicit]

Constructor.

##### Parameters

in	_parent	Parent of this link.
----	---------	----------------------

#### 10.73.2.2 virtual gazebo::physics::Link::~~Link ( ) [virtual]

Destructor.

### 10.73.3 Member Function Documentation

#### 10.73.3.1 void gazebo::physics::Link::AddChildJoint ( JointPtr \_joint )

Joints that have this **Link** (p. 404) as a parent **Link** (p. 404).

##### Parameters

in	_joint	<b>Joint</b> (p. 371) that is a child of this link.
----	--------	---

10.73.3.2 `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.
----	---------------	---------------

10.73.3.3 `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.

10.73.3.4 `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.

10.73.3.5 `void gazebo::physics::Link::AddParentJoint ( JointPtr _joint )`

Joints that have this **Link** (p. 404) as a child **Link** (p. 404).

#### Parameters

in	<i>_joint</i>	<b>Joint</b> (p. 371) that is a parent of this link.
----	---------------	--

10.73.3.6 `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	<i>_force</i>	Force to add.
----	---------------	---------------

10.73.3.7 `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	<i>_torque</i>	Torque value to add.
----	----------------	----------------------

10.73.3.8 `virtual void gazebo::physics::Link::AddTorque ( const math::Vector3 & _torque )` [pure virtual]

Add a torque to the body.

## Parameters

in	<i>_torque</i>	Torque value to add to the link.
----	----------------	----------------------------------

10.73.3.9 `void gazebo::physics::Link::AttachStaticModel ( ModelPtr & _model, const math::Pose & _offset )`

Attach a static model to this link.

## Parameters

in	<i>_model</i>	Pointer to a static model.
in	<i>_offset</i>	Pose relative to this link to place the model.

10.73.3.10 `template<typename T > event::ConnectionPtr gazebo::physics::Link::ConnectEnabled ( T _subscriber )`  
[inline]

Connect to the add entity signal.

## Parameters

in	<i>_subscriber</i>	Subscriber callback function.
----	--------------------	-------------------------------

## Returns

Pointer to the connection, which must be kept in scope.

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

10.73.3.11 `void gazebo::physics::Link::DetachAllStaticModels ( )`

Detach all static models from this link.

10.73.3.12 `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.
----	-------------------	--------------------------------------

10.73.3.13 void gazebo::physics::Link::DisconnectEnabled ( event::ConnectionPtr & \_conn ) [inline]

Disconnect to the add entity signal.

#### Parameters

in	_conn	Connection pointer to disconnect.
----	-------	-----------------------------------

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

10.73.3.14 void gazebo::physics::Link::FillMsg ( msgs::Link & \_msg )

Fill a link message.

#### Parameters

out	_msg	Message to fill
-----	------	-----------------

10.73.3.15 void gazebo::physics::Link::Fini ( ) [virtual]

Finalize the body.

Reimplemented from **gazebo::physics::Entity** (p. 277).

10.73.3.16 double gazebo::physics::Link::GetAngularDamping ( ) const

Get the angular damping factor.

#### Returns

Angular damping.

10.73.3.17 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. 277).

10.73.3.18 Link\_V gazebo::physics::Link::GetChildJointsLinks ( ) const

Returns a vector of children Links connected by joints.

#### Returns

**A** (p. 107) vector of children Links connected by joints.

**10.73.3.19 CollisionPtr gazebo::physics::Link::GetCollision ( const std::string & *\_name* )**

Get a child collision by name.

**Parameters**

<i>in</i>	<i>_name</i>	Name of the collision object.
-----------	--------------	-------------------------------

**Returns**

Pointer to the collision, NULL if the name was not found.

**10.73.3.20 CollisionPtr gazebo::physics::Link::GetCollision ( unsigned int *\_index* ) const**

Get a child collision by index.

**Parameters**

<i>in</i>	<i>_index</i>	Index of the collision object.
-----------	---------------	--------------------------------

**Returns**

Pointer to the collision, NULL if the name was not found.

**10.73.3.21 CollisionPtr gazebo::physics::Link::GetCollisionById ( unsigned int *\_id* ) const**

Get a collision by id.

**Parameters**

<i>in</i>	<i>_id</i>	Id of the collision object to find.
-----------	------------	-------------------------------------

**Returns**

Pointer to the collision, NULL if the id is invalid.

**10.73.3.22 Collision\_V gazebo::physics::Link::GetCollisions ( ) const**

Get all the child collisions.

**Returns**

**A** (p. 107) std::vector of all the child collisions.

**10.73.3.23 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.



10.73.3.24 `virtual bool gazebo::physics::Link::GetGravityMode ( ) [pure virtual]`

Get the gravity mode.

**Returns**

True if gravity is enabled.

10.73.3.25 `InertiaPtr gazebo::physics::Link::GetInertial ( ) const [inline]`

Get the inertia of the link.

**Returns**

Inertia of the link.

References inertial.

10.73.3.26 `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.

10.73.3.27 `double gazebo::physics::Link::GetLinearDamping ( ) const`

Get the linear damping factor.

**Returns**

Linear damping.

10.73.3.28 `ModelPtr gazebo::physics::Link::GetModel ( ) const`

Get the model that this body belongs to.

**Returns**

**Model** (p. 469) that this body belongs to.

10.73.3.29 `Link_V gazebo::physics::Link::GetParentJointsLinks ( ) const`

Returns a vector of parent Links connected by joints.

**Returns**

Vector of parent Links connected by joints.

10.73.3.30 **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. 279).

10.73.3.31 **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. 279).

10.73.3.32 **math::Vector3** gazebo::physics::Link::GetRelativeForce ( ) const

Get the force applied to the body.

**Returns**

Force applied to the body.

10.73.3.33 **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. 279).

10.73.3.34 **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. 280).

10.73.3.35 `math::Vector3 gazebo::physics::Link::GetRelativeTorque ( ) const`

Get the torque applied to the body.

#### Returns

Torque applied to the body.

10.73.3.36 `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.

10.73.3.37 `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. 415).

#### Returns

The number of sensors created by the link.

10.73.3.38 `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. 415).

#### Note

**A** (p. 107) **Link** (p. 404) does not manage or maintain a pointer to a **sensors::Sensor** (p. 672). Access to a Sensor object is accomplished through the **sensors::SensorManager** (p. 683). 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.

10.73.3.39 `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.280).

10.73.3.40 `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.

10.73.3.41 `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.280).

10.73.3.42 `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.

10.73.3.43 `virtual void gazebo::physics::Link::Init ( ) [virtual]`

Initialize the body.

Reimplemented from **gazebo::physics::Base** (p.141).

10.73.3.44 `virtual void gazebo::physics::Link::Load ( sdf::ElementPtr _sdf ) [virtual]`

Load the body based on an SDF element.

**Parameters**

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

Reimplemented from **gazebo::physics::Entity** (p.281).

10.73.3.45 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. 282).

10.73.3.46 void gazebo::physics::Link::ProcessMsg ( const msgs::Link & \_msg )

Update parameters from a message.

#### Parameters

in	<i>_msg</i>	Message to read.
----	-------------	------------------

10.73.3.47 void gazebo::physics::Link::RemoveChildJoint ( JointPtr *joint* )

Remove Joints that have this **Link** (p. 404) as a parent **Link** (p. 404).

#### Parameters

in	<i>joint</i>	<b>Joint</b> (p. 371) that is a child of this link.
----	--------------	---

10.73.3.48 void gazebo::physics::Link::RemoveParentJoint ( JointPtr *joint* )

Remove Joints that have this **Link** (p. 404) as a child **Link** (p. 404).

#### Parameters

in	<i>joint</i>	<b>Joint</b> (p. 371) that is a parent of this link.
----	--------------	--

10.73.3.49 void gazebo::physics::Link::Reset ( ) [virtual]

Reset the link.

Reimplemented from **gazebo::physics::Entity** (p. 282).

10.73.3.50 void gazebo::physics::Link::SetAngularAccel ( const math::Vector3 & *accel* )

Set the angular acceleration of the body.

#### Parameters

in	<i>accel</i>	Angular acceleration.
----	--------------	-----------------------

10.73.3.51 virtual void gazebo::physics::Link::SetAngularDamping ( double *damping* ) [pure virtual]

Set the angular damping factor.

## Parameters

in	<i>_damping</i>	Angular damping factor.
----	-----------------	-------------------------

10.73.3.52 `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.
----	-------------	-------------------

10.73.3.53 `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.
----	-----------------	---

10.73.3.54 `void gazebo::physics::Link::SetCollideMode ( const std::string & _mode )`

Set the collide mode of the body.

## Parameters

in	<i>_mode</i>	<b>Collision</b> (p. 190) 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
----	--------------	--

10.73.3.55 `virtual void gazebo::physics::Link::SetEnabled ( bool _enable ) const [pure virtual]`

Set whether this body is enabled.

## Parameters

in	<i>_enable</i>	True to enable the link in the physics engine.
----	----------------	--

10.73.3.56 `virtual void gazebo::physics::Link::SetForce ( const math::Vector3 & _force ) [pure virtual]`

Set the force applied to the body.

## Parameters

in	<i>_force</i>	Force value.
----	---------------	--------------

10.73.3.57 virtual void gazebo::physics::Link::SetGravityMode ( bool *\_mode* ) [pure virtual]

Set whether gravity affects this body.

#### Parameters

in	<i>_mode</i>	True to enable gravity.
----	--------------	-------------------------

10.73.3.58 void gazebo::physics::Link::SetInertial ( const InertialPtr & *\_inertial* )

Set the mass of the link.

[in] *\_inertial* **Inertial** (p. 360) value for the link.

10.73.3.59 virtual void gazebo::physics::Link::SetKinematic ( const bool & *\_kinematic* ) [virtual]

Implement this function.

Set whether this body is in the kinematic state.

#### Parameters

in	<i>_kinematic</i>	True to make the link kinematic only.
----	-------------------	---------------------------------------

10.73.3.60 void gazebo::physics::Link::SetLaserRetro ( float *\_retro* )

Set the laser retro reflectiveness.

#### Parameters

in	<i>_retro</i>	Retro value for all child collisions.
----	---------------	---------------------------------------

10.73.3.61 void gazebo::physics::Link::SetLinearAccel ( const math::Vector3 & *\_accel* )

Set the linear acceleration of the body.

#### Parameters

in	<i>_accel</i>	Linear acceleration.
----	---------------	----------------------

10.73.3.62 virtual void gazebo::physics::Link::SetLinearDamping ( double *\_damping* ) [pure virtual]

Set the linear damping factor.

#### Parameters

in	<i>_damping</i>	Linear damping factor.
----	-----------------	------------------------

10.73.3.63 `virtual void gazebo::physics::Link::SetLinearVel ( const math::Vector3 & _vel ) [pure virtual]`

Set the linear velocity of the body.

Parameters

in	_vel	Linear velocity.
----	------	------------------

10.73.3.64 `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. 143).

10.73.3.65 `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.
----	---------	----------------------------

10.73.3.66 `void gazebo::physics::Link::SetState ( const LinkState & _state )`

Set the current link state.

Parameters

in	_state	The state to set the link to.
----	--------	-------------------------------

10.73.3.67 `virtual void gazebo::physics::Link::SetTorque ( const math::Vector3 & _torque ) [pure virtual]`

Set the torque applied to the body.

Parameters

in	_torque	Torque value.
----	---------	---------------

10.73.3.68 `virtual void gazebo::physics::Link::Update ( ) [virtual]`

Update the body.

Reimplemented from `gazebo::physics::Base` (p. 143).



10.73.3.69 virtual void gazebo::physics::Link::UpdateMass ( ) [inline],[virtual]

Update the mass matrix.

10.73.3.70 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. 284).

10.73.3.71 virtual void gazebo::physics::Link::UpdateSurface ( ) [inline],[virtual]

Update surface parameters.

### 10.73.4 Member Data Documentation

10.73.4.1 math::Vector3 gazebo::physics::Link::angularAccel [protected]

Angular acceleration.

10.73.4.2 std::vector<math::Pose> gazebo::physics::Link::attachedModelsOffset [protected]

Offsets for the attached models.

10.73.4.3 std::vector<std::string> gazebo::physics::Link::cgVisuals [protected]

Center of gravity visual elements.

10.73.4.4 InertialPtr gazebo::physics::Link::inertial [protected]

**Inertial** (p. 360) properties.

Referenced by GetInertial().

10.73.4.5 math::Vector3 gazebo::physics::Link::linearAccel [protected]

Linear acceleration.

10.73.4.6 std::vector<std::string> gazebo::physics::Link::visuals [protected]

**Link** (p. 404) visual elements.

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

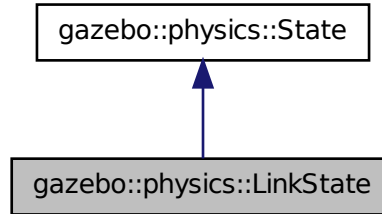
- **Link.hh**

## 10.74 gazebo::physics::LinkState Class Reference

Store state information of a **physics::Link** (p. 404) 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. 404).*
- 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

### 10.74.1 Detailed Description

Store state information of a **physics::Link** (p. 404) object.

This class captures the entire state of a **Link** (p. 404) at one specific time during a simulation run.

**State** (p. 729) of a **Link** (p. 404) includes the state of itself all its child **Collision** (p. 190) entities.

### 10.74.2 Constructor & Destructor Documentation

#### 10.74.2.1 gazebo::physics::LinkState::LinkState ( )

Default constructor.

#### 10.74.2.2 gazebo::physics::LinkState::LinkState ( const **LinkPtr** \_link ) [explicit]

Constructor.

Build a **LinkState** (p. 422) from an existing **Link** (p. 404).

#### Parameters

in	<b>_model</b>	Pointer to the <b>Link</b> (p. 404) from which to gather state info.
----	---------------	--

#### 10.74.2.3 gazebo::physics::LinkState::LinkState ( const **sdf::ElementPtr** \_sdf ) [explicit]

Constructor.

Build a **LinkState** (p. 422) from SDF data

## Parameters

in	<code>_sdf</code>	SDF data to load a link state from.
----	-------------------	-------------------------------------

10.74.2.4 `virtual gazebo::physics::LinkState::~~LinkState ( ) [virtual]`

Destructor.

### 10.74.3 Member Function Documentation

10.74.3.1 `void gazebo::physics::LinkState::FillSDF ( sdf::ElementPtr _sdf )`

Populate a state SDF element with data from the object.

## Parameters

out	<code>_sdf</code>	SDF element to populate.
-----	-------------------	--------------------------

10.74.3.2 `const math::Pose& gazebo::physics::LinkState::GetAcceleration ( ) const`

Get the link acceleration.

## Returns

The acceleration represented as a **math::Pose** (p. 573).

10.74.3.3 `CollisionState gazebo::physics::LinkState::GetCollisionState ( unsigned int _index ) const`

Get a collision state.

Get a **Collision** (p. 190) **State** (p. 729) based on an index, where index is in the range of 0...**LinkState::GetCollisionStateCount** (p. 425).

## Parameters

in	<code>_index</code>	Index of the <b>CollisionState</b> (p. 199).
----	---------------------	--

## Returns

**State** (p. 729) of the **Collision** (p. 190).

## Exceptions

<b>common::Exception</b> (p. 310)	When <code>_index</code> is invalid.
--------------------------------------	--------------------------------------

10.74.3.4 `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. 199) with the matching name, if any.

#### Parameters

<code>in</code>	<code>_collisionName</code>	Name of the <b>CollisionState</b> (p. 199)
-----------------	-----------------------------	--

#### Returns

**State** (p. 729) of the **Collision** (p. 190).

#### Exceptions

<b><i>common::Exception</i></b> (p. 310)	When <code>_collisionName</code> is invalid
---	---

#### 10.74.3.5 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. 199) recorded.

#### 10.74.3.6 const std::vector<CollisionState>& gazebo::physics::LinkState::GetCollisionStates ( ) const

Get the collision states.

#### Returns

**A** (p. 107) vector of collision states.

#### 10.74.3.7 const math::Pose& gazebo::physics::LinkState::GetPose ( ) const

Get the link pose.

#### Returns

The **math::Pose** (p. 573) of the **Link** (p. 404).

#### 10.74.3.8 const math::Pose& gazebo::physics::LinkState::GetVelocity ( ) const

Get the link velocity.

#### Returns

The velocity represented as a **math::Pose** (p. 573).

10.74.3.9 `const math::Pose& gazebo::physics::LinkState::GetWrench ( ) const`

Get the force applied to the **Link** (p. 404).

#### Returns

Magnitude of the force.

10.74.3.10 `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.

10.74.3.11 `virtual void gazebo::physics::LinkState::Load ( const sdf::ElementPtr _elem ) [virtual]`

Load state from SDF element.

Load **LinkState** (p. 422) 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. 732).

10.74.3.12 `LinkState gazebo::physics::LinkState::operator+ ( const LinkState & _state ) const`

Addition operator.

#### Parameters

<code>in</code>	<code>_pt</code>	<b>A</b> (p. 107) state to add.
-----------------	------------------	---------------------------------

#### Returns

The resulting state.

10.74.3.13 `LinkState gazebo::physics::LinkState::operator- ( const LinkState & _state ) const`

Subtraction operator.

#### Parameters

<code>in</code>	<code>_pt</code>	<b>A</b> (p. 107) state to subtract.
-----------------	------------------	--------------------------------------

**Returns**

The resulting state.

**10.74.3.14 LinkState& gazebo::physics::LinkState::operator= ( const LinkState & *\_state* )**

Assignment operator.

**Parameters**

<i>in</i>	<i>_state</i>	<b>State</b> (p. 729) value
-----------	---------------	-----------------------------

**Returns**

this

**10.74.4 Friends And Related Function Documentation****10.74.4.1 std::ostream& operator<< ( std::ostream & *\_out*, const gazebo::physics::LinkState & *\_state* ) [friend]**

Stream insertion operator.

**Parameters**

<i>in</i>	<i>_out</i>	output stream
<i>in</i>	<i>_state</i>	<b>Link</b> (p. 404) state to output

**Returns**

the stream

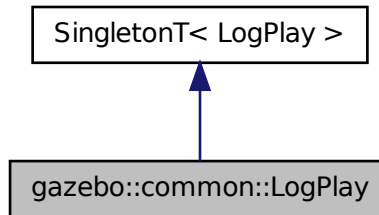
The documentation for this class was generated from the following file:

- **LinkState.hh**

**10.75 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

### 10.75.1 Member Function Documentation

#### 10.75.1.1 bool gazebo::common::LogPlay::GetChunk ( unsigned int *\_index*, std::string & *\_data* )

Get data for a particular chunk index.

#### Parameters

in	<i>_index</i>	Index of the chunk.
out	<i>_data</i>	Storage for the chunk's data.



**Returns**

True if the `_index` was valid.

**10.75.1.2** `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.

**10.75.1.3** `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. 430) has not been called at least once.

**10.75.1.4** `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.

**10.75.1.5** `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.

**10.75.1.6** `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. 613).

**10.75.1.7** bool gazebo::common::LogPlay::IsOpen ( ) const

Return true if a file is open.

**Returns**

True if a log file is open.

**10.75.1.8** 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	<i>_logFile</i>	The file to load
----	-----------------	------------------

**Exceptions**

<b><i>Exception</i></b> (p. 310)
----------------------------------

**10.75.1.9** bool gazebo::common::LogPlay::Step ( std::string & \_data )

Step through the open log file.

**Parameters**

out	<i>_data</i>	Data from next entry in the log file.
-----	--------------	---------------------------------------

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

- **LogPlay.hh**

## 10.76 Logplay Class Reference

Open and playback log files that were recorded using LogRecord.

### 10.76.1 Detailed Description

Open and playback log files that were recorded using LogRecord.

Use **Logplay** (p. 430) 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:

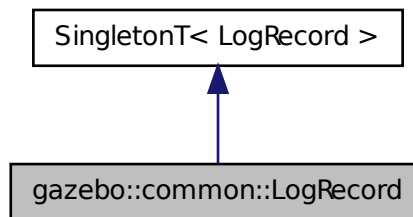
- [LogPlay.hh](#)

## 10.77 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::ostream &)> \_logCallback)  
*Add an object to a log file.*
- const std::string & **GetEncoding** () const  
*Get the encoding used.*
- bool **GetRunning** () const  
*Return true if running.*
- bool **Init** (const std::string &\_subdir)  
*Initialize logging into a subdirectory.*
- bool **Remove** (const std::string &\_name)  
*Remove an entity from a log.*
- void **Start** (const std::string &\_encoding="bz2")  
*Start the logger.*
- void **Stop** ()  
*Stop the logger.*

### Additional Inherited Members

#### 10.77.1 Detailed Description

addtogroup gazebo\_common

Handles logging of data to disk

The **LogRecord** (p. 431) 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. 427) for playback functionality.

State information for an entity may be logged through the **LogRecord::Add** (p. 432) function, and stopped through the **LogRecord::Remove** (p. 433) function. Data may be logged into a single file, or split into many separate files by specifying different filenames for the **LogRecord::Add** (p. 432) function.

The **LogRecord** (p. 431) is updated at the start of each simulation step. This guarantees that all data is stored.

#### See Also

**Logplay** (p. 430), State

## 10.77.2 Member Function Documentation

10.77.2.1 `void gazebo::common::LogRecord::Add ( const std::string & _name, const std::string & _filename, boost::function< bool(std::ostream &)> _logCallback )`

Add an object to a log file.

Add a new object to a log. An object can be any valid named object in simulation, including the world itself. Duplicate additions are ignored. Objects can be added to the same file by specifying the same `_filename`.

#### Parameters

<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

<b>Exception</b> (p. 310)
---------------------------

10.77.2.2 `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.

10.77.2.3 `bool gazebo::common::LogRecord::GetRunning ( ) const`

Return true if running.

#### Returns

True if **LogRecord** (p. 431) has been started.

**10.77.2.4** `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><i>_subdir</i></code>	Directory to record to
-----------------	-----------------------------	------------------------

**Returns**

True if successful.

**10.77.2.5** `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><i>_name</i></code>	Name of the log
-----------------	---------------------------	-----------------

**Returns**

True if the entity existed and was removed. False if the entity was not registered with the logger.

**10.77.2.6** `void gazebo::common::LogRecord::Start ( const std::string & _encoding = "bz2" )`

Start the logger.

**Parameters**

<code>in</code>	<code><i>_encoding</i></code>	The type of encoding (txt, or bz2).
-----------------	-------------------------------	-------------------------------------

**10.77.2.7** `void gazebo::common::LogRecord::Stop ( )`

Stop the logger.

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

- **LogRecord.hh**

**10.78 gazebo::Master Class Reference**

**A** (p. 107) ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

```
#include <gazebo_core.hh>
```

## Public Member Functions

- **Master** ()  
*Constructor.*
- virtual **~Master** ()  
*Destructor.*
- void **Fini** ()  
*Finalize the master.*
- void **Init** (uint16\_t \_port)  
*Initialize.*
- void **Run** ()  
*Run the master.*
- void **RunOnce** ()  
*Run the master one iteration.*
- void **RunThread** ()  
*Run the master in a new thread.*
- void **Stop** ()  
*Stop the master.*

### 10.78.1 Detailed Description

**A** (p. 107) 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. 433), runs World update and sensor generation loops.

### 10.78.2 Constructor & Destructor Documentation

#### 10.78.2.1 gazebo::Master::Master ( )

Constructor.

#### 10.78.2.2 virtual gazebo::Master::~~Master ( ) [virtual]

Destructor.

### 10.78.3 Member Function Documentation

#### 10.78.3.1 void gazebo::Master::Fini ( )

Finalize the master.

#### 10.78.3.2 void gazebo::Master::Init ( uint16\_t \_port )

Initialize.

## Parameters

in	<code>_port</code>	The master's port
----	--------------------	-------------------

## 10.78.3.3 void gazebo::Master::Run ( )

Run the master.

## 10.78.3.4 void gazebo::Master::RunOnce ( )

Run the master one iteration.

## 10.78.3.5 void gazebo::Master::RunThread ( )

Run the master in a new thread.

## 10.78.3.6 void gazebo::Master::Stop ( )

Stop the master.

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

- **Master.hh**

## 10.79 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.*

### 10.79.1 Detailed Description

Encapsulates description of a material.

## 10.79.2 Member Enumeration Documentation

### 10.79.2.1 enum gazebo::common::Material::BlendMode

Enumerator:

**ADD**  
**MODULATE**  
**REPLACE**  
**BLEND\_COUNT**

### 10.79.2.2 enum gazebo::common::Material::ShadeMode

Enumerator:

**FLAT**  
**GOURAUD**  
**PHONG**  
**BLINN**  
**SHADE\_COUNT**

## 10.79.3 Constructor & Destructor Documentation

### 10.79.3.1 gazebo::common::Material::Material ( )

Constructor.

### 10.79.3.2 virtual gazebo::common::Material::~Material ( ) [virtual]

Destructor.

### 10.79.3.3 gazebo::common::Material::Material ( const Color & \_clr )

Create a material with a default color.

Parameters

in	<i>_clr</i>	<b>Color</b> (p. 203) of the material
----	-------------	---------------------------------------

## 10.79.4 Member Function Documentation

### 10.79.4.1 Color gazebo::common::Material::GetAmbient ( ) const

Get the ambient color.

Returns

The ambient color

10.79.4.2 void gazebo::common::Material::GetBlendFactors ( double & *\_srcFactor*, double & *\_dstFactor* )

Get the blend factors.

Parameters

in	<i>_srcFactor</i>	Source factor is returned in this variable
in	<i>_dstFactor</i>	Destination factor is returned in this variable

10.79.4.3 **BlendMode** gazebo::common::Material::GetBlendMode ( ) const

Get the blending mode.

Returns

the blend mode

10.79.4.4 bool gazebo::common::Material::GetDepthWrite ( ) const

Get depth write.

Returns

the depth write enabled state

10.79.4.5 **Color** gazebo::common::Material::GetDiffuse ( ) const

Get the diffuse color.

Returns

The diffuse color

10.79.4.6 **Color** gazebo::common::Material::GetEmissive ( ) const

Get the emissive color.

Returns

The emissive color

10.79.4.7 bool gazebo::common::Material::GetLighting ( ) const

Get lighting enabled.

Returns

the lighting enabled state

10.79.4.8 `std::string gazebo::common::Material::GetName ( ) const`

Get the name of the material.

**Returns**

The name of the material

10.79.4.9 `double gazebo::common::Material::GetPointSize ( ) const`

Get the point size.

**Returns**

the point size

10.79.4.10 `ShadeMode gazebo::common::Material::GetShadeMode ( ) const`

Get the shading mode.

**Returns**

the shading mode

10.79.4.11 `double gazebo::common::Material::GetShininess ( ) const`

Get the shininess.

**Returns**

The shininess value

10.79.4.12 `Color gazebo::common::Material::GetSpecular ( ) const`

Get the specular color.

**Returns**

The specular color

10.79.4.13 `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

10.79.4.14 `double gazebo::common::Material::GetTransparency ( ) const`

Get the transparency percentage (0..1)

#### Returns

The transparency percentage

10.79.4.15 `void gazebo::common::Material::SetAmbient ( const Color & _clr )`

Set the ambient color.

#### Parameters

in	_clr	The ambient color
----	------	-------------------

10.79.4.16 `void gazebo::common::Material::SetBlendFactors ( double _srcFactor, double _dstFactor )`

Set the blende factors.

Will be interpreted as:  $(\text{texture} * \_srcFactor) + (\text{scene\_pixel} * \_dstFactor)$

#### Parameters

in	_srcFactor	The source factor
in	_dstFactor	The destination factor

10.79.4.17 `void gazebo::common::Material::SetBlendMode ( BlendMode _b )`

Set the blending mode.

#### Parameters

in	_b	the blend mode
----	----	----------------

10.79.4.18 `void gazebo::common::Material::SetDepthWrite ( bool _value )`

Set depth write.

#### Parameters

in	_value	the depth write enabled state
----	--------	-------------------------------

10.79.4.19 `void gazebo::common::Material::SetDiffuse ( const Color & _clr )`

Set the diffuse color.

## Parameters

in	<code>_clr</code>	The diffuse color
----	-------------------	-------------------

10.79.4.20 `void gazebo::common::Material::SetEmissive ( const Color & _clr )`

Set the emissive color.

## Parameters

in	<code>_clr</code>	The emissive color
----	-------------------	--------------------

10.79.4.21 `void gazebo::common::Material::SetLighting ( bool _value )`

Set lighting enabled.

## Parameters

in	<code>_value</code>	the lighting enabled state
----	---------------------	----------------------------

10.79.4.22 `void gazebo::common::Material::SetPointSize ( double _size )`

Set the point size.

## Parameters

in	<code>_size</code>	the size
----	--------------------	----------

10.79.4.23 `void gazebo::common::Material::SetShadeMode ( ShadeMode _b )`

Set the shading mode param[in] the shading mode.

10.79.4.24 `void gazebo::common::Material::SetShininess ( double _t )`

Set the shininess.

## Parameters

in	<code>_t</code>	The shininess value
----	-----------------	---------------------

10.79.4.25 `void gazebo::common::Material::SetSpecular ( const Color & _clr )`

Set the specular color.

## Parameters

in	<code>_clr</code>	The specular color
----	-------------------	--------------------

10.79.4.26 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
----	-------------	--

10.79.4.27 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>

10.79.4.28 void gazebo::common::Material::SetTransparency ( double *\_t* )

Set the transparency percentage (0..1)

#### Parameters

in	<i>_t</i>	The amount of transparency (0..1)
----	-----------	-----------------------------------

## 10.79.5 Friends And Related Function Documentation

10.79.5.1 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.

## 10.79.6 Member Data Documentation

10.79.6.1 Color gazebo::common::Material::ambient [protected]

the ambient light color

10.79.6.2 BlendMode gazebo::common::Material::blendMode [protected]

blend mode

10.79.6.3 std::string gazebo::common::Material::BlendModeStr[BLEND\_COUNT] [static]

10.79.6.4 Color gazebo::common::Material::diffuse [protected]

the diffuse lighth color

10.79.6.5 **Color** gazebo::common::Material::emissive [protected]

the emissive light color

10.79.6.6 **std::string** gazebo::common::Material::name [protected]

the name of the material

10.79.6.7 **double** gazebo::common::Material::pointSize [protected]

point size

10.79.6.8 **ShadeMode** gazebo::common::Material::shadeMode [protected]

the shade mode

10.79.6.9 **std::string** gazebo::common::Material::ShadeModeStr[SHADE\_COUNT] [static]

10.79.6.10 **double** gazebo::common::Material::shininess [protected]

shininess value (0 to 1)

10.79.6.11 **Color** gazebo::common::Material::specular [protected]

the specular light color

10.79.6.12 **std::string** gazebo::common::Material::texImage [protected]

the texture image file name

10.79.6.13 **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**

## 10.80 gazebo::math::Matrix3 Class Reference

**A** (p. 107) 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** ()  
*Destructor.*
- 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.*

### 10.80.1 Detailed Description

**A** (p. 107) 3x3 matrix class.

### 10.80.2 Constructor & Destructor Documentation

#### 10.80.2.1 gazebo::math::Matrix3::Matrix3 ( )

Constructor.

### 10.80.2.2 gazebo::math::Matrix3::Matrix3 ( const Matrix3 & \_m )

Copy constructor.

#### Parameters

_m	Matrix to copy
----	----------------

### 10.80.2.3 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

### 10.80.2.4 virtual gazebo::math::Matrix3::~~Matrix3 ( ) [virtual]

Destructor.

## 10.80.3 Member Function Documentation

### 10.80.3.1 bool gazebo::math::Matrix3::operator==( const Matrix3 & \_m ) const

Equality test operator.

#### Parameters

in	_m	<b>Matrix3</b> (p. 444) to test
----	----	---------------------------------

#### Returns

True if equal (using the default tolerance of 1e-6)

### 10.80.3.2 const double\* gazebo::math::Matrix3::operator[]( size\_t \_row ) const [inline]

Array subscript operator.

#### Parameters

in	_row	row index
----	------	-----------

**Returns**

a pointer to the row

References m.

**10.80.3.3** `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.

**10.80.3.4** `void gazebo::math::Matrix3::SetCol ( unsigned int _c, const Vector3 & _v )`

Set a column.

**Parameters**

<code>in</code>	<code>_c</code>	The colum index (0, 1, 2)
<code>in</code>	<code>_v</code>	The value to set in each row of the column

**10.80.3.5** `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

**10.80.3.6** `void gazebo::math::Matrix3::SetFromAxis ( const Vector3 & _axis, double _angle )`

Set the matrix from an axis and angle.

**Parameters**

<code>in</code>	<code>_axis</code>	the axis
<code>in</code>	<code>_angle</code>	ccw rotation around the axis in radians

**10.80.4 Friends And Related Function Documentation**

10.80.4.1 `std::ostream& operator<< ( std::ostream & _out, const gazebo::math::Matrix3 & _m )` [friend]

Stream insertion operator.

#### Parameters

in	<code>_out</code>	Output stream
in	<code>_m</code>	Matrix to output

#### Returns

the stream

### 10.80.5 Member Data Documentation

10.80.5.1 `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**

## 10.81 gazebo::math::Matrix4 Class Reference

**A** (p. 107) 3x3 matrix class.

```
#include <math/gzmath.hh>
```

### Public Member Functions

- **Matrix4 ()**  
*Constructor.*
- **Matrix4 (const Matrix4 &\_m)**  
*Copy constructor.*
- **Matrix4 (double \_v00, double \_v01, double \_v02, double \_v03, double \_v10, double \_v11, double \_v12, double \_v13, double \_v20, double \_v21, double \_v22, double \_v23, double \_v30, double \_v31, double \_v32, double \_v33)**  
*Constructor.*
- virtual **~Matrix4 ()**  
*Destructor.*
- **math::Pose GetAsPose () const**  
*Get the transformation as **math::Pose** (p. 573).*
- **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. 821).

- **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.

### 10.81.1 Detailed Description

**A** (p. 107) 3x3 matrix class.

### 10.81.2 Constructor & Destructor Documentation

#### 10.81.2.1 gazebo::math::Matrix4::Matrix4 ( )

Constructor.

#### 10.81.2.2 gazebo::math::Matrix4::Matrix4 ( const Matrix4 & \_m )

Copy constructor.

##### Parameters

_m	Matrix to copy
----	----------------

#### 10.81.2.3 gazebo::math::Matrix4::Matrix4 ( double \_v00, double \_v01, double \_v02, double \_v03, double \_v10, double \_v11, double \_v12, double \_v13, double \_v20, double \_v21, double \_v22, double \_v23, double \_v30, double \_v31, double \_v32, double \_v33 )

Constructor.

##### Parameters

in	_v00	Row 0, Col 0 value
in	_v01	Row 0, Col 1 value
in	_v02	Row 0, Col 2 value
in	_v03	Row 0, Col 3 value
in	_v10	Row 1, Col 0 value
in	_v11	Row 1, Col 1 value
in	_v12	Row 1, Col 2 value
in	_v13	Row 1, Col 3 value
in	_v20	Row 2, Col 0 value
in	_v21	Row 2, Col 1 value
in	_v22	Row 2, Col 2 value
in	_v23	Row 2, Col 3 value
in	_v30	Row 3, Col 0 value
in	_v31	Row 3, Col 1 value
in	_v32	Row 3, Col 2 value
in	_v33	Row 3, Col 3 value

#### 10.81.2.4 virtual gazebo::math::Matrix4::~~Matrix4 ( ) [virtual]

Destructor.

### 10.81.3 Member Function Documentation

10.81.3.1 **math::Pose** gazebo::math::Matrix4::GetAsPose ( ) const

Get the transformation as **math::Pose** (p. 573).

Returns

the pose

10.81.3.2 **Vector3** gazebo::math::Matrix4::GetEulerRotation ( unsigned int *solution\_number* = 1 ) const

Get the rotation as a Euler angles.

Returns

the rotation

10.81.3.3 **Quaternion** gazebo::math::Matrix4::GetRotation ( ) const

Get the rotation as a quaternion.

Returns

the rotation

10.81.3.4 **Vector3** gazebo::math::Matrix4::GetTranslation ( ) const

Get the translational values as a **Vector3** (p. 821).

Returns

x,y,z

10.81.3.5 **Matrix4** gazebo::math::Matrix4::Inverse ( ) const

Return the inverse matrix.

10.81.3.6 **bool** gazebo::math::Matrix4::IsAffine ( ) const

Return true if the matrix is affine.

Returns

true if the matrix is affine, false otherwise

10.81.3.7 **Matrix4** gazebo::math::Matrix4::operator\* ( const **Matrix4** & *\_mat* ) const

Multiplication operator.

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

**Returns**

This matrix \* `_mat`

**10.81.3.8 Matrix4 gazebo::math::Matrix4::operator\* ( const Matrix3 & \_mat ) const**

Multiplication operator.

**Parameters**

<code>_mat</code>	Incoming matrix
-------------------	-----------------

**Returns**

This matrix \* `_mat`

**10.81.3.9 Vector3 gazebo::math::Matrix4::operator\* ( const Vector3 & \_vec ) const**

Multiplication operator.

**Parameters**

<code>_vec</code>	<b>Vector3</b> (p. 821)
-------------------	-------------------------

**Returns**

Resulting vector from multiplication

**10.81.3.10 Matrix4& gazebo::math::Matrix4::operator= ( const Matrix4 & \_mat )**

Equal operator.

this = `_mat`

**Parameters**

<code>_mat</code>	Incoming matrix
-------------------	-----------------

**Returns**

itself

**10.81.3.11 const Matrix4& gazebo::math::Matrix4::operator= ( const Matrix3 & \_mat )**

Equal operator for 3x3 matrix.



## Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

## Returns

itself

10.81.3.12 `bool gazebo::math::Matrix4::operator==( const Matrix4 & _m ) const`

Equality operator.

## Parameters

<code>in</code>	<code>_m</code>	<b>Matrix3</b> (p. 444) to test
-----------------	-----------------	---------------------------------

## Returns

true if the 2 matrices are equal (using the tolerance 1e-6), false otherwise

10.81.3.13 `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.

10.81.3.14 `const double* gazebo::math::Matrix4::operator[]( size_t _row ) const [inline]`

## Parameters

<code>in</code>	<code>_row</code>	the row index
-----------------	-------------------	---------------

## Returns

the row

References m.

10.81.3.15 `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

## 10.81.3.16 void gazebo::math::Matrix4::SetScale ( const Vector3 &amp; \_s )

Set the scale.

## Parameters

in	<code>_s</code>	scale
----	-----------------	-------

## 10.81.3.17 void gazebo::math::Matrix4::SetTranslate ( const Vector3 &amp; \_t )

Set the translational values [ (0, 3) (1, 3) (2, 3) ].

## Parameters

in	<code>_t</code>	Values to set
----	-----------------	---------------

## 10.81.3.18 Vector3 gazebo::math::Matrix4::TransformAffine ( const Vector3 &amp; \_v ) const

Perform an affine transformation.

## Parameters

<code>_v</code>	<b>Vector3</b> (p. 821) value for the transformation
-----------------	--

## Returns

The result of the transformation

## 10.81.4 Friends And Related Function Documentation

10.81.4.1 `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

### 10.81.5 Member Data Documentation

10.81.5.1 `const Matrix4 gazebo::math::Matrix4::IDENTITY` [static]

Identity matrix.

10.81.5.2 `double gazebo::math::Matrix4::m[4][4]` [protected]

The 4x4 matrix.

Referenced by `operator[]()`.

10.81.5.3 `const Matrix4 gazebo::math::Matrix4::ZERO` [static]

Zero matrix.

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

- **Matrix4.hh**

## 10.82 gazebo::common::Mesh Class Reference

**A** (p. 107) 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.*

### 10.82.1 Detailed Description

**A** (p. 107) 3D mesh.

### 10.82.2 Constructor & Destructor Documentation

#### 10.82.2.1 gazebo::common::Mesh::Mesh ( )

Constructor.

#### 10.82.2.2 virtual gazebo::common::Mesh::~~Mesh ( ) [virtual]

Destructor.

### 10.82.3 Member Function Documentation

#### 10.82.3.1 int gazebo::common::Mesh::AddMaterial ( **Material** \* *\_mat* )

Add a material to the mesh.

##### Parameters

in	<i>_mat</i>	the material
----	-------------	--------------

##### Returns

Index of this material

#### 10.82.3.2 void gazebo::common::Mesh::AddSubMesh ( **SubMesh** \* *\_child* )

Add a submesh mesh.

The **Mesh** (p. 455) object takes ownership of the submesh.

##### Parameters

in	<i>_child</i>	the submesh
----	---------------	-------------

#### 10.82.3.3 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

10.82.3.4 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
----	----------------	------------------------------

10.82.3.5 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

10.82.3.6 unsigned int gazebo::common::Mesh::GetIndexCount ( ) const

Return the number of indices.

Returns

the count

10.82.3.7 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

10.82.3.8 unsigned int gazebo::common::Mesh::GetMaterialCount ( ) const

Get the number of materials.

Returns

the count

10.82.3.9 math::Vector3 gazebo::common::Mesh::GetMax ( ) const

Get the maximum X, Y, Z values.

**Returns**

the upper bounds of the bounding box

**10.82.3.10** `math::Vector3 gazebo::common::Mesh::GetMin ( ) const`

Get the minimum X, Y, Z values.

**Returns**

the lower bounds of the bounding box

**10.82.3.11** `std::string gazebo::common::Mesh::GetName ( ) const`

Get the name of this mesh.

**Returns**

the name

**10.82.3.12** `unsigned int gazebo::common::Mesh::GetNormalCount ( ) const`

Return the number of normals.

**Returns**

the count

**10.82.3.13** `std::string gazebo::common::Mesh::GetPath ( ) const`

Get the path which contains the mesh resource.

**Returns**

the path to the mesh resource

**10.82.3.14** `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.

**10.82.3.15** `const SubMesh* gazebo::common::Mesh::GetSubMesh ( unsigned int i ) const`

Get a child mesh.

**Parameters**

<code>in</code>	<code>_i</code>	the index
-----------------	-----------------	-----------

**Returns**

the submesh. An exception is thrown if the index is out of bounds

**10.82.3.16** `unsigned int gazebo::common::Mesh::GetSubMeshCount ( ) const`

Get the number of children.

**Returns**

the count

**10.82.3.17** `unsigned int gazebo::common::Mesh::GetTexCoordCount ( ) const`

Return the number of texture coordinates.

**Returns**

the count

**10.82.3.18** `unsigned int gazebo::common::Mesh::GetVertexCount ( ) const`

Return the number of vertices.

**Returns**

the count

**10.82.3.19** `bool gazebo::common::Mesh::HasSkeleton ( ) const`

Return true if mesh is attached to a skeleton.

**10.82.3.20** `void gazebo::common::Mesh::RecalculateNormals ( )`

Recalculate all the normals of each face defined by three indices.

**10.82.3.21** `void gazebo::common::Mesh::Scale ( double _factor )`

Scale all vertices by `_factor`.

**Parameters**

<code><i>_factor</i></code>	Scaling factor
-----------------------------	----------------



10.82.3.22 void gazebo::common::Mesh::SetName ( const std::string & *\_n* )

Set the name of this mesh.

#### Parameters

in	<i>_n</i>	the name to set
----	-----------	-----------------

10.82.3.23 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
----	--------------	---------------

10.82.3.24 void gazebo::common::Mesh::SetScale ( const math::Vector3 & *\_factor* )

Scale all vertices by the *\_factor* vector.

#### Parameters

in	<i>_factor</i>	Scaling vector
----	----------------	----------------

10.82.3.25 void gazebo::common::Mesh::SetSkeleton ( Skeleton \* *\_skel* )

Set the mesh skeleton.

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

- **Mesh.hh**

## 10.83 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.*

### 10.83.1 Detailed Description

Creates CSG meshes.

### 10.83.2 Member Enumeration Documentation

#### 10.83.2.1 enum gazebo::common::MeshCSG::BooleanOperation

An enumeration of the boolean operations.

Enumerator:

**UNION**

**INTERSECTION**

**DIFFERENCE**

### 10.83.3 Constructor & Destructor Documentation

#### 10.83.3.1 gazebo::common::MeshCSG::MeshCSG ( )

Constructor.

#### 10.83.3.2 virtual gazebo::common::MeshCSG::~~MeshCSG ( ) [virtual]

Destructor.

### 10.83.4 Member Function Documentation

#### 10.83.4.1 **Mesh\*** gazebo::common::MeshCSG::CreateBoolean ( const **Mesh** \*\_m1, const **Mesh** \*\_m2, const int \_operation, const **math::Pose** & \_offset = **math::Pose::Zero** )

Create a boolean mesh from two meshes.

#### Parameters

in	<i>_m1</i>	the parent mesh in the boolean operation
in	<i>_m2</i>	the child mesh in the boolean operation
in	<i>_operation</i>	the boolean operation applied to the two meshes
in	<i>_offset</i>	_m2's pose offset from _m1

#### Returns

a pointer to the created mesh

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

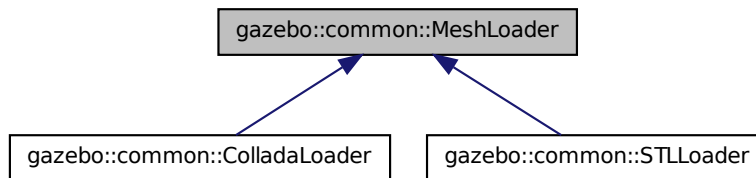
- MeshCSG.hh

## 10.84 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.*

#### 10.84.1 Detailed Description

Base class for loading meshes.

#### 10.84.2 Constructor & Destructor Documentation

##### 10.84.2.1 gazebo::common::MeshLoader::MeshLoader ( )

Constructor.

##### 10.84.2.2 virtual gazebo::common::MeshLoader::~~MeshLoader ( ) [virtual]

Destructor.

### 10.84.3 Member Function Documentation

10.84.3.1 virtual **Mesh\*** gazebo::common::MeshLoader::Load ( const std::string & \_filename ) [pure virtual]

Load a 3D mesh.

#### Parameters

in	_filename	the path to the mesh
----	-----------	----------------------

#### Returns

a pointer to the created mesh

Implemented in **gazebo::common::ColladaLoader** (p. 189), and **gazebo::common::STLLoader** (p. 734).

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

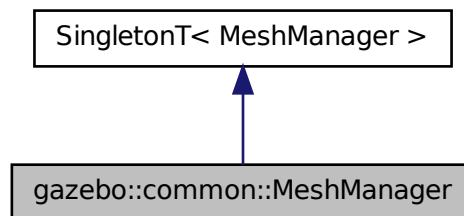
- **MeshLoader.hh**

## 10.85 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

### 10.85.1 Detailed Description

Maintains and manages all meshes.

### 10.85.2 Member Function Documentation

#### 10.85.2.1 void gazebo::common::MeshManager::AddMesh ( **Mesh** \* \_mesh )

Add a mesh to the manager.

This **MeshManager** (p. 464) takes ownership of the mesh and will destroy it. See `~MeshManager`.

#### Parameters

<code>in</code>	<code>the</code>	mesh to add.
-----------------	------------------	--------------

10.85.2.2 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

10.85.2.3 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

10.85.2.4 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

10.85.2.5 void gazebo::common::MeshManager::CreateCylinder ( const std::string & *\_name*, float *\_radius*, float *\_height*, int *\_rings*, int *\_segments* )

Create a cylinder mesh.

#### Parameters

in	<i>_name</i>	the name of the new mesh
in	<i>_radius</i>	the radius of the cylinder in the x y plane
in	<i>_height</i>	the height along z
in	<i>_rings</i>	the number of circles along the height
in	<i>_segments</i>	the number of segment per circle

10.85.2.6 void gazebo::common::MeshManager::CreatePlane ( const std::string & *\_name*, const math::Plane & *\_plane*, const math::Vector2d & *\_segments*, const math::Vector2d & *\_uvTile* )

Create mesh for a plane.

## Parameters

in	<i>_name</i>	
in	<i>_plane</i>	plane parameters
in	<i>_segments</i>	number of segments in x and y
in	<i>_uvTile</i>	the texture tile size in x and y

10.85.2.7 void gazebo::common::MeshManager::CreatePlane ( const std::string & *\_name*, const math::Vector3 & *\_normal*, double *\_d*, const math::Vector2d & *\_size*, const math::Vector2d & *\_segments*, const math::Vector2d & *\_uvTile* )

Create mesh for a plane.

## Parameters

in	<i>_name</i>	the name of the new mesh
in	<i>_normal</i>	the normal to the plane
in	<i>_d</i>	distance from the origin along normal
in	<i>_size</i>	the size of the plane in x and y
in	<i>_segments</i>	the number of segments in x and y
in	<i>_uvTile</i>	the texture tile size in x and y

10.85.2.8 void gazebo::common::MeshManager::CreateSphere ( const std::string & *\_name*, float *\_radius*, int *\_rings*, int *\_segments* )

Create a sphere mesh.

## Parameters

in	<i>_name</i>	the name of the mesh
in	<i>_radius</i>	radius of the sphere in meter
in	<i>_rings</i>	number of circles on th y axis
in	<i>_segments</i>	number of segment per circle

10.85.2.9 void gazebo::common::MeshManager::CreateTube ( const std::string & *\_name*, float *\_innerRadius*, float *\_outterRadius*, float *\_height*, int *\_rings*, int *\_segments* )

Create a tube mesh.

Generates rings inside and outside the cylinder Needs at least two rings and 3 segments

## Parameters

in	<i>_name</i>	the name of the new mesh
in	<i>_innerRadius</i>	the inner radius of the tube in the x y plane
in	<i>_outterRadius</i>	the outer radius of the tube in the x y plane
in	<i>_height</i>	the height along z
in	<i>_rings</i>	the number of circles along the height
in	<i>_segments</i>	the number of segment per circle

10.85.2.10 void gazebo::common::MeshManager::GenSphericalTexCoord ( const Mesh \* *\_mesh*, math::Vector3 *\_center* )

generate spherical texture coordinates

10.85.2.11 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

10.85.2.12 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

10.85.2.13 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
----	--------------	----------------------

10.85.2.14 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

10.85.2.15 const Mesh\* gazebo::common::MeshManager::Load ( const std::string & *\_filename* )

Load a mesh from a file.



## Parameters

in	<i>_filename</i>	the path to the mesh
----	------------------	----------------------

## Returns

a pointer to the created mesh

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

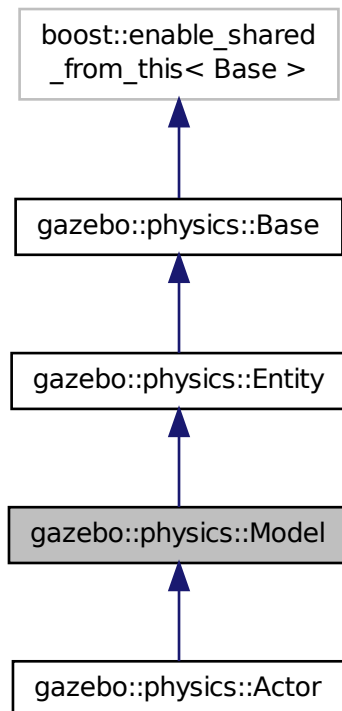
- **MeshManager.hh**

## 10.86 gazebo::physics::Model Class Reference

**A** (p. 107) 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  
*Get a link by id.*
  - `Link_V` **GetLinks** () const  
*Construct and return a vector of **Link** (p. 404)'s in this model Note this constructs the vector of **Link** (p. 404)'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. 404), 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. 371) Animation.*

  - void **SetJointPosition** (const std::string &\_jointName, double \_position)

*Set the positions of a **Joint** (p. 371) 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. 469) by specifying desired Pose of a **Link** (p. 404) within the **Model** (p. 469).*

  - void **SetLinkWorldPose** (const math::Pose &\_pose, const LinkPtr &\_link)

Set the Pose of the entire **Model** (p. 469) by specifying desired Pose of a **Link** (p. 404) within the **Model** (p. 469).

- 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. 473)
- std::vector< **math::Pose** > **attachedModelsOffset**  
used by **Model::AttachStaticModel** (p. 473)

### Additional Inherited Members

#### 10.86.1 Detailed Description

**A** (p. 107) model is a collection of links, joints, and plugins.

#### 10.86.2 Constructor & Destructor Documentation

10.86.2.1 gazebo::physics::Model::Model ( **BasePtr** \_parent ) [explicit]

Constructor.

Parameters

in	_parent	Parent object.
----	---------	----------------

10.86.2.2 virtual gazebo::physics::Model::~~Model ( ) [virtual]

Destructor.

#### 10.86.3 Member Function Documentation

10.86.3.1 void gazebo::physics::Model::AttachStaticModel ( ModelIPtr & *\_model*, math::Pose *\_offset* )

Attach a static model to this model.

This function takes as input a static **Model** (p. 469), which is a **Model** (p. 469) that has been marked as static (no physics simulation), and attaches it to this **Model** (p. 469) with a given offset.

This function is useful when you want to simulate a grasp of a static object, or move a static object around using a dynamic model.

If you are in doubt, do not use this function.

#### Parameters

in	<i>_model</i>	Pointer to the static model.
in	<i>_offset</i>	Offset, relative to this <b>Model</b> (p. 469), to place <i>_model</i> .

10.86.3.2 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. 473).

10.86.3.3 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.
----	-------------	--

10.86.3.4 virtual void gazebo::physics::Model::Fini ( ) [virtual]

Finalize the model.

Reimplemented from **gazebo::physics::Entity** (p. 277).

Reimplemented in **gazebo::physics::Actor** (p. 110).

10.86.3.5 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.

10.86.3.6 `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.277).

10.86.3.7 `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

10.86.3.8 `JointControllerPtr gazebo::physics::Model::GetJointController ( )` [inline]

Get a handle to the Controller for the joints in this model.

#### Returns

**A** (p. 107) handle to the Controller for the joints in this model.

10.86.3.9 `unsigned int gazebo::physics::Model::GetJointCount ( ) const`

Get the number of joints.

#### Returns

Get the number of joints.

10.86.3.10 `const Joint_V& gazebo::physics::Model::GetJoints ( ) const`

Get the joints.

#### Returns

Vector of joints.

10.86.3.11 `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.

10.86.3.12 `LinkPtr gazebo::physics::Model::GetLinkById ( unsigned int _id ) const`

Get a link by id.

#### Returns

Pointer to the link, NULL if the id is invalid.

10.86.3.13 `Link_V gazebo::physics::Model::GetLinks ( ) const`

Construct and return a vector of **Link** (p. 404)'s in this model Note this constructs the vector of **Link** (p. 404)'s on the fly, could be costly.

#### Returns

a vector of **Link** (p. 404)'s in this model

10.86.3.14 `unsigned int gazebo::physics::Model::GetPluginCount ( ) const`

Get the number of plugins this model has.

#### Returns

Number of plugins associated with this model.

10.86.3.15 `virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularAccel ( ) const` [virtual]

Get the angular acceleration of the entity.

#### Returns

**math::Vector3** (p. 821), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 279).

10.86.3.16 virtual **math::Vector3** gazebo::physics::Model::GetRelativeAngularVel ( ) const [virtual]

Get the angular velocity of the entity.

Returns

**math::Vector3** (p. 821), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 279).

10.86.3.17 virtual **math::Vector3** gazebo::physics::Model::GetRelativeLinearAccel ( ) const [virtual]

Get the linear acceleration of the entity.

Returns

**math::Vector3** (p. 821), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 279).

10.86.3.18 virtual **math::Vector3** gazebo::physics::Model::GetRelativeLinearVel ( ) const [virtual]

Get the linear velocity of the entity.

Returns

**math::Vector3** (p. 821), set to 0, 0, 0 if the model has no body.

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

10.86.3.19 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. 140).

Reimplemented in **gazebo::physics::Actor** (p. 110).

10.86.3.20 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.



10.86.3.21 virtual **math::Vector3** gazebo::physics::Model::GetWorldAngularAccel ( ) const [virtual]

Get the angular acceleration of the entity in the world frame.

Returns

**math::Vector3** (p. 821), set to 0, 0, 0 if the model has no body.

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

10.86.3.22 virtual **math::Vector3** gazebo::physics::Model::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the entity in the world frame.

Returns

**math::Vector3** (p. 821), set to 0, 0, 0 if the model has no body.

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

10.86.3.23 virtual **math::Vector3** gazebo::physics::Model::GetWorldLinearAccel ( ) const [virtual]

Get the linear acceleration of the entity in the world frame.

Returns

**math::Vector3** (p. 821), set to 0, 0, 0 if the model has no body.

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

10.86.3.24 virtual **math::Vector3** gazebo::physics::Model::GetWorldLinearVel ( ) const [virtual]

Get the linear velocity of the entity in the world frame.

Returns

**math::Vector3** (p. 821), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 281).

10.86.3.25 virtual void gazebo::physics::Model::Init ( ) [virtual]

Initialize the model.

Reimplemented from **gazebo::physics::Base** (p. 141).

Reimplemented in **gazebo::physics::Actor** (p. 111).

10.86.3.26 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. 281).

**10.86.3.27** void **gazebo::physics::Model::LoadJoints** ( )

Load all the joints.

**10.86.3.28** void **gazebo::physics::Model::LoadPlugins** ( )

Load all plugins.

Load all plugins specified in the SDF for the model.

**10.86.3.29** virtual void **gazebo::physics::Model::OnPoseChange** ( ) [protected],[virtual]

Callback when the pose of the model has been changed.

Implements **gazebo::physics::Entity** (p. 282).

**10.86.3.30** void **gazebo::physics::Model::ProcessMsg** ( const msgs::Model & *\_msg* )

Update parameters from a model message.

#### Parameters

in	<code>_msg</code>	Message to process.
----	-------------------	---------------------

**10.86.3.31** virtual void **gazebo::physics::Model::RemoveChild** ( EntityPtr *\_child* ) [virtual]

Remove a child.

#### Parameters

in	<code>_child</code>	Remove a child entity.
----	---------------------	------------------------

**10.86.3.32** void **gazebo::physics::Model::Reset** ( ) [virtual]

Reset the model.

Reimplemented from **gazebo::physics::Entity** (p. 282).

**10.86.3.33** void **gazebo::physics::Model::SetAngularAccel** ( const math::Vector3 & *\_vel* )

Set the angular acceleration of the model, and all its links.

## Parameters

in	<i>_vel</i>	The new angular acceleration
----	-------------	------------------------------

10.86.3.34 void gazebo::physics::Model::SetAngularVel ( const math::Vector3 & *\_vel* )

Set the angular velocity of the model, and all its links.

## Parameters

in	<i>_vel</i>	The new angular velocity.
----	-------------	---------------------------

10.86.3.35 void gazebo::physics::Model::SetAutoDisable ( bool *\_disable* )

Allow the model the auto disable.

This is ignored if the model has joints.

## Parameters

in	<i>_disable</i>	If true, the model is allowed to auto disable.
----	-----------------	--

10.86.3.36 void gazebo::physics::Model::SetCollideMode ( const std::string & *\_mode* )

This is not implemented in **Link** (p. 404), which means this function doesn't do anything.

Set the collide mode of the model.

## Parameters

in	<i>_mode</i>	The collision mode
----	--------------	--------------------

10.86.3.37 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.
----	-----------------	-------------------------------

10.86.3.38 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.
----	---------------	---

10.86.3.39 void gazebo::physics::Model::SetJointAnimation ( const std::map< std::string, common::NumericAnimationPtr > & \_anim, boost::function< void()> \_onComplete = NULL )

**Joint** (p. 371) Animation.

#### Parameters

in	<code>_anim</code>	Map of joint names to their position animation.
in	<code>_onComplete</code>	Callback function for when the animation completes.

10.86.3.40 void gazebo::physics::Model::SetJointPosition ( const std::string & \_jointName, double \_position )

Set the positions of a **Joint** (p. 371) by name.

#### See Also

**JointController::SetJointPosition** (p. 386)

#### Parameters

in	<code>_jointName</code>	Name of the joint to set.
in	<code>_position</code>	Position to set the joint to.

10.86.3.41 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. 386).

#### Parameters

in	<code>_jointPositions</code>	Map of joint names to their positions.
----	------------------------------	--

10.86.3.42 void gazebo::physics::Model::SetLaserRetro ( const float \_retro )

Set the laser retro reflectiveness of the model.

#### Parameters

in	<code>_retro</code>	Retro reflectance value.
----	---------------------	--------------------------

10.86.3.43 void gazebo::physics::Model::SetLinearAccel ( const math::Vector3 & \_vel )

Set the linear acceleration of the model, and all its links.

## Parameters

in	<code>_vel</code>	The new linear acceleration.
----	-------------------	------------------------------

10.86.3.44 `void gazebo::physics::Model::SetLinearVel ( const math::Vector3 & _vel )`

Set the linear velocity of the model, and all its links.

## Parameters

in	<code>_vel</code>	The new linear velocity.
----	-------------------	--------------------------

10.86.3.45 `void gazebo::physics::Model::SetLinkWorldPose ( const math::Pose & _pose, std::string _linkName )`

Set the Pose of the entire **Model** (p. 469) by specifying desired Pose of a **Link** (p. 404) within the **Model** (p. 469).

Doing so, keeps the configuration of the **Model** (p. 469) unchanged, i.e. all **Joint** (p. 371) angles are unchanged.

## Parameters

in	<code>_pose</code>	Pose to set the link to.
in	<code>_linkName</code>	Name of the link to set.

10.86.3.46 `void gazebo::physics::Model::SetLinkWorldPose ( const math::Pose & _pose, const LinkPtr & _link )`

Set the Pose of the entire **Model** (p. 469) by specifying desired Pose of a **Link** (p. 404) within the **Model** (p. 469).

Doing so, keeps the configuration of the **Model** (p. 469) unchanged, i.e. all **Joint** (p. 371) angles are unchanged.

## Parameters

in	<code>_pose</code>	Pose to set the link to.
in	<code>_link</code>	Pointer to the link to set.

10.86.3.47 `void gazebo::physics::Model::SetState ( const ModelState & _state )`

Set the current model state.

## Parameters

in	<code>_state</code>	<b>State</b> (p. 729) to set the model to.
----	---------------------	--

10.86.3.48 `virtual void gazebo::physics::Model::StopAnimation ( ) [virtual]`

Stop the current animations.

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

10.86.3.49 `void gazebo::physics::Model::Update ( ) [virtual]`

Update the model.

Reimplemented from `gazebo::physics::Base` (p. 143).

10.86.3.50 `virtual void gazebo::physics::Model::UpdateParameters ( sdf::ElementPtr _sdf ) [virtual]`

Update the parameters using new sdf values.

#### Parameters

in	<code>_sdf</code>	SDF values to update from.
----	-------------------	----------------------------

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

Reimplemented in `gazebo::physics::Actor` (p. 111).

## 10.86.4 Member Data Documentation

10.86.4.1 `std::vector<ModelPtr> gazebo::physics::Model::attachedModels [protected]`

used by `Model::AttachStaticModel` (p. 473)

10.86.4.2 `std::vector<math::Pose> gazebo::physics::Model::attachedModelsOffset [protected]`

used by `Model::AttachStaticModel` (p. 473)

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

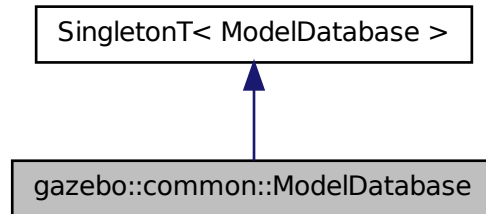
- `Model.hh`

## 10.87 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)  
*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

### 10.87.1 Detailed Description

Connects to model database, and has utility functions to find models.

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

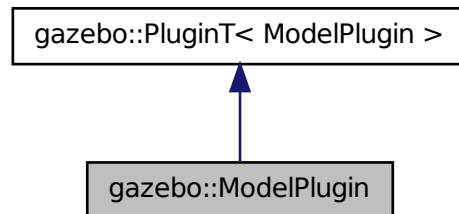
- **ModelDatabase.hh**

## 10.88 gazebo::ModelPlugin Class Reference

**A** (p. 107) plugin with access to **physics::Model** (p. 469).

```
#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

#### 10.88.1 Detailed Description

**A** (p. 107) plugin with access to **physics::Model** (p. 469).

See reference.

#### 10.88.2 Constructor & Destructor Documentation



10.88.2.1 gazebo::ModelPlugin::ModelPlugin ( ) [inline]

Constructor.

References gazebo::MODEL\_PLUGIN, and gazebo::PluginT< ModelPlugin >::type.

10.88.2.2 virtual gazebo::ModelPlugin::~~ModelPlugin ( ) [inline],[virtual]

Destructor.

### 10.88.3 Member Function Documentation

10.88.3.1 virtual void gazebo::ModelPlugin::Init ( ) [inline],[virtual]

Override this method for custom plugin initialization behavior.

10.88.3.2 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.

10.88.3.3 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:

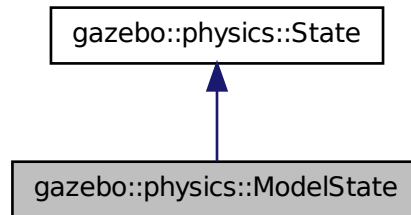
- **common/Plugin.hh**

## 10.89 gazebo::physics::ModelState Class Reference

Store state information of a **physics::Model** (p. 469) 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. 371) state.*
- **JointState GetJointState** (const std::string &\_jointName) const  
*Get a **Joint** (p. 371) state by **Joint** (p. 371) 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. 404) 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

### 10.89.1 Detailed Description

Store state information of a **physics::Model** (p. 469) object.

This class captures the entire state of a **Model** (p. 469) at one specific time during a simulation run.

**State** (p. 729) of a **Model** (p. 469) includes the state of all its child Links and Joints.

### 10.89.2 Constructor & Destructor Documentation

#### 10.89.2.1 gazebo::physics::ModelState::ModelState ( )

Default constructor.

#### 10.89.2.2 gazebo::physics::ModelState::ModelState ( const ModelPtr \_model ) [explicit]

Constructor.

Build a **ModelState** (p. 485) from an existing **Model** (p. 469).

#### Parameters

in	_model	Pointer to the model from which to gather state info.
----	--------	---

#### 10.89.2.3 gazebo::physics::ModelState::ModelState ( const sdf::ElementPtr \_sdf ) [explicit]

Constructor.

Build a **ModelState** (p. 485) from SDF data

#### Parameters

in	_sdf	SDF data to load a model state from.
----	------	--------------------------------------

10.89.2.4 virtual gazebo::physics::ModelState::~~ModelState ( ) [virtual]

Destructor.

### 10.89.3 Member Function Documentation

10.89.3.1 void gazebo::physics::ModelState::FillSDF ( sdf::ElementPtr \_sdf )

Populate a state SDF element with data from the object.

#### Parameters

out	_sdf	SDF element to populate.
-----	------	--------------------------

10.89.3.2 JointState gazebo::physics::ModelState::GetJointState ( unsigned int \_index ) const

Get a **Joint** (p. 371) state.

Return a **JointState** (p. 387) based on a index, where index is between 0...**ModelState::GetJointStateCount()** (p. 489).

#### Parameters

in	_index	Index of a <b>JointState</b> (p. 387).
----	--------	--

#### Returns

**State** (p. 729) of a **Joint** (p. 371).

#### Exceptions

<b>common::Exception</b> (p. 310)	When _index is out of range.
--------------------------------------	------------------------------

10.89.3.3 JointState gazebo::physics::ModelState::GetJointState ( const std::string & \_jointName ) const

Get a **Joint** (p. 371) state by **Joint** (p. 371) name.

Searches through all JointStates. Returns the **JointState** (p. 387) with the matching name, if any.

#### Parameters

in	_jointName	Name of the <b>JointState</b> (p. 387).
----	------------	---

## Returns

**State** (p. 729) of the **Joint** (p. 371).

## Exceptions

<b><i>common::Exception</i></b> (p. 310)	When <code>_jointName</code> is invalid.
---	--

## 10.89.3.4 unsigned int gazebo::physics::ModelState::GetJointStateCount ( ) const

Get the number of joint states.

Returns the number of JointStates recorded.

## Returns

Number of JointStates.

## 10.89.3.5 const std::vector&lt;JointState&gt;&amp; gazebo::physics::ModelState::GetJointStates ( ) const

Get the joint states.

## Returns

**A** (p. 107) vector of joint states.

10.89.3.6 LinkState gazebo::physics::ModelState::GetLinkState ( unsigned int *\_index* ) const

Get a link state.

Get a **Link** (p. 404) **State** (p. 729) based on an index, where index is in the range of 0...**ModelState::GetLinkStateCount** (p. 490)

## Parameters

<code>in</code>	<code>_index</code>	Index of the <b>LinkState</b> (p. 422)
-----------------	---------------------	--

## Returns

**State** (p. 729) of the **Link** (p. 404).

## Exceptions

<b><i>common::Exception</i></b> (p. 310)	When <code>_index</code> is out of range.
---	---

10.89.3.7 LinkState gazebo::physics::ModelState::GetLinkState ( const std::string & *\_linkName* ) const

Get a link state by **Link** (p. 404) name.

Searches through all LinkStates. Returns the **LinkState** (p. 422) with the matching name, if any.

#### Parameters

in	<code>_linkName</code>	Name of the <b>LinkState</b> (p. 422)
----	------------------------	---------------------------------------

#### Returns

**State** (p. 729) of the **Link** (p. 404).

#### Exceptions

<b><i>common::Exception</i></b> (p. 310)	When <code>_linkName</code> is invalid.
---	---

10.89.3.8 `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. 422) recorded.

10.89.3.9 `const std::vector<LinkState>& gazebo::physics::ModelState::GetLinkStates ( ) const`

Get the link states.

#### Returns

**A** (p. 107) vector of link states.

10.89.3.10 `const math::Pose& gazebo::physics::ModelState::GetPose ( ) const`

Get the stored model pose.

#### Returns

The **math::Pose** (p. 573) of the **Model** (p. 469).

10.89.3.11 `bool gazebo::physics::ModelState::HasJointState ( const std::string & _jointName ) const`

Return true if there is a joint with the specified name.

#### Parameters

in	<code>_jointName</code>	Name of the Jointtate.
----	-------------------------	------------------------

**Returns**

True if the joint exists in the model.

**10.89.3.12** `bool gazebo::physics::ModelState::HasLinkState ( const std::string & _linkName ) const`

Return true if there is a link with the specified name.

**Parameters**

<code>in</code>	<code>_linkName</code>	Name of the <b>LinkState</b> (p. 422).
-----------------	------------------------	--

**Returns**

True if the link exists in the model.

**10.89.3.13** `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.

**10.89.3.14** `virtual void gazebo::physics::ModelState::Load ( const sdf::ElementPtr _elem ) [virtual]`

Load state from SDF element.

Load **ModelState** (p. 485) 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. 732).

**10.89.3.15** `ModelState gazebo::physics::ModelState::operator+ ( const ModelState & _state ) const`

Addition operator.

**Parameters**

<code>in</code>	<code>_pt</code>	<b>A</b> (p. 107) state to subtract.
-----------------	------------------	--------------------------------------

**Returns**

The resulting state.

10.89.3.16 **ModelState** gazebo::physics::ModelState::operator- ( const ModelState & *\_state* ) const

Subtraction operator.

#### Parameters

<i>in</i>	<i>_pt</i>	<b>A</b> (p. 107) state to subtract.
-----------	------------	--------------------------------------

#### Returns

The resulting state.

10.89.3.17 **ModelState&** gazebo::physics::ModelState::operator= ( const ModelState & *\_state* )

Assignment operator.

#### Parameters

<i>in</i>	<i>_state</i>	<b>State</b> (p. 729) value
-----------	---------------	-----------------------------

#### Returns

*this*

## 10.89.4 Friends And Related Function Documentation

10.89.4.1 **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>	<b>Model</b> (p. 469) state to output.

#### Returns

The stream.

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

- **ModelState.hh**

## 10.90 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.*

### 10.90.1 Detailed Description

Generic description of a mouse event.

## 10.90.2 Member Enumeration Documentation

### 10.90.2.1 enum gazebo::common::MouseEvent::Buttons

Standard mouse buttons enumeration.

Enumerator:

***NO\_BUTTON***  
***LEFT***  
***MIDDLE***  
***RIGHT***

### 10.90.2.2 enum gazebo::common::MouseEvent::EventType

Mouse event types enumeration.

Enumerator:

***NO\_EVENT***  
***MOVE***  
***PRESS***  
***RELEASE***  
***SCROLL***

## 10.90.3 Constructor & Destructor Documentation

### 10.90.3.1 gazebo::common::MouseEvent::MouseEvent ( ) [inline]

Constructor.

## 10.90.4 Member Data Documentation

### 10.90.4.1 bool gazebo::common::MouseEvent::alt

Alt key press flag.

### 10.90.4.2 unsigned int gazebo::common::MouseEvent::button

The button which caused the event.

### 10.90.4.3 unsigned int gazebo::common::MouseEvent::buttons

State of the buttons when the event was generated.

### 10.90.4.4 bool gazebo::common::MouseEvent::control

Control key press flag.

10.90.4.5 `bool gazebo::common::MouseEvent::dragging`

Flag for mouse drag motion.

10.90.4.6 `float gazebo::common::MouseEvent::moveScale`

Scaling factor.

10.90.4.7 `math::Vector2i gazebo::common::MouseEvent::pos`

Mouse pointer position on the screen.

10.90.4.8 `math::Vector2i gazebo::common::MouseEvent::pressPos`

Position of button press.

10.90.4.9 `math::Vector2i gazebo::common::MouseEvent::prevPos`

Previous position.

10.90.4.10 `math::Vector2i gazebo::common::MouseEvent::scroll`

Scroll position.

10.90.4.11 `bool gazebo::common::MouseEvent::shift`

Shift key press flag.

10.90.4.12 `EventType gazebo::common::MouseEvent::type`

Event type.

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

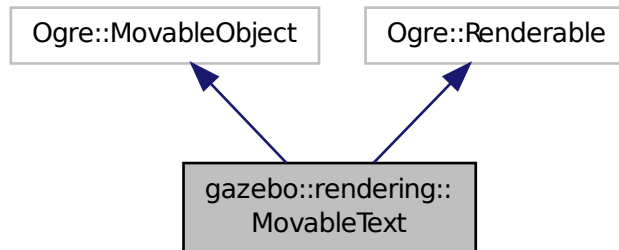
- **MouseEvent.hh**

## 10.91 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

### 10.91.1 Detailed Description

Movable text.

### 10.91.2 Member Enumeration Documentation

#### 10.91.2.1 enum gazebo::rendering::MovableText::HorizAlign

Horizontal alignment.

Enumerator:

**H\_LEFT** Left alignment.

**H\_CENTER** Center alignment.

### 10.91.2.2 enum gazebo::rendering::MovableText::VertAlign

vertical alignment

Enumerator:

**V\_BELOW** Align below.

**V\_ABOVE** Align above.

### 10.91.3 Constructor & Destructor Documentation

#### 10.91.3.1 gazebo::rendering::MovableText::MovableText ( )

Constructor.

#### 10.91.3.2 virtual gazebo::rendering::MovableText::~~MovableText ( ) [virtual]

Destructor.

### 10.91.4 Member Function Documentation

#### 10.91.4.1 void gazebo::rendering::MovableText::setupGeometry ( ) [protected]

#### 10.91.4.2 void gazebo::rendering::MovableText::updateColors ( ) [protected]

#### 10.91.4.3 math::Box gazebo::rendering::MovableText::GetAABB ( )

Get the axis aligned bounding box of the text.

Returns

The axis aligned bounding box.

#### 10.91.4.4 float gazebo::rendering::MovableText::GetBaseline ( ) const

Get the baseline height.

Returns

Baseline height

#### 10.91.4.5 float gazebo::rendering::MovableText::getBoundingRadius ( ) const [protected]

#### 10.91.4.6 float gazebo::rendering::MovableText::GetCharHeight ( ) const

Set the height of a characters return Height of the characters.

10.91.4.7 `const common::Color& gazebo::rendering::MovableText::GetColor ( ) const`

Get the text color.

Returns

Texture color.

10.91.4.8 `const std::string& gazebo::rendering::MovableText::GetFont ( ) const`

Get the font.

Returns

The font name

10.91.4.9 `const Ogre::LightList& gazebo::rendering::MovableText::getLights ( void ) const` [protected]

10.91.4.10 `const Ogre::MaterialPtr& gazebo::rendering::MovableText::getMaterial ( void ) const` [protected]

10.91.4.11 `void gazebo::rendering::MovableText::getRenderOperation ( Ogre::RenderOperation & op )` [protected]

10.91.4.12 `bool gazebo::rendering::MovableText::GetShowOnTop ( ) const`

True = text is displayed on top.

Returns

True if `MovableText::SetShownOnTop(true)` was called.

10.91.4.13 `float gazebo::rendering::MovableText::GetSpaceWidth ( ) const`

Get the width of a space.

Returns

Space width

10.91.4.14 `float gazebo::rendering::MovableText::getSquaredViewDepth ( const Ogre::Camera * cam ) const` [protected]

10.91.4.15 `const std::string& gazebo::rendering::MovableText::GetText ( ) const`

Get the displayed text.

Returns

The displayed text.

10.91.4.16 void gazebo::rendering::MovableText::getWorldTransforms ( Ogre::Matrix4 \* *xform* ) const [protected]

10.91.4.17 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

10.91.4.18 void gazebo::rendering::MovableText::SetBaseline ( float *\_height* )

Set the baseline height of the text.

#### Parameters

in	<i>_height</i>	Baseline height
----	----------------	-----------------

10.91.4.19 void gazebo::rendering::MovableText::SetCharHeight ( float *\_height* )

Set the height of a character.

#### Parameters

in	<i>_height</i>	Height of the characters.
----	----------------	---------------------------

10.91.4.20 void gazebo::rendering::MovableText::SetColor ( const common::Color & *\_color* )

Set the text color.

#### Parameters

in	<i>_color</i>	Text color.
----	---------------	-------------

10.91.4.21 void gazebo::rendering::MovableText::SetFontName ( const std::string & *\_font* )

Set the font.

#### Parameters

in	<i>_font</i>	Name of the font
----	--------------	------------------



10.91.4.22 void gazebo::rendering::MovableText::SetShowOnTop ( bool *\_show* )

True = text always is displayed on top.

#### Parameters

in	<i>_show</i>	Set to true to render the text on top of all other drawables.
----	--------------	---

10.91.4.23 void gazebo::rendering::MovableText::SetSpaceWidth ( float *\_width* )

Set the width of a space.

#### Parameters

in	<i>_width</i>	space width
----	---------------	-------------

10.91.4.24 void gazebo::rendering::MovableText::SetText ( const std::string & *\_text* )

Set the text to display.

#### Parameters

in	<i>_text</i>	The text to display.
----	--------------	----------------------

10.91.4.25 void gazebo::rendering::MovableText::SetTextAlignment ( const HorizAlign & *\_hAlign*, const VertAlign & *\_vAlign* )

Set the alignment of the text.

#### Parameters

in	<i>_hAlign</i>	Horizontal alignment
in	<i>_vAlign</i>	Vertical alignment

10.91.4.26 void gazebo::rendering::MovableText::Update ( )

Update the text.

10.91.4.27 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**

## 10.92 gazebo::msgs::MsgFactory Class Reference

**A** (p. 107) 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.*

### 10.92.1 Detailed Description

**A** (p. 107) factory that generates protobuf message based on a string type.

### 10.92.2 Member Function Documentation

10.92.2.1 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.
-----	---------------------	---

10.92.2.2 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.

10.92.2.3 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:

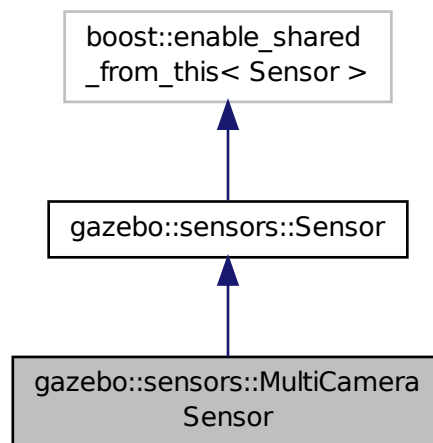
- [MsgFactory.hh](#)

## 10.93 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. 157).*
- 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 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

#### 10.93.1 Detailed Description

Multiple camera sensor.

This sensor type can create one or more synchronized cameras.

#### 10.93.2 Constructor & Destructor Documentation

##### 10.93.2.1 gazebo::sensors::MultiCameraSensor::MultiCameraSensor ( )

Constructor.

##### 10.93.2.2 virtual gazebo::sensors::MultiCameraSensor::~MultiCameraSensor ( ) [virtual]

Destructor.

#### 10.93.3 Member Function Documentation

##### 10.93.3.1 virtual void gazebo::sensors::MultiCameraSensor::Fini ( ) [protected],[virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 676).

##### 10.93.3.2 rendering::CameraPtr gazebo::sensors::MultiCameraSensor::GetCamera ( unsigned int \_index ) const

Returns a pointer to a **rendering::Camera** (p. 157).

#### Parameters

<code>in</code>	<code>_index</code>	Index of the camera to get
-----------------	---------------------	----------------------------

**Returns**

The Pointer to the camera sensor.

**See Also**

**MultiCameraSensor::GetCameraCount** (p. 505)

10.93.3.3 unsigned int gazebo::sensors::MultiCameraSensor::GetCameraCount ( ) const

Get the number of cameras.

**Returns**

The number of cameras.

10.93.3.4 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. 505)

10.93.3.5 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. 505)

10.93.3.6 `unsigned int gazebo::sensors::MultiCameraSensor::GetImageWidth ( unsigned int _index ) const`

Gets the width of the image in pixels.

#### Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the camera
-----------------	----------------------------	---------------------

#### Returns

The image width in pixels.

#### See Also

**MultiCameraSensor::GetCameraCount** (p. 505)

10.93.3.7 `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. 677).

10.93.3.8 `virtual void gazebo::sensors::MultiCameraSensor::Init ( )` [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 678).

10.93.3.9 `virtual void gazebo::sensors::MultiCameraSensor::Load ( const std::string & _worldName )` [virtual]

Load the sensor with default parameters.

#### Parameters

<code>in</code>	<code><i>_worldName</i></code>	Name of world to load from.
-----------------	--------------------------------	-----------------------------

Reimplemented from **gazebo::sensors::Sensor** (p. 678).

10.93.3.10 `bool gazebo::sensors::MultiCameraSensor::SaveFrame ( const std::vector< std::string > & _filenames )`

Saves the camera image(s) to the disk.

#### Parameters

<code>in</code>	<code><i>_filenames</i></code>	The name of the files for each camera.
-----------------	--------------------------------	--

**Returns**

True if successful, false if unsuccessful.

**See Also**

**MultiCameraSensor::GetCameraCount** (p. 505)

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

Reimplemented from **gazebo::sensors::Sensor** (p. 679).

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

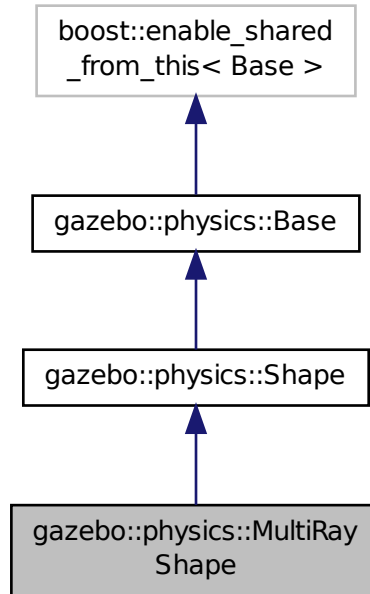
- **MultiCameraSensor.hh**

## 10.94 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

### 10.94.1 Detailed Description

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

### 10.94.2 Constructor & Destructor Documentation

10.94.2.1 `gazebo::physics::MultiRayShape::MultiRayShape ( CollisionPtr _parent ) [explicit]`

Constructor.

#### Parameters

in	<code>_parent</code>	Parent collision shape.
----	----------------------	-------------------------

10.94.2.2 `virtual gazebo::physics::MultiRayShape::~~MultiRayShape ( ) [virtual]`

Destructor.

### 10.94.3 Member Function Documentation

10.94.3.1 `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.

10.94.3.2 `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.

10.94.3.3 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.

10.94.3.4 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. 695).

10.94.3.5 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.

10.94.3.6 math::Angle gazebo::physics::MultiRayShape::GetMaxAngle ( ) const

Get the maximum angle.

**Returns**

Maximum angle of ray scan.

10.94.3.7 double gazebo::physics::MultiRayShape::GetMaxRange ( ) const

Get the maximum range.

**Returns**

Maximum range of all the rays.

**10.94.3.8 math::Angle gazebo::physics::MultiRayShape::GetMinAngle ( ) const**

Get the minimum angle.

**Returns**

Minimum angle of ray scan.

**10.94.3.9 double gazebo::physics::MultiRayShape::GetMinRange ( ) const**

Get the minimum range.

**Returns**

Minimum range of all the rays.

**10.94.3.10 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.

**10.94.3.11 double gazebo::physics::MultiRayShape::GetResRange ( ) const**

Get the range resolution.

**Returns**

Range resolution of all the rays.

**10.94.3.12 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.

**10.94.3.13 int gazebo::physics::MultiRayShape::GetSampleCount ( ) const**

Get the horizontal sample count.

**Returns**

Horizontal sample count.

**10.94.3.14 double gazebo::physics::MultiRayShape::GetScanResolution ( ) const**

Get the horizontal resolution.

**Returns**

Horizontal resolution.

**10.94.3.15 math::Angle gazebo::physics::MultiRayShape::GetVerticalMaxAngle ( ) const**

Get the vertical max angle.

**Returns**

Vertical max angle.

**10.94.3.16 math::Angle gazebo::physics::MultiRayShape::GetVerticalMinAngle ( ) const**

Get the vertical min angle.

**Returns**

Vertical min angle.

**10.94.3.17 int gazebo::physics::MultiRayShape::GetVerticalSampleCount ( ) const**

Get the vertical sample count.

**Returns**

Vertical sample count.

**10.94.3.18 double gazebo::physics::MultiRayShape::GetVerticalScanResolution ( ) const**

Get the vertical range resolution.

**Returns**

Vertical range resolution.

10.94.3.19 `virtual void gazebo::physics::MultiRayShape::Init ( ) [virtual]`

Init the shape.

Implements `gazebo::physics::Shape` (p. 695).

10.94.3.20 `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. 695).

10.94.3.21 `void gazebo::physics::MultiRayShape::Update ( ) [virtual]`

Update the ray collisions.

Reimplemented from `gazebo::physics::Base` (p. 143).

10.94.3.22 `virtual void gazebo::physics::MultiRayShape::UpdateRays ( ) [protected],[pure virtual]`

Physics engine specific method for updating the rays.

## 10.94.4 Member Data Documentation

10.94.4.1 `sdf::ElementPtr gazebo::physics::MultiRayShape::horzElem [protected]`

Horizontal SDF element pointer.

10.94.4.2 `event::EventT<void()> gazebo::physics::MultiRayShape::newLaserScans [protected]`

New laser scans event.

Referenced by `ConnectNewLaserScans()`, and `DisconnectNewLaserScans()`.

10.94.4.3 `math::Pose gazebo::physics::MultiRayShape::offset [protected]`

Pose offset of all the rays.

10.94.4.4 `sdf::ElementPtr gazebo::physics::MultiRayShape::rangeElem [protected]`

Range SDF element pointer.

10.94.4.5 `sdf::ElementPtr gazebo::physics::MultiRayShape::rayElem` [protected]

Ray SDF element pointer.

10.94.4.6 `std::vector<RayShapePtr> gazebo::physics::MultiRayShape::rays` [protected]

Ray data.

10.94.4.7 `sdf::ElementPtr gazebo::physics::MultiRayShape::scanElem` [protected]

Scan SDF element pointer.

10.94.4.8 `sdf::ElementPtr gazebo::physics::MultiRayShape::vertElem` [protected]

Vertical SDF element pointer.

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

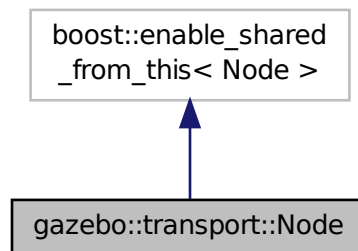
- **MultiRayShape.hh**

## 10.95 gazebo::transport::Node Class Reference

**A** (p. 107) 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)  
*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.*

### 10.95.1 Detailed Description

**A** (p. 107) node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.



## 10.95.2 Constructor & Destructor Documentation

### 10.95.2.1 gazebo::transport::Node::Node ( )

Constructor.

### 10.95.2.2 virtual gazebo::transport::Node::~~Node ( ) [virtual]

Destructor.

## 10.95.3 Member Function Documentation

### 10.95.3.1 template<typename M > transport::PublisherPtr gazebo::transport::Node::Advertise ( const std::string & *\_topic*, unsigned int *\_queueLimit* = 1000 ) [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

#### Returns

Pointer to new publisher object

References DecodeTopicName(), and SingletonT< T >::Instance().

### 10.95.3.2 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().

### 10.95.3.3 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

10.95.3.4 `void gazebo::transport::Node::Fini ( )`

Finalize the node.

10.95.3.5 `unsigned int gazebo::transport::Node::GetId ( ) const`

Get the unique ID of the node.

**Returns**

The unique ID of the node

Referenced by `Subscribe()`.

10.95.3.6 `std::string gazebo::transport::Node::GetMsgType ( const std::string & _topic ) const`

Get the message type for a topic.

**Parameters**

<code>in</code>	<code>_topic</code>	The topic
-----------------	---------------------	-----------

**Returns**

The message type

10.95.3.7 `std::string gazebo::transport::Node::GetTopicNamespace ( ) const`

Get the topic namespace for this node.

**Returns**

The namespace

10.95.3.8 `bool gazebo::transport::Node::HandleData ( const std::string & _topic, const std::string & _msg )`

Handle incoming data.

**Parameters**

<code>in</code>	<code>_topic</code>	Topic for which the data was received
<code>in</code>	<code>_msg</code>	The message that was received

**Returns**

true if the message was handled successfully, false otherwise

**10.95.3.9 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.

**10.95.3.10 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 <b>Master</b> (p. 433)
-----------	---------------	---

**10.95.3.11 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.

**10.95.3.12 void gazebo::transport::Node::ProcessIncoming ( )**

Process incoming messages.

**10.95.3.13 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

**10.95.3.14 void gazebo::transport::Node::RemoveCallback ( const std::string & *\_topic*, unsigned int *\_id* )**

10.95.3.15 `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. 746) object

References `DecodeTopicName()`, `GetId()`, `SingletonT< T >::Instance()`, and `gazebo::transport::Subscriber::Set-CallbackId()`.

10.95.3.16 `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. 746) object

References `DecodeTopicName()`, `GetId()`, `SingletonT< T >::Instance()`, and `gazebo::transport::Subscriber::Set-CallbackId()`.

10.95.3.17 `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. 746) object

References DecodeTopicName(), GetId(), gazebo::transport::SubscribeOptions::Init(), SingletonT< T >::Instance(), and gazebo::transport::Subscriber::SetCallbackId().

**10.95.3.18 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. 746) 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**

## 10.96 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*

### 10.96.1 Detailed Description

Node animation.

### 10.96.2 Constructor & Destructor Documentation

#### 10.96.2.1 gazebo::common::NodeAnimation::NodeAnimation ( const std::string & *\_name* )

constructor

#### Parameters

in	<i>_name</i>	the name of the node
----	--------------	----------------------

#### 10.96.2.2 gazebo::common::NodeAnimation::~~NodeAnimation ( )

Destructor. It empties the key frames list.

### 10.96.3 Member Function Documentation

10.96.3.1 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

10.96.3.2 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

10.96.3.3 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)

10.96.3.4 unsigned int gazebo::common::NodeAnimation::GetFrameCount ( ) const

Returns the number of key frames.

#### Returns

the count

10.96.3.5 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

10.96.3.6 `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><i>in</i></code>	<code><i>_i</i></code>	the index
------------------------	------------------------	-----------

#### Returns

a pair that contains the time and transformation. **Time** (p. 760) is -1 if the index is out of bounds

10.96.3.7 `double gazebo::common::NodeAnimation::GetLength ( ) const`

Returns the duration of the animations.

#### Returns

the time of the last animation

10.96.3.8 `std::string gazebo::common::NodeAnimation::GetName ( ) const`

Returns the name.

#### Returns

the name

10.96.3.9 `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><i>in</i></code>	<code><i>_x</i></code>	the value along x. You must ensure that <i>_x</i> is within a valid range.
------------------------	------------------------	--

10.96.3.10 `void gazebo::common::NodeAnimation::Scale ( const double _scale )`

Scales each transformation in the key frames.

This only affects the translational values.

#### Parameters

<code><i>in</i></code>	<code><i>_scale</i></code>	the scaling factor
------------------------	----------------------------	--------------------



10.96.3.11 void gazebo::common::NodeAnimation::SetName ( const std::string & *\_name* )

Changes the name of the animation.

#### Parameters

<i>in</i>	<i>the</i>	new name
-----------	------------	----------

### 10.96.4 Member Data Documentation

10.96.4.1 std::map<double, math::Matrix4> gazebo::common::NodeAnimation::keyFrames [protected]

the dictionary of key frames, indexed by time

10.96.4.2 double gazebo::common::NodeAnimation::length [protected]

the duration of the animations (time of last key frame)

10.96.4.3 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**

## 10.97 gazebo::common::NodeAssignment Struct Reference

Vertex to node weighted assignment for skeleton animation visualization.

```
#include <Mesh.hh>
```

### Public Attributes

- unsigned int **nodeIndex**  
*node (or bone) index*
- unsigned int **vertexIndex**  
*index of the vertex*
- float **weight**  
*the weight (between 0 and 1)*

### 10.97.1 Detailed Description

Vertex to node weighted assignment for skeleton animation visualization.

## 10.97.2 Member Data Documentation

### 10.97.2.1 unsigned int gazebo::common::NodeAssignment::nodeIndex

node (or bone) index

### 10.97.2.2 unsigned int gazebo::common::NodeAssignment::vertexIndex

index of the vertex

### 10.97.2.3 float gazebo::common::NodeAssignment::weight

the weight (between 0 and 1)

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

- **Mesh.hh**

## 10.98 gazebo::common::NodeTransform Class Reference

**NodeTransform** (p. 526) **Skeleton.hh** (p. 1053) common/common.hh

```
#include <Skeleton.hh>
```

### Public Types

- enum **TransformType** { **TRANSLATE**, **ROTATE**, **SCALE**, **MATRIX** }  
*Enumeration of the transform types.*

### Public Member Functions

- **NodeTransform** (**TransformType** \_type=**MATRIX**)  
*Constructor.*
- **NodeTransform** (**math::Matrix4** \_mat, std::string \_sid="\_default\_", **TransformType** \_type=**MATRIX**)  
*Constructor.*
- **~NodeTransform** ()  
*Destructor. It does nothing.*
- **math::Matrix4** **Get** ()  
*Returns the transformation matrix.*
- std::string **GetSID** ()  
*Returns thr SID.*
- **TransformType** **GetType** ()  
*Returns the transformation type.*
- **math::Matrix4** **operator**() ()  
*Matrix cast operator.*
- **math::Matrix4** **operator\*** (**NodeTransform** \_t)  
*Node transform multiplication operator.*

- **math::Matrix4 operator\*** (**math::Matrix4** \_m)  
*Matrix multiplication operator.*
- void **PrintSource** ()  
*Prints the transform matrix to std::err stream.*
- void **RecalculateMatrix** ()  
*Sets the transform matrix from the source according to the type.*
- void **Set** (**math::Matrix4** \_mat)  
*Assign a transformation.*
- void **SetComponent** (unsigned int \_idx, double \_value)  
*Set a transformation matrix component value.*
- void **SetSID** (std::string \_sid)  
*Set the SID.*
- void **SetSourceValues** (**math::Matrix4** \_mat)  
*Set source data values \_param[in] \_mat the values.*
- void **SetSourceValues** (**math::Vector3** \_vec)  
*Set source data values.*
- void **SetSourceValues** (**math::Vector3** \_axis, double \_angle)  
*Sets source matrix values from roation.*
- void **SetType** (**TransformType** \_type)  
*Set transform type.*

## Protected Attributes

- std::string **sid**  
*the sid*
- std::vector< double > **source**  
*source data values (can be a matrix, a position or rotation)*
- **math::Matrix4** **transform**  
*transform*
- **TransformType** **type**  
*transform type*

### 10.98.1 Detailed Description

**NodeTransform** (p. 526) **Skeleton.hh** (p. 1053) common/common.hh

**A** (p. 107) transformation node

### 10.98.2 Member Enumeration Documentation

#### 10.98.2.1 enum gazebo::common::NodeTransform::TransformType

Enumeration of the transform types.

Enumerator:

**TRANSLATE**  
**ROTATE**  
**SCALE**  
**MATRIX**

### 10.98.3 Constructor & Destructor Documentation

#### 10.98.3.1 gazebo::common::NodeTransform::NodeTransform ( TransformType *\_type* = MATRIX )

Constructor.

##### Parameters

in	<i>_type</i>	the type of transform
----	--------------	-----------------------

#### 10.98.3.2 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

#### 10.98.3.3 gazebo::common::NodeTransform::~~NodeTransform ( )

Destructor. It does nothing.

### 10.98.4 Member Function Documentation

#### 10.98.4.1 math::Matrix4 gazebo::common::NodeTransform::Get ( )

Returns the transformation matrix.

##### Returns

the matrix

#### 10.98.4.2 std::string gazebo::common::NodeTransform::GetSID ( )

Returns thr SID.

##### Returns

the SID

#### 10.98.4.3 TransformType gazebo::common::NodeTransform::GetType ( )

Returns the transformation type.

##### Returns

the type

10.98.4.4 `math::Matrix4 gazebo::common::NodeTransform::operator() ( )`

Matrix cast operator.

## Returns

the transform

10.98.4.5 `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

10.98.4.6 `math::Matrix4 gazebo::common::NodeTransform::operator* ( math::Matrix4 _m )`

Matrix multiplication operator.

## Parameters

in	_m	a matrix
----	----	----------

## Returns

transform matrix multiplied by \_m

10.98.4.7 `void gazebo::common::NodeTransform::PrintSource ( )`

Prints the transform matrix to std::err stream.

10.98.4.8 `void gazebo::common::NodeTransform::RecalculateMatrix ( )`

Sets the transform matrix from the source according to the type.

10.98.4.9 `void gazebo::common::NodeTransform::Set ( math::Matrix4 _mat )`

Assign a transformation.

## Parameters

in	_mat	the transform
----	------	---------------

10.98.4.10 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

10.98.4.11 void gazebo::common::NodeTransform::SetSID ( std::string *\_sid* )

Set the SID.

#### Parameters

in	<i>_sid</i>	the sid
----	-------------	---------

10.98.4.12 void gazebo::common::NodeTransform::SetSourceValues ( math::Matrix4 *\_mat* )

Set source data values *\_param[in]* *\_mat* the values.

10.98.4.13 void gazebo::common::NodeTransform::SetSourceValues ( math::Vector3 *\_vec* )

Set source data values.

10.98.4.14 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

10.98.4.15 void gazebo::common::NodeTransform::SetType ( TransformType *\_type* )

Set transform type.

#### Parameters

in	<i>_type</i>	the type
----	--------------	----------

## 10.98.5 Member Data Documentation

10.98.5.1 std::string gazebo::common::NodeTransform::sid [protected]

the sid

10.98.5.2 `std::vector<double>` gazebo::common::NodeTransform::source [protected]

source data values (can be a matrix, a position or rotation)

10.98.5.3 `math::Matrix4` gazebo::common::NodeTransform::transform [protected]

transform

10.98.5.4 `TransformType` gazebo::common::NodeTransform::type [protected]

transform type

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

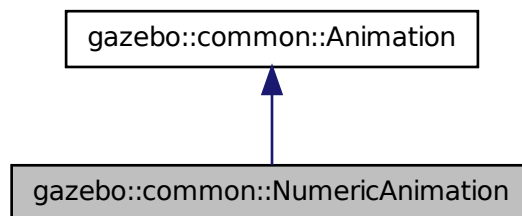
- **Skeleton.hh**

## 10.99 gazebo::common::NumericAnimation Class Reference

**A** (p. 107) 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

### 10.99.1 Detailed Description

**A** (p. 107) numeric animation.

### 10.99.2 Constructor & Destructor Documentation

10.99.2.1 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

10.99.2.2 virtual gazebo::common::NumericAnimation::~~NumericAnimation ( ) [virtual]

Destructor.

### 10.99.3 Member Function Documentation

10.99.3.1 NumericKeyFrame\* gazebo::common::NumericAnimation::CreateKeyFrame ( double *\_time* )

Create a numeric keyframe at the given time.

#### Parameters

in	<i>_time</i>	<b>Time</b> (p. 760) at which to create the keyframe
----	--------------	--

#### Returns

Pointer to the new keyframe

10.99.3.2 void gazebo::common::NumericAnimation::GetInterpolatedKeyFrame ( NumericKeyFrame & *\_kf* ) const

Get a keyframe using the animation's current time.

#### Parameters

out	<i>_kf</i>	<b>NumericKeyFrame</b> (p. 533) reference to hold the interpolated result
-----	------------	---

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

- **Animation.hh**

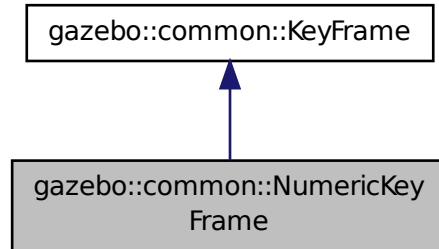


## 10.100 gazebo::common::NumericKeyFrame Class Reference

**A** (p. 107) keyframe for a **NumericAnimation** (p. 531).

```
#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*

#### 10.100.1 Detailed Description

**A** (p. 107) keyframe for a **NumericAnimation** (p. 531).

#### 10.100.2 Constructor & Destructor Documentation

##### 10.100.2.1 gazebo::common::NumericKeyFrame::NumericKeyFrame ( double *\_time* )

Constructor.

## Parameters

in	<code>_time</code>	<b>Time</b> (p. 760) of the keyframe
----	--------------------	--------------------------------------

10.100.2.2 `virtual gazebo::common::NumericKeyFrame::~~NumericKeyFrame ( )` [virtual]

Destructor.

### 10.100.3 Member Function Documentation

10.100.3.1 `const double& gazebo::common::NumericKeyFrame::GetValue ( ) const`

Get the value of the keyframe.

## Returns

the value of the keyframe

10.100.3.2 `void gazebo::common::NumericKeyFrame::SetValue ( const double & _value )`

Set the value of the keyframe.

## Parameters

in	<code>_value</code>	The new value
----	---------------------	---------------

### 10.100.4 Member Data Documentation

10.100.4.1 `double gazebo::common::NumericKeyFrame::value` [protected]

numeric value

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

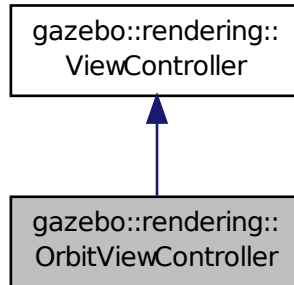
- **KeyFrame.hh**

## 10.101 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

### 10.101.1 Detailed Description

Orbit view controller.

### 10.101.2 Constructor & Destructor Documentation

#### 10.101.2.1 gazebo::rendering::OrbitViewController::OrbitViewController ( `UserCameraPtr _camera` )

Constructor.

##### Parameters

in	<code>_camera</code>	Pointer to the camera to control.
----	----------------------	-----------------------------------

#### 10.101.2.2 virtual gazebo::rendering::OrbitViewController::~~OrbitViewController ( ) [virtual]

Destructor.

### 10.101.3 Member Function Documentation

#### 10.101.3.1 math::Vector3 gazebo::rendering::OrbitViewController::GetFocalPoint ( ) const

Get the focal point.

##### Returns

The focal point

#### 10.101.3.2 static std::string gazebo::rendering::OrbitViewController::GetTypeString ( ) [static]

Get the type name of this view controller.

##### Returns

The view controller name: "orbit".

#### 10.101.3.3 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. 848).

10.101.3.4 void gazebo::rendering::OrbitViewController::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. 849).

10.101.3.5 virtual void gazebo::rendering::OrbitViewController::HandleMouseEvent ( const common::MouseEvent & *\_event* ) [virtual]

Handle a mouse event.

#### Parameters

in	<i>_event</i>	The mouse event.
----	---------------	------------------

Implements **gazebo::rendering::ViewController** (p. 849).

10.101.3.6 virtual void gazebo::rendering::OrbitViewController::Init ( ) [virtual]

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 849).

10.101.3.7 virtual void gazebo::rendering::OrbitViewController::Init ( const math::Vector3 & *\_focalPoint* ) [virtual]

Initialize the controller with a focal point.

#### Parameters

in	<i>_focalPoint</i>	Point to look at.
----	--------------------	-------------------

Reimplemented from **gazebo::rendering::ViewController** (p. 849).

10.101.3.8 void gazebo::rendering::OrbitViewController::SetDistance ( float *\_d* )

Set the distance to the focal point.

#### Parameters

in	<i>_d</i>	The distance from the focal point.
----	-----------	------------------------------------

10.101.3.9 void gazebo::rendering::OrbitViewController::SetFocalPoint ( const math::Vector3 & *\_fp* )

Set the focal point.

## Parameters

in	<code>_fp</code>	The focal point
----	------------------	-----------------

10.101.3.10 virtual void gazebo::rendering::OrbitViewController::Update ( ) [virtual]

Update.

Implements **gazebo::rendering::ViewController** (p. 850).

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

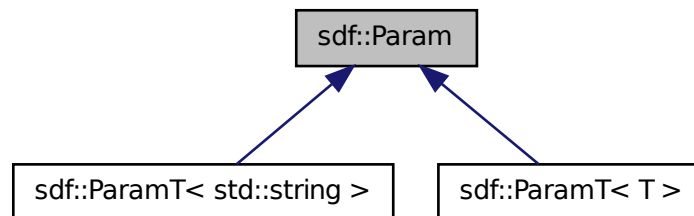
- **OrbitViewController.hh**

## 10.102 sdf::Param Class Reference

**A** (p. 107) 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**

### 10.102.1 Detailed Description

**A** (p. 107) parameter class.

### 10.102.2 Constructor & Destructor Documentation

10.102.2.1 sdf::Param::Param ( Param \* \_newParam )

Constructor.

10.102.2.2 virtual sdf::Param::~~Param ( ) [virtual]

Destructor.

### 10.102.3 Member Function Documentation

10.102.3.1 virtual Param\* sdf::Param::Clone ( ) const [pure virtual]

Implemented in **sdf::ParamT< T >** (p. 546), and **sdf::ParamT< std::string >** (p. 546).

10.102.3.2 bool sdf::Param::Get ( bool & \_value )

10.102.3.3 bool sdf::Param::Get ( int & \_value )

10.102.3.4 bool sdf::Param::Get ( unsigned int & \_value )

10.102.3.5 bool sdf::Param::Get ( float & \_value )

10.102.3.6 bool sdf::Param::Get ( double & \_value )

10.102.3.7 bool sdf::Param::Get ( char & \_value )



10.102.3.8 `bool sdf::Param::Get ( std::string & _value )`

10.102.3.9 `bool sdf::Param::Get ( gazebo::math::Vector3 & _value )`

10.102.3.10 `bool sdf::Param::Get ( gazebo::math::Vector2i & _value )`

10.102.3.11 `bool sdf::Param::Get ( gazebo::math::Vector2d & _value )`

10.102.3.12 `bool sdf::Param::Get ( gazebo::math::Quaternion & _value )`

10.102.3.13 `bool sdf::Param::Get ( gazebo::math::Pose & _value )`

10.102.3.14 `bool sdf::Param::Get ( gazebo::common::Color & _value )`

10.102.3.15 `bool sdf::Param::Get ( gazebo::common::Time & _value )`

10.102.3.16 `virtual std::string sdf::Param::GetAsString ( ) const [inline],[virtual]`

Get the type.

Reimplemented in `sdf::ParamT< T >` (p. 546), and `sdf::ParamT< std::string >` (p. 546).

10.102.3.17 `virtual std::string sdf::Param::GetDefaultAsString ( ) const [inline],[virtual]`

Reimplemented in `sdf::ParamT< T >` (p. 546), and `sdf::ParamT< std::string >` (p. 546).

10.102.3.18 `std::string sdf::Param::GetDescription ( ) const`

Get the description of the parameter.

10.102.3.19 `const std::string& sdf::Param::GetKey ( ) const [inline]`

References key.

Referenced by `sdf::ParamT< std::string >::Clone()`, and `sdf::ParamT< std::string >::Set()`.

10.102.3.20 `bool sdf::Param::GetRequired ( ) const [inline]`

References required.

10.102.3.21 `bool sdf::Param::GetSet ( ) const [inline]`

Return true if the parameter has been set.

10.102.3.22 `std::string sdf::Param::GetTypeNames ( ) const`

10.102.3.23 `bool sdf::Param::IsBool ( ) const`

10.102.3.24 `bool sdf::Param::IsChar ( ) const`

- 10.102.3.25 `bool sdf::Param::IsColor ( ) const`
- 10.102.3.26 `bool sdf::Param::IsDouble ( ) const`
- 10.102.3.27 `bool sdf::Param::IsFloat ( ) const`
- 10.102.3.28 `bool sdf::Param::IsInt ( ) const`
- 10.102.3.29 `bool sdf::Param::IsPose ( ) const`
- 10.102.3.30 `bool sdf::Param::IsQuaternion ( ) const`
- 10.102.3.31 `bool sdf::Param::IsStr ( ) const`
- 10.102.3.32 `bool sdf::Param::IsTime ( ) const`
- 10.102.3.33 `bool sdf::Param::IsUInt ( ) const`
- 10.102.3.34 `bool sdf::Param::IsVector2d ( ) const`
- 10.102.3.35 `bool sdf::Param::IsVector2i ( ) const`
- 10.102.3.36 `bool sdf::Param::IsVector3 ( ) const`
- 10.102.3.37 `virtual void sdf::Param::Reset ( ) [pure virtual]`

Reset the parameter.

Implemented in `sdf::ParamT< T >` (p. 546), and `sdf::ParamT< std::string >` (p. 546).

- 10.102.3.38 `bool sdf::Param::Set ( const bool & _value )`

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

- 10.102.3.39 `bool sdf::Param::Set ( const int & _value )`
- 10.102.3.40 `bool sdf::Param::Set ( const unsigned int & _value )`
- 10.102.3.41 `bool sdf::Param::Set ( const float & _value )`
- 10.102.3.42 `bool sdf::Param::Set ( const double & _value )`
- 10.102.3.43 `bool sdf::Param::Set ( const char & _value )`
- 10.102.3.44 `bool sdf::Param::Set ( const std::string & _value )`
- 10.102.3.45 `bool sdf::Param::Set ( const char * _value )`
- 10.102.3.46 `bool sdf::Param::Set ( const gazebo::math::Vector3 & _value )`
- 10.102.3.47 `bool sdf::Param::Set ( const gazebo::math::Vector2i & _value )`

10.102.3.48 `bool sdf::Param::Set ( const gazebo::math::Vector2d & _value )`

10.102.3.49 `bool sdf::Param::Set ( const gazebo::math::Quaternion & _value )`

10.102.3.50 `bool sdf::Param::Set ( const gazebo::math::Pose & _value )`

10.102.3.51 `bool sdf::Param::Set ( const gazebo::common::Color & _value )`

10.102.3.52 `bool sdf::Param::Set ( const gazebo::common::Time & _value )`

10.102.3.53 `void sdf::Param::SetDescription ( const std::string & _desc )`

Set the description of the parameter.

10.102.3.54 `virtual bool sdf::Param::SetFromString ( const std::string & ) [inline], [virtual]`

Set the parameter value from a string.

Reimplemented in `sdf::ParamT< T >` (p. 546), and `sdf::ParamT< std::string >` (p. 546).

10.102.3.55 `template<typename T> void sdf::Param::SetUpdateFunc ( T _updateFunc ) [inline]`

Update function.

References updateFunc.

10.102.3.56 `virtual void sdf::Param::Update ( ) [pure virtual]`

Implemented in `sdf::ParamT< T >` (p. 547), and `sdf::ParamT< std::string >` (p. 547).

## 10.102.4 Member Data Documentation

10.102.4.1 `std::string sdf::Param::description [protected]`

Referenced by `sdf::ParamT< std::string >::Clone()`, and `sdf::ParamT< std::string >::ParamT()`.

10.102.4.2 `std::string sdf::Param::key [protected]`

Referenced by `GetKey()`, `sdf::ParamT< std::string >::ParamT()`, and `sdf::ParamT< std::string >::Set()`.

10.102.4.3 `bool sdf::Param::required [protected]`

Referenced by `sdf::ParamT< std::string >::Clone()`, `GetRequired()`, `sdf::ParamT< std::string >::ParamT()`, and `sdf::ParamT< std::string >::Set()`.

10.102.4.4 `bool sdf::Param::set [protected]`

10.102.4.5 `std::string sdf::Param::typeName` [protected]

Referenced by `sdf::ParamT< std::string >::Clone()`, and `sdf::ParamT< std::string >::ParamT()`.

10.102.4.6 `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**

## 10.103 `ParamT< T >` Class Template Reference

```
#include <CommonTypes.hh>
```

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

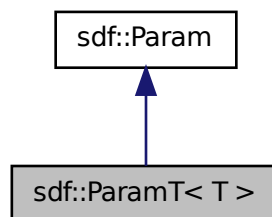
- **CommonTypes.hh**

## 10.104 `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)

#### 10.104.1 Detailed Description

```
template<typename T>class sdf::ParamT< T >
```

Templatized parameter class.

#### 10.104.2 Constructor & Destructor Documentation

10.104.2.1 `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.

10.104.2.2 `template<typename T> virtual sdf::ParamT< T >::~~ParamT ( ) [inline],[virtual]`

Destructor.

### 10.104.3 Member Function Documentation

10.104.3.1 `template<typename T> virtual Param* sdf::ParamT< T >::Clone ( ) const [inline],[virtual]`

Implements **sdf::Param** (p. 540).

10.104.3.2 `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. 541).

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

10.104.3.3 `template<typename T> virtual std::string sdf::ParamT< T >::GetDefaultAsString ( ) const [inline],[virtual]`

Reimplemented from **sdf::Param** (p. 541).

10.104.3.4 `template<typename T> T sdf::ParamT< T >::GetDefaultValue ( ) const [inline]`

Get the value.

10.104.3.5 `template<typename T> T sdf::ParamT< T >::GetValue ( ) const [inline]`

Get the value.

10.104.3.6 `template<typename T> T sdf::ParamT< T >::operator* ( ) const [inline]`

10.104.3.7 `template<typename T> virtual void sdf::ParamT< T >::Reset ( ) [inline],[virtual]`

Reset to default value.

Implements **sdf::Param** (p. 542).

10.104.3.8 `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()`.

10.104.3.9 `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. 543).

10.104.3.10 `template<typename T> void sdf::ParamT< T >::SetValue ( const T & _value ) [inline]`

Set the value of the parameter.

10.104.3.11 `template<typename T> virtual void sdf::ParamT< T >::Update ( ) [inline], [virtual]`

Update param value.

Implements `sdf::Param` (p. 543).

## 10.104.4 Friends And Related Function Documentation

10.104.4.1 `template<typename T> std::ostream& operator<< ( std::ostream & _out, const ParamT< T > & _p ) [friend]`

## 10.104.5 Member Data Documentation

10.104.5.1 `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()`.

10.104.5.2 `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**

## 10.105 gazebo::physics::PhysicsEngine Class Reference

**Base** (p. 133) 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. 693) 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. 558).*
- 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.*
- 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 gavity vector.*
- virtual void **SetMaxContacts** (double \_maxContacts)
  - : Remove this function, and replace it with a more generic property map*
- virtual void **SetSeed** (uint32\_t \_seed)
  - 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.*

### 10.105.1 Detailed Description

**Base** (p. 133) class for a physics engine.

### 10.105.2 Constructor & Destructor Documentation

#### 10.105.2.1 gazebo::physics::PhysicsEngine::PhysicsEngine ( WorldPtr *\_world* ) [explicit]

Default constructor.

##### Parameters

in	<i>_world</i>	Pointer to the world.
----	---------------	-----------------------

#### 10.105.2.2 virtual gazebo::physics::PhysicsEngine::~PhysicsEngine ( ) [virtual]

Destructor.

### 10.105.3 Member Function Documentation

#### 10.105.3.1 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.

#### 10.105.3.2 CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision ( const std::string & *\_shapeType*, const std::string & *\_linkName* )

Create a collision.

##### Parameters

in	<i>_shapeType</i>	Type of collision to create.
in	<i>_linkName</i>	Name of the parent link.

10.105.3.3 `virtual JointPtr gazebo::physics::PhysicsEngine::CreateJoint ( const std::string & _type, ModelPtr _parent )`  
`[pure virtual]`

Create a new joint.

#### Parameters

<code>in</code>	<code><i>_type</i></code>	Type of joint to create.
<code>in</code>	<code><i>_parent</i></code>	<b>Model</b> (p. 469) parent.

10.105.3.4 `virtual LinkPtr gazebo::physics::PhysicsEngine::CreateLink ( ModelPtr _parent )` `[pure virtual]`

Create a new body.

#### Parameters

<code>in</code>	<code><i>_parent</i></code>	Parent model for the link.
-----------------	-----------------------------	----------------------------

10.105.3.5 `virtual ShapePtr gazebo::physics::PhysicsEngine::CreateShape ( const std::string & _shapeType, CollisionPtr _collision )`  
`[pure virtual]`

Create a **physics::Shape** (p. 693) object.

#### Parameters

<code>in</code>	<code><i>_shapeType</i></code>	Type of shape to create.
<code>in</code>	<code><i>_collision</i></code>	<b>Collision</b> (p. 190) parent.

10.105.3.6 `virtual void gazebo::physics::PhysicsEngine::DebugPrint ( ) const` `[pure virtual]`

Debug print out of the physic engine state.

10.105.3.7 `virtual void gazebo::physics::PhysicsEngine::Fini ( )` `[virtual]`

Finilize the physics engine.

10.105.3.8 `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.

10.105.3.9 `ContactManager* gazebo::physics::PhysicsEngine::GetContactManager ( ) const`

Get a pointer to the contact manger.

**Returns**

Pointer to the contact manager.

10.105.3.10 `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.

10.105.3.11 `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. 229) surface layer depth.

10.105.3.12 `virtual math::Vector3 gazebo::physics::PhysicsEngine::GetGravity ( ) const [virtual]`

Return the gravity vector.

**Returns**

The gravity vector.

10.105.3.13 `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 allowed contacts.

10.105.3.14 `boost::recursive_mutex* gazebo::physics::PhysicsEngine::GetPhysicsUpdateMutex ( ) const [inline]`

returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 558).

**Returns**

Pointer to the physics mutex.

References physicsUpdateMutex.

10.105.3.15 `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.

10.105.3.16 `virtual int gazebo::physics::PhysicsEngine::GetSORPGSPreconItrs ( ) [inline],[virtual]`

: Remove this function, and replace it with a more generic property map access functions to set ODE parameters.

**Returns**

SORPGS precondition iterations.

10.105.3.17 `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.

10.105.3.18 `virtual double gazebo::physics::PhysicsEngine::GetStepTime ( ) [pure virtual]`

Get the simulation step time.

**Returns**

Simulation step time.

10.105.3.19 `double gazebo::physics::PhysicsEngine::GetUpdatePeriod ( )`

Get the simulation update period.

**Returns**

Simulation update period.

10.105.3.20 `double gazebo::physics::PhysicsEngine::GetUpdateRate ( )`

Get the simulation update rate.

**Returns**

Update rate.

10.105.3.21 `virtual double gazebo::physics::PhysicsEngine::GetWorldCFM ( ) [inline],[virtual]`

: Remove this function, and replace it with a more generic property map

Get **World** (p. 875) CFM.

Returns

**World** (p. 875) CFM.

10.105.3.22 `virtual double gazebo::physics::PhysicsEngine::GetWorldERP ( ) [inline],[virtual]`

: Remove this function, and replace it with a more generic property map

Get **World** (p. 875) ERP.

Returns

**World** (p. 875) ERP.

10.105.3.23 `virtual void gazebo::physics::PhysicsEngine::Init ( ) [pure virtual]`

Initialize the physics engine.

10.105.3.24 `virtual void gazebo::physics::PhysicsEngine::InitForThread ( ) [pure virtual]`

Init the engine for threads.

10.105.3.25 `virtual void gazebo::physics::PhysicsEngine::Load ( sdf::ElementPtr _sdf ) [virtual]`

Load the physics engine.

Parameters

in	_sdf	Pointer to the SDF parameters.
----	------	--------------------------------

10.105.3.26 `virtual void gazebo::physics::PhysicsEngine::OnPhysicsMsg ( ConstPhysicsPtr & _msg ) [protected],[virtual]`

virtual callback for gztopic "~/physics".

Parameters

in	_msg	Physics message.
----	------	------------------

10.105.3.27 `virtual void gazebo::physics::PhysicsEngine::OnRequest ( ConstRequestPtr & _msg ) [protected],[virtual]`

virtual callback for gztopic "~/request".

## Parameters

in	<code>_msg</code>	Request message.
----	-------------------	------------------

10.105.3.28 `virtual void gazebo::physics::PhysicsEngine::Reset ( ) [inline],[virtual]`

Rest the physics engine.

10.105.3.29 `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.
----	---------------------------	--

10.105.3.30 `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.
----	-------------------	--------------------------

10.105.3.31 `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
----	--------------------------	---------------------

10.105.3.32 `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.
----	-----------------------	---------------------

10.105.3.33 `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.
----	---------------------------	-----------------------------

10.105.3.34 `virtual void gazebo::physics::PhysicsEngine::SetSeed ( uint32_t _seed ) [virtual]`

Set the random number seed for the physics engine.

#### Parameters

in	<code>_seed</code>	The random number seed.
----	--------------------	-------------------------

10.105.3.35 `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.
----	--------------------	-----------------------

10.105.3.36 `virtual void gazebo::physics::PhysicsEngine::SetSORPGSPreconIters ( unsigned int _iters ) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

#### Parameters

in	<code>_iter</code>	Number of iterations.
----	--------------------	-----------------------

10.105.3.37 `virtual void gazebo::physics::PhysicsEngine::SetSORPGSW ( double _w ) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

#### Parameters

in	<code>_w</code>	SORPGSW value.
----	-----------------	----------------

10.105.3.38 `virtual void gazebo::physics::PhysicsEngine::SetStepTime ( double _value ) [pure virtual]`

Set the simulation step time.



## Parameters

in	_value	Value of the step time.
----	--------	-------------------------

10.105.3.39 void gazebo::physics::PhysicsEngine::SetUpdateRate ( double \_value )

Set the simulation update rate.

## Parameters

in	_value	Value of the update rate.
----	--------	---------------------------

10.105.3.40 virtual void gazebo::physics::PhysicsEngine::SetWorldCFM ( double \_cfm ) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

## Parameters

in	_cfm	Constraint force mixing.
----	------	--------------------------

10.105.3.41 virtual void gazebo::physics::PhysicsEngine::SetWorldERP ( double \_erp ) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

## Parameters

in	_erp	Error reduction parameter.
----	------	----------------------------

10.105.3.42 virtual void gazebo::physics::PhysicsEngine::UpdateCollision ( ) [pure virtual]

Update the physics engine collision.

10.105.3.43 virtual void gazebo::physics::PhysicsEngine::UpdatePhysics ( ) [inline],[virtual]

Update the physics engine.

## 10.105.4 Member Data Documentation

10.105.4.1 **ContactManager\*** gazebo::physics::PhysicsEngine::contactManager [protected]

Class that handles all contacts generated by the physics engine.

10.105.4.2 **transport::NodePtr** gazebo::physics::PhysicsEngine::node [protected]

Node for communication.

10.105.4.3 `transport::SubscriberPtr gazebo::physics::PhysicsEngine::physicsSub` [protected]

Subscribe to the physics topic.

10.105.4.4 `boost::recursive_mutex* gazebo::physics::PhysicsEngine::physicsUpdateMutex` [protected]

Mutex to protect the update cycle.

Referenced by `GetPhysicsUpdateMutex()`.

10.105.4.5 `transport::SubscriberPtr gazebo::physics::PhysicsEngine::requestSub` [protected]

Subscribe to the request topic.

10.105.4.6 `transport::PublisherPtr gazebo::physics::PhysicsEngine::responsePub` [protected]

Response publisher.

10.105.4.7 `sdf::ElementPtr gazebo::physics::PhysicsEngine::sdf` [protected]

Our SDF values.

10.105.4.8 `WorldPtr gazebo::physics::PhysicsEngine::world` [protected]

Pointer to the world.

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

- **PhysicsEngine.hh**

## 10.106 gazebo::physics::PhysicsFactory Class Reference

The physics factory instantiates different physics engines.

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

### Static Public Member Functions

- 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.*

### 10.106.1 Detailed Description

The physics factory instantiates different physics engines.

## 10.106.2 Member Function Documentation

10.106.2.1 `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>	<b>World</b> (p. 875) to pass to the created physics engine.

10.106.2.2 `static void gazebo::physics::PhysicsFactory::RegisterAll ( ) [static]`

Register everything.

10.106.2.3 `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**

## 10.107 gazebo::common::PID Class Reference

Generic **PID** (p. 559) 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. 559) controller.*
- void **GetErrors** (double &\_pe, double &\_ie, double &\_de)  
*Return **PID** (p. 559) 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. 559) controller.*
  - void **SetCmdMax** (double \_c)  
*Set the maximum value for the command.*
  - void **SetCmdMin** (double \_c)  
*Set the maximum value for the command.*
  - void **SetDGain** (double \_d)  
*Set the derivative Gain.*
  - void **SetIGain** (double \_i)  
*Set the integral Gain.*
  - void **SetIMax** (double \_i)  
*Set the integral upper limit.*
  - void **SetIMin** (double \_i)  
*Set the integral lower limit.*
  - void **SetPGain** (double \_p)  
*Set the proportional Gain.*
  - double **Update** (double \_error, **common::Time** \_dt)  
*Update the Pid loop with nonuniform time step size.*

### 10.107.1 Detailed Description

Generic **PID** (p. 559) 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.

### 10.107.2 Constructor & Destructor Documentation

10.107.2.1 **gazebo::common::PID::PID** ( double *p* = 0.0, double *i* = 0.0, double *d* = 0.0, double *imax* = 0.0, double *imin* = 0.0, double *cmdMax* = 0.0, double *cmdMin* = 0.0 )

Constructor, zeros out Pid values when created and initialize Pid-gains and integral term limits:[iMax:iMin]-[I1:I2].

#### Parameters

in	<i>_p</i>	The proportional gain.
in	<i>_i</i>	The integral gain.
in	<i>_d</i>	The derivative gain.
in	<i>_imax</i>	The integral upper limit.
in	<i>_imin</i>	The integral lower limit.

10.107.2.2 virtual gazebo::common::PID::~~PID( ) [virtual]

Destructor.

### 10.107.3 Member Function Documentation

10.107.3.1 double gazebo::common::PID::GetCmd( )

Return current command for this **PID** (p. 559) controller.

Returns

the command value

10.107.3.2 void gazebo::common::PID::GetErrors( double & \_pe, double & \_ie, double & \_de )

Return **PID** (p. 559) error terms for the controller.

Parameters

in	_pe	The proportional error.
in	_ie	The integral error.
in	_de	The derivative error.

10.107.3.3 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	_p	The proportional gain.
in	_i	The integral gain.
in	_d	The derivative gain.
in	_imax	The integral upper limit.
in	_imin	The integral lower limit.

10.107.3.4 PID& gazebo::common::PID::operator=( const PID & \_p ) [inline]

Assignment operator.

Parameters

in	_p	a reference to a <b>PID</b> (p. 559) to assign values from
----	----	--

Returns

reference to this instance

References Reset().

### 10.107.3.5 void gazebo::common::PID::Reset ( )

Reset the errors and command.

Referenced by operator=().

### 10.107.3.6 void gazebo::common::PID::SetCmd ( double *\_cmd* )

Set current target command for this **PID** (p. 559) controller.

#### Parameters

<i>in</i>	<i>_cmd</i>	New command
-----------	-------------	-------------

### 10.107.3.7 void gazebo::common::PID::SetCmdMax ( double *\_c* )

Set the maximum value for the command.

#### Parameters

<i>in</i>	<i>_c</i>	The maximum value
-----------	-----------	-------------------

### 10.107.3.8 void gazebo::common::PID::SetCmdMin ( double *\_c* )

Set the maximum value for the command.

#### Parameters

<i>in</i>	<i>_c</i>	The maximum value
-----------	-----------	-------------------

### 10.107.3.9 void gazebo::common::PID::SetDGain ( double *\_d* )

Set the derivative Gain.

#### Parameters

<i>in</i>	<i>_p</i>	derivative gain value
-----------	-----------	-----------------------

### 10.107.3.10 void gazebo::common::PID::SetIGain ( double *\_i* )

Set the integral Gain.

#### Parameters

<i>in</i>	<i>_p</i>	integral gain value
-----------	-----------	---------------------

### 10.107.3.11 void gazebo::common::PID::SetIMax ( double *\_i* )

Set the integral upper limit.

## Parameters

<code>in</code>	<code>_p</code>	integral upper limit value
-----------------	-----------------	----------------------------

10.107.3.12 `void gazebo::common::PID::SetIMin ( double i )`

Set the integral lower limit.

## Parameters

<code>in</code>	<code>_p</code>	integral lower limit value
-----------------	-----------------	----------------------------

10.107.3.13 `void gazebo::common::PID::SetPGain ( double p )`

Set the proportional Gain.

## Parameters

<code>in</code>	<code>_p</code>	proportional gain value
-----------------	-----------------	-------------------------

10.107.3.14 `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**

## 10.108 gazebo::math::Plane Class Reference

**A** (p. 107) 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 distance.*

- **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. 563) offset.*

- **Vector3** **normal**

*Plane (p. 563) normal.*

- **Vector2d** **size**

*Plane (p. 563) size.*

## 10.108.1 Detailed Description

**A** (p. 107) plane and related functions.

## 10.108.2 Constructor & Destructor Documentation

### 10.108.2.1 gazebo::math::Plane::Plane ( )

Constructor.

### 10.108.2.2 gazebo::math::Plane::Plane ( const Vector3 & \_normal, double \_offset = 0.0 )

Constructor from a normal and a distance.

#### Parameters

in	<i>_normal</i>	The plane normal
in	<i>_offset</i>	Offset along the normal

### 10.108.2.3 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	<code>_offset</code>	Offset along the normal
----	----------------------	-------------------------

#### 10.108.2.4 virtual gazebo::math::Plane::~~Plane ( ) [virtual]

Destructor.

### 10.108.3 Member Function Documentation

#### 10.108.3.1 double gazebo::math::Plane::Distance ( const Vector3 & *\_origin*, const Vector3 & *\_dir* ) const

Get distance to the plane give an origin and direction.

##### Parameters

in	<code>_origin</code>	the origin
in	<code>_dir</code>	a direction

##### Returns

the shortest distance

#### 10.108.3.2 Plane& gazebo::math::Plane::operator= ( const Plane & *\_p* )

Equal operator.

##### Parameters

	<code>_p</code>	another plane
--	-----------------	---------------

##### Returns

itself

#### 10.108.3.3 void gazebo::math::Plane::Set ( const Vector3 & *\_normal*, const Vector2d & *\_size*, double *offset* )

Set the plane.

##### Parameters

in	<code>_normal</code>	The plane normal
in	<code>_size</code>	Size of the plane
in	<code>_offset</code>	Offset along the normal

### 10.108.4 Member Data Documentation

10.108.4.1 `double gazebo::math::Plane::d`

**Plane** (p. 563) offset.

10.108.4.2 `Vector3 gazebo::math::Plane::normal`

**Plane** (p. 563) normal.

10.108.4.3 `Vector2d gazebo::math::Plane::size`

**Plane** (p. 563) size.

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

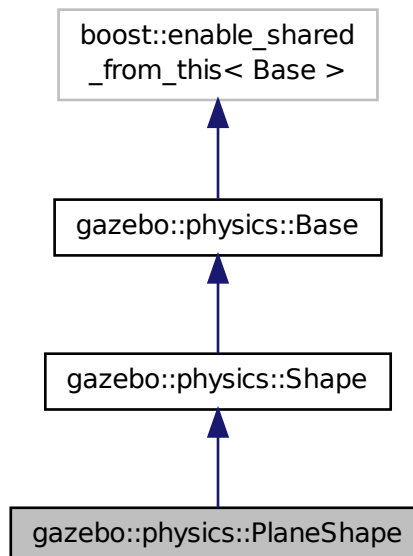
- **Plane.hh**

## 10.109 `gazebo::physics::PlaneShape` Class Reference

**Collision** (p. 190) for an infinite plane.

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

Inheritance diagram for `gazebo::physics::PlaneShape`:



## Public Member Functions

- **PlaneShape** (**CollisionPtr** \_parent)  
*Constructor.*
- virtual  $\sim$ **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

### 10.109.1 Detailed Description

**Collision** (p. 190) 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.

### 10.109.2 Constructor & Destructor Documentation

10.109.2.1 gazebo::physics::PlaneShape::PlaneShape ( **CollisionPtr** \_parent ) [explicit]

Constructor.

Parameters

in	<b>_parent</b>	<b>Link</b> (p. 404) to which we are attached.
----	----------------	--

10.109.2.2 virtual gazebo::physics::PlaneShape::~~PlaneShape ( ) [virtual]

Destructor.

### 10.109.3 Member Function Documentation

10.109.3.1 `virtual void gazebo::physics::PlaneShape::CreatePlane ( ) [virtual]`

Create the plane.

10.109.3.2 `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. 695).

10.109.3.3 `math::Vector3 gazebo::physics::PlaneShape::GetNormal ( ) const`

Get the plane normal.

#### Returns

The plane normal.

10.109.3.4 `math::Vector2d gazebo::physics::PlaneShape::GetSize ( ) const`

Get the size.

#### Returns

Size of the plane.

10.109.3.5 `virtual void gazebo::physics::PlaneShape::Init ( ) [virtual]`

Initialize the plane.

Implements `gazebo::physics::Shape` (p. 695).

10.109.3.6 `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. 695).

10.109.3.7 `virtual void gazebo::physics::PlaneShape::SetAltitude ( const math::Vector3 & _pos ) [virtual]`

Set the altitude of the plane.

#### Parameters

in	<i>_pos</i>	Position of the plane.
----	-------------	------------------------

10.109.3.8 `void gazebo::physics::PlaneShape::SetNormal ( const math::Vector3 & _norm )`

Set the normal.

#### Parameters

in	<i>_norm</i>	Plane normal.
----	--------------	---------------

10.109.3.9 `void gazebo::physics::PlaneShape::SetSize ( const math::Vector2d & _size )`

Set the size.

#### Parameters

in	<i>_size</i>	2D size of the plane.
----	--------------	-----------------------

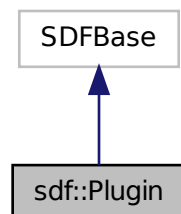
The documentation for this class was generated from the following file:

- **PlaneShape.hh**

## 10.110 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**

### 10.110.1 Constructor & Destructor Documentation

10.110.1.1 `sdf::Plugin::Plugin ( )` [`inline`]

### 10.110.2 Member Function Documentation

10.110.2.1 `void sdf::Plugin::Clear ( )` [`inline`]

References data.

10.110.2.2 `void sdf::Plugin::Print ( const std::string & prefix )` [`inline`]

References filename, and name.

### 10.110.3 Member Data Documentation

10.110.3.1 `std::vector<ParamT<std::string>> sdf::Plugin::data`

Referenced by Clear().

10.110.3.2 `ParamT<std::string> sdf::Plugin::filename`

Referenced by Print().

10.110.3.3 `ParamT<std::string> sdf::Plugin::name`

Referenced by Print().

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

- `sdf/interface/Plugin.hh`

## 10.111 gazebo::PluginT< T > Class Template Reference

**A** (p. 107) 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.*

#### 10.111.1 Detailed Description

```
template<class T>class gazebo::PluginT< T >
```

**A** (p. 107) class which all plugins must inherit from.

#### 10.111.2 Member Typedef Documentation

10.111.2.1 template<class T> typedef T\* gazebo::PluginT< T >::TPtr

plugin pointer type definition

### 10.111.3 Member Function Documentation

10.111.3.1 `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

10.111.3.2 `template<class T> std::string gazebo::PluginT< T >::GetFilename ( ) const [inline]`

Get the name of the handler.

10.111.3.3 `template<class T> std::string gazebo::PluginT< T >::GetHandle ( ) const [inline]`

Get the short name of the handler.

10.111.3.4 `template<class T> PluginType gazebo::PluginT< T >::GetType ( ) const [inline]`

Returns the type of the plugin.

#### Returns

type of the plugin

### 10.111.4 Member Data Documentation

10.111.4.1 `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()`.

10.111.4.2 `template<class T> std::string gazebo::PluginT< T >::handle [protected]`

Short name.

Referenced by `gazebo::PluginT< ModelPlugin >::Create()`, and `gazebo::PluginT< ModelPlugin >::GetHandle()`.



10.111.4.3 `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`

## 10.112 gazebo::math::Pose Class Reference

Encapsulates a position and rotation in three space.

```
#include <math/gzmath.hh>
```

### Public Member Functions

- **Pose** ()  
*Default constructors.*
- **Pose** (const **Vector3** &\_pos, const **Quaternion** &\_rot)  
*Constructor.*
- **Pose** (double \_x, double \_y, double \_z, double \_roll, double \_pitch, double \_yaw)  
*Constructor.*
- **Pose** (const **Pose** &\_pose)  
*Copy constructor.*
- virtual  $\sim$ **Pose** ()  
*Destructor.*
- **Pose CoordPoseSolve** (const **Pose** &\_b) const  
*Find the inverse of a pose; i.e., if  $b = this + a$ , given  $b$  and  $this$ , find  $a$ .*
- **Vector3 CoordPositionAdd** (const **Vector3** &\_pos) const  
*Add one point to a vector: result = this + pos.*
- **Vector3 CoordPositionAdd** (const **Pose** &\_pose) const  
*Add one point to another: result = this + pose.*
- **Vector3 CoordPositionSub** (const **Pose** &\_pose) const  
*Subtract one position from another: result = this - pose.*
- **Quaternion CoordRotationAdd** (const **Quaternion** &\_rot) const  
*Add one rotation to another: result = this->rot + rot.*
- **Quaternion CoordRotationSub** (const **Quaternion** &\_rot) const  
*Subtract one rotation from another: result = this->rot - rot.*
- void **Correct** ()  
*Fix any nan values.*
- **Pose GetInverse** () const  
*Get the inverse of this pose.*
- bool **IsFinite** () const  
*See if a pose is finite (e.g., not nan)*
- bool **operator!=** (const **Pose** &\_pose) const  
*Inequality operator.*

- **Pose operator\*** (const **Pose** &\_pose)  
*Multiplication operator.*
- **Pose operator+** (const **Pose** &\_pose) const  
*Addition operator.*
- 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. 821) and a **Quaternion** (p. 598).*
- 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.*

### 10.112.1 Detailed Description

Encapsulates a position and rotation in three space.

## 10.112.2 Constructor & Destructor Documentation

### 10.112.2.1 gazebo::math::Pose::Pose ( )

Default constructors.

Referenced by operator-().

### 10.112.2.2 gazebo::math::Pose::Pose ( const Vector3 & *\_pos*, const Quaternion & *\_rot* )

Constructor.

#### Parameters

in	<i>_pos</i>	<b>A</b> (p. 107) position
in	<i>_rot</i>	<b>A</b> (p. 107) rotation

### 10.112.2.3 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.

### 10.112.2.4 gazebo::math::Pose::Pose ( const Pose & *\_pose* )

Copy constructor.

#### Parameters

in	<i>_pose</i>	<b>Pose</b> (p. 573) to copy
----	--------------	------------------------------

### 10.112.2.5 virtual gazebo::math::Pose::~~Pose ( ) [virtual]

Destructor.

## 10.112.3 Member Function Documentation

### 10.112.3.1 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  $\text{this}$ , find  $a$ .

## Parameters

in	<code>_b</code>	the other pose
----	-----------------	----------------

### 10.112.3.2 `Vector3 gazebo::math::Pose::CoordPositionAdd ( const Vector3 & _pos ) const`

Add one point to a vector: result = this + pos.

## Parameters

in	<code>_pos</code>	Position to add to this pose
----	-------------------	------------------------------

## Returns

the resulting position

### 10.112.3.3 `Vector3 gazebo::math::Pose::CoordPositionAdd ( const Pose & _pose ) const`

Add one point to another: result = this + pose.

## Parameters

in	<code>_pose</code>	The <b>Pose</b> (p. 573) to add
----	--------------------	---------------------------------

## Returns

The resulting position

### 10.112.3.4 `Vector3 gazebo::math::Pose::CoordPositionSub ( const Pose & _pose ) const` `[inline]`

Subtract one position from another: result = this - pose.

## Parameters

in	<code>_pose</code>	<b>Pose</b> (p. 573) 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-().

### 10.112.3.5 `Quaternion gazebo::math::Pose::CoordRotationAdd ( const Quaternion & _rot ) const`

Add one rotation to another: result = this->rot + rot.

## Parameters

<code>in</code>	<code>_rot</code>	Rotation to add
-----------------	-------------------	-----------------

## Returns

The resulting rotation

### 10.112.3.6 Quaternion gazebo::math::Pose::CoordRotationSub ( const Quaternion & \_rot ) const `[inline]`

Subtract one rotation from another: result = this->rot - rot.

## Parameters

<code>in</code>	<code>_rot</code>	The rotation to subtract
-----------------	-------------------	--------------------------

## Returns

The resulting rotation

References gazebo::math::Quaternion::GetInverse(), gazebo::math::Quaternion::Normalize(), and rot.

Referenced by operator-().

### 10.112.3.7 void gazebo::math::Pose::Correct ( ) `[inline]`

Fix any nan values.

References gazebo::math::Vector3::Correct(), gazebo::math::Quaternion::Correct(), pos, and rot.

### 10.112.3.8 Pose gazebo::math::Pose::GetInverse ( ) const

Get the inverse of this pose.

## Returns

the inverse pose

### 10.112.3.9 bool gazebo::math::Pose::IsFinite ( ) const

See if a pose is finite (e.g., not nan)

### 10.112.3.10 bool gazebo::math::Pose::operator!= ( const Pose & \_pose ) const

Inequality operator.

## Parameters

<code>in</code>	<code>_pose</code>	<b>Pose</b> (p. 573) for comparison
-----------------	--------------------	-------------------------------------

**Returns**

True if not equal

**10.112.3.11 Pose gazebo::math::Pose::operator\* ( const Pose & *\_pose* )**

Multiplication operator.

**Parameters**

<i>in</i>	<i>_pose</i>	the other pose
-----------	--------------	----------------

**Returns**

itself

**10.112.3.12 Pose gazebo::math::Pose::operator+ ( const Pose & *\_pose* ) const**

Addition operator.

**Parameters**

<i>in</i>	<i>_pose</i>	<b>Pose</b> (p. 573) to add to this pose
-----------	--------------	--

**Returns**

The resulting pose

**10.112.3.13 const Pose& gazebo::math::Pose::operator+= ( const Pose & *\_pose* )**

Add-Equals operator.

**Parameters**

<i>in</i>	<i>_pose</i>	<b>Pose</b> (p. 573) to add to this pose
-----------	--------------	--

**Returns**

The resulting pose

**10.112.3.14 Pose gazebo::math::Pose::operator- ( ) const [inline]**

Negation operator.

**Returns**

The resulting pose

References Pose().

10.112.3.15 **Pose** gazebo::math::Pose::operator- ( const Pose & *\_pose* ) const [inline]

Subtraction operator.

#### Parameters

in	<i>_pose</i>	<b>Pose</b> (p. 573) to subtract from this one
----	--------------	--

#### Returns

The resulting pose

References CoordPositionSub(), CoordRotationSub(), Pose(), and rot.

10.112.3.16 **const Pose&** gazebo::math::Pose::operator-= ( const Pose & *\_pose* )

Subtraction operator.

#### Parameters

in	<i>_pose</i>	<b>Pose</b> (p. 573) to subtract from this one
----	--------------	--

#### Returns

The resulting pose

10.112.3.17 **bool** gazebo::math::Pose::operator== ( const Pose & *\_pose* ) const

Equality operator.

#### Parameters

in	<i>_pose</i>	<b>Pose</b> (p. 573) for comparison
----	--------------	-------------------------------------

#### Returns

True if equal

10.112.3.18 **void** gazebo::math::Pose::Reset ( )

Reset the pose.

10.112.3.19 **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

10.112.3.20 `void gazebo::math::Pose::Round ( int _precision )`

Round all values to `_precision` decimal places.

**Parameters**

<code>in</code>	<code><i>_precision</i></code>	
-----------------	--------------------------------	--

10.112.3.21 `void gazebo::math::Pose::Set ( const Vector3 & _pos, const Quaternion & _rot )`

Set the pose from a **Vector3** (p. 821) and a **Quaternion** (p. 598).

**Parameters**

<code>in</code>	<code><i>_pos</i></code>	The position.
<code>in</code>	<code><i>_rot</i></code>	The rotation.

10.112.3.22 `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**

<code>in</code>	<code><i>_x</i></code>	x position in meters.
<code>in</code>	<code><i>_y</i></code>	y position in meters.
<code>in</code>	<code><i>_z</i></code>	z position in meters.
<code>in</code>	<code><i>_roll</i></code>	Roll (rotation about X-axis) in radians.
<code>in</code>	<code><i>_pitch</i></code>	Pitch (rotation about y-axis) in radians.
<code>in</code>	<code><i>_yaw</i></code>	Pitch (rotation about z-axis) in radians.

**10.112.4 Friends And Related Function Documentation**

10.112.4.1 `std::ostream& operator<< ( std::ostream & _out, const gazebo::math::Pose & _pose )` [`friend`]

Stream insertion operator.

**Parameters**

<code>in</code>	<code><i>_out</i></code>	output stream
<code>in</code>	<code><i>_pose</i></code>	pose to output

**Returns**

the stream



10.112.4.2 `std::istream& operator>> ( std::istream & _in, gazebo::math::Pose & _pose )` [`friend`]

Stream extraction operator.

#### Parameters

<code><i>in</i></code>	<code><i>_in</i></code>	the input stream
<code><i>in</i></code>	<code><i>_pose</i></code>	the pose

#### Returns

the stream

### 10.112.5 Member Data Documentation

#### 10.112.5.1 Vector3 gazebo::math::Pose::pos

The position.

Referenced by `CoordPositionSub()`, and `Correct()`.

#### 10.112.5.2 Quaternion gazebo::math::Pose::rot

The rotation.

Referenced by `CoordPositionSub()`, `CoordRotationSub()`, `Correct()`, and `operator-()`.

#### 10.112.5.3 `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:

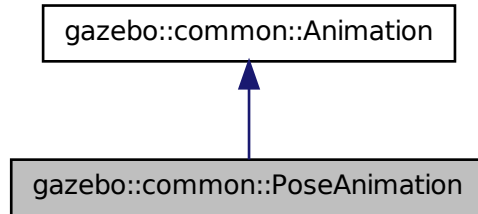
- **Pose.hh**

## 10.113 gazebo::common::PoseAnimation Class Reference

**A** (p. 107) 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

### 10.113.1 Detailed Description

**A** (p. 107) pose animation.

### 10.113.2 Constructor & Destructor Documentation

10.113.2.1 gazebo::common::PoseAnimation::PoseAnimation ( const std::string & \_name, double \_length, bool \_loop )

Constructor.

## Parameters

in	<i>_name</i>	String name of the animation. This should be unique.
in	<i>_length</i>	Length of the animation in seconds
in	<i>_loop</i>	True == loop the animation

10.113.2.2 virtual gazebo::common::PoseAnimation::~~PoseAnimation ( ) [virtual]

Destructor.

## 10.113.3 Member Function Documentation

10.113.3.1 void gazebo::common::PoseAnimation::BuildInterpolationSplines ( ) const [protected]

Update the pose splines.

10.113.3.2 PoseKeyFrame\* gazebo::common::PoseAnimation::CreateKeyFrame ( double *\_time* )

Create a pose keyframe at the given time.

## Parameters

in	<i>_time</i>	<b>Time</b> (p. 760) at which to create the keyframe
----	--------------	--

## Returns

Pointer to the new keyframe

10.113.3.3 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame ( PoseKeyFrame & *\_kf* ) const

Get a keyframe using the animation's current time.

## Parameters

out	<i>_kf</i>	<b>PoseKeyFrame</b> (p. 584) reference to hold the interpolated result
-----	------------	--

10.113.3.4 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame ( double *\_time*, PoseKeyFrame & *\_kf* ) const [protected]

Get a keyframe using a passed in time.

## Parameters

in	<i>_time</i>	<b>Time</b> (p. 760) in seconds
out	<i>_kf</i>	<b>PoseKeyFrame</b> (p. 584) reference to hold the interpolated result

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

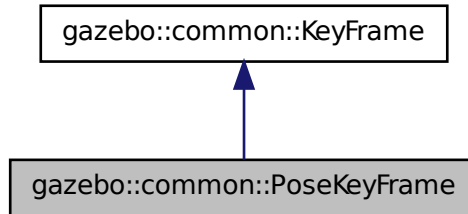
- **Animation.hh**

## 10.114 gazebo::common::PoseKeyFrame Class Reference

**A** (p. 107) keyframe for a **PoseAnimation** (p. 581).

```
#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*

#### 10.114.1 Detailed Description

**A** (p. 107) keyframe for a **PoseAnimation** (p. 581).

## 10.114.2 Constructor & Destructor Documentation

### 10.114.2.1 gazebo::common::PoseKeyFrame::PoseKeyFrame ( double *\_time* )

Constructor.

#### Parameters

in	<i>_time</i>	of the keyframe
----	--------------	-----------------

### 10.114.2.2 virtual gazebo::common::PoseKeyFrame::~~PoseKeyFrame ( ) [virtual]

Destructor.

## 10.114.3 Member Function Documentation

### 10.114.3.1 const math::Quaternion& gazebo::common::PoseKeyFrame::GetRotation ( ) const

Get the rotation of the keyframe.

#### Returns

The rotation amount

### 10.114.3.2 const math::Vector3& gazebo::common::PoseKeyFrame::GetTranslation ( ) const

Get the translation of the keyframe.

#### Returns

The translation amount

### 10.114.3.3 void gazebo::common::PoseKeyFrame::SetRotation ( const math::Quaternion & *\_rot* )

Set the rotation for the keyframe.

#### Parameters

in	<i>_rot</i>	Rotation amount
----	-------------	-----------------

### 10.114.3.4 void gazebo::common::PoseKeyFrame::SetTranslation ( const math::Vector3 & *\_trans* )

Set the translation for the keyframe.

#### Parameters

in	<i>_trans</i>	Translation amount
----	---------------	--------------------

### 10.114.4 Member Data Documentation

#### 10.114.4.1 `math::Quaternion gazebo::common::PoseKeyFrame::rotate` [protected]

the rotation quaternion

#### 10.114.4.2 `math::Vector3 gazebo::common::PoseKeyFrame::translate` [protected]

the translation vector

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

- **KeyFrame.hh**

## 10.115 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.*

### 10.115.1 Detailed Description

Projects a material onto surface, light a light projector.

## 10.115.2 Constructor & Destructor Documentation

### 10.115.2.1 gazebo::rendering::Projector::Projector ( VisualPtr *\_parent* )

Constructor.

#### Parameters

in	<i>_parent</i>	Name of the parent visual.
----	----------------	----------------------------

### 10.115.2.2 virtual gazebo::rendering::Projector::~~Projector ( ) [virtual]

Destructor.

## 10.115.3 Member Function Documentation

### 10.115.3.1 VisualPtr gazebo::rendering::Projector::GetParent ( )

Get the parent visual.

#### Returns

Pointer of the parent visual.

### 10.115.3.2 void gazebo::rendering::Projector::Load ( sdf::ElementPtr *\_sdf* )

Load from an sdf pointer.

#### Parameters

in	<i>_sdf</i>	Pointer to the SDF element.
----	-------------	-----------------------------

### 10.115.3.3 void gazebo::rendering::Projector::Load ( const msgs::Projector & *\_msg* )

Load from a message.

#### Parameters

in	<i>_msg</i>	Load from a message.
----	-------------	----------------------

### 10.115.3.4 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.

10.115.3.5 `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.
----	-----------------------	-------------------------------

10.115.3.6 `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.
----	---------------------------	---------------------------------

10.115.3.7 `void gazebo::rendering::Projector::Toggle ( )`

Toggle the activation of the projector.

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

- **Projector.hh**

## 10.116 gazebo::transport::Publication Class Reference

**A** (p. 107) 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.*

### 10.116.1 Detailed Description

**A** (p. 107) publication for a topic.

This facilitates transport of messages

### 10.116.2 Constructor & Destructor Documentation

#### 10.116.2.1 gazebo::transport::Publication::Publication ( const std::string & \_topic, const std::string & \_msgType )

Constructor.

##### Parameters

in	_topic	The topic we're publishing
in	_msgType	The type of the topic we're publishing

10.116.2.2 `virtual gazebo::transport::Publication::~~Publication ( ) [virtual]`

Destructor.

### 10.116.3 Member Function Documentation

10.116.3.1 `void gazebo::transport::Publication::AddPublisher ( PublisherPtr _pub )`

Add a publisher.

#### Parameters

<code>in, out</code>	<code>_pub</code>	Pointer to publisher object to be added
----------------------	-------------------	---

Referenced by `gazebo::transport::TopicManager::Advertise()`.

10.116.3.2 `void gazebo::transport::Publication::AddSubscription ( const CallbackHelperPtr & _callback )`

Subscribe a callback to our topic.

#### Parameters

<code>in</code>	<code>_callback</code>	The callback
-----------------	------------------------	--------------

Referenced by `gazebo::transport::TopicManager::Advertise()`.

10.116.3.3 `void gazebo::transport::Publication::AddSubscription ( const NodePtr & _node )`

Subscribe a node to our topic.

#### Parameters

<code>in</code>	<code>_node</code>	The node
-----------------	--------------------	----------

10.116.3.4 `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
-----------------	-----------------------	---

10.116.3.5 `unsigned int gazebo::transport::Publication::GetCallbackCount ( ) const`

Get the number of callbacks.

#### Returns

The number of callbacks

10.116.3.6 `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()`.

10.116.3.7 `std::string gazebo::transport::Publication::GetMsgType ( ) const`

Get the type of message.

#### Returns

The type of message

10.116.3.8 `unsigned int gazebo::transport::Publication::GetNodeCount ( ) const`

Get the number of nodes.

#### Returns

The number of nodes

10.116.3.9 `unsigned int gazebo::transport::Publication::GetRemoteSubscriptionCount ( )`

Get the number of remote subscriptions.

#### Returns

The number of remote subscriptions

10.116.3.10 `unsigned int gazebo::transport::Publication::GetTransportCount ( ) const`

Get the number of transports.

#### Returns

The number of transports

10.116.3.11 `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

10.116.3.12 `void gazebo::transport::Publication::LocalPublish ( const std::string & _data )`

Publish data to local subscribers (skip serialization)

**Parameters**

<i>in</i>	<i>_data</i>	The data to be published
-----------	--------------	--------------------------

10.116.3.13 `void gazebo::transport::Publication::Publish ( const google::protobuf::Message & _msg, const boost::function< void()> & _cb = NULL )`

Publish data to remote subscribers.

**Parameters**

<i>in</i>	<i>_msg</i>	Message to be published
<i>in</i>	<i>_cb</i>	If non-null, callback to be invoked after publishing is completed

10.116.3.14 `void gazebo::transport::Publication::RemoveSubscription ( const NodePtr & _node )`

Unsubscribe a node from our topic.

**Parameters**

<i>in</i>	<i>_node</i>	The node
-----------	--------------	----------

10.116.3.15 `void gazebo::transport::Publication::RemoveSubscription ( const std::string & _host, unsigned int _port )`

Unsubscribe a a node by host/port from our topic.

**Parameters**

<i>in</i>	<i>_host</i>	The node's hostname
<i>in</i>	<i>_port</i>	The node's port

10.116.3.16 `void gazebo::transport::Publication::RemoveTransport ( const std::string & _host, unsigned int _port )`

Remove a transport.

**Parameters**

<i>in</i>	<i>_host</i>	The transport's hostname
<i>in</i>	<i>_port</i>	The transport's port

10.116.3.17 void gazebo::transport::Publication::SetLocallyAdvertised ( bool *\_value* )

Set whether this topic has been advertised from this process.

#### Parameters

in	<i>_value</i>	If true, the topic was locally advertise, otherwise it was not
----	---------------	--

Referenced by gazebo::transport::TopicManager::Advertise().

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

- **Publication.hh**

## 10.117 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.*

### 10.117.1 Detailed Description

transport/transport.hh

Reads data from a remote advertiser, and passes the data along to local subscribers

## 10.117.2 Constructor & Destructor Documentation

10.117.2.1 `gazebo::transport::PublicationTransport::PublicationTransport ( const std::string & _topic, const std::string & _msgType )`

Constructor.

### Parameters

in	<code>_topic</code>	Topic that we're publishing
in	<code>_topic</code>	Type of the topic that we're publishing

10.117.2.2 `virtual gazebo::transport::PublicationTransport::~~PublicationTransport ( ) [virtual]`

Destructor.

## 10.117.3 Member Function Documentation

10.117.3.1 `void gazebo::transport::PublicationTransport::AddCallback ( const boost::function< void(const std::string &)> & _cb )`

Add a callback to the transport.

### Parameters

in	<code>_cb</code>	The callback to be added
----	------------------	--------------------------

10.117.3.2 `void gazebo::transport::PublicationTransport::Fini ( )`

Finalize the transport.

10.117.3.3 `const ConnectionPtr gazebo::transport::PublicationTransport::GetConnection ( ) const`

Get the underlying connection.

### Returns

Pointer to the underlying connection

10.117.3.4 `std::string gazebo::transport::PublicationTransport::GetMsgType ( ) const`

Get the topic type.

### Returns

The topic type

10.117.3.5 `std::string gazebo::transport::PublicationTransport::GetTopic ( ) const`

Get the topic name.

#### Returns

The topic name

10.117.3.6 `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**

## 10.118 gazebo::transport::Publisher Class Reference

**A** (p. 107) 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)  
*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.*

### 10.118.1 Detailed Description

**A** (p. 107) publisher of messages on a topic.

### 10.118.2 Constructor & Destructor Documentation

10.118.2.1 gazebo::transport::Publisher::Publisher ( const std::string & \_topic, const std::string & \_msgType, unsigned int \_limit, bool \_latch )

Deprecated.

10.118.2.2 gazebo::transport::Publisher::Publisher ( const std::string & \_topic, const std::string & \_msgType, unsigned int \_limit )

Constructor.

#### Parameters

in	<code>_topic</code>	Name of topic to be published
in	<code>_msgType</code>	Type of the message to be published
in	<code>_limit</code>	Maximum number of outgoing messages to queue

10.118.2.3 virtual gazebo::transport::Publisher::~~Publisher ( ) [virtual]

Destructor.

### 10.118.3 Member Function Documentation

10.118.3.1 bool gazebo::transport::Publisher::GetLatching ( ) const

Deprecated.

10.118.3.2 std::string gazebo::transport::Publisher::GetMsgType ( ) const

Get the message type.



**Returns**

The message type

10.118.3.3 `unsigned int gazebo::transport::Publisher::GetOutgoingCount ( ) const`

Get the number of outgoing messages.

**Returns**

The number of outgoing messages

10.118.3.4 `std::string gazebo::transport::Publisher::GetPrevMsg ( ) const`

Get the previously published message.

**Returns**

The previously published message, if any

10.118.3.5 `std::string gazebo::transport::Publisher::GetTopic ( ) const`

Get the topic name.

**Returns**

The topic name

10.118.3.6 `bool gazebo::transport::Publisher::HasConnections ( ) const`

Are there any connections?

**Returns**

true if there are any connections, false otherwise

10.118.3.7 `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

10.118.3.8 `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>_message</code>	Message to be published
<code>in</code>	<code>_block</code>	Whether to block until the message is actually written out

10.118.3.9 `void gazebo::transport::Publisher::SendMessage ( )`

Send latest message over the wire. For internal use only.

10.118.3.10 `void gazebo::transport::Publisher::SetPublication ( PublicationPtr & _publication, int _i )`

Set the publication object for a particular publication.

#### Parameters

<code>in</code>	<code>_publication</code>	Pointer to the publication object to be set
<code>in</code>	<code>_i</code>	Index into publications vector that will be set

10.118.3.11 `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**

## 10.119 gazebo::math::Quaternion Class Reference

**A** (p. 107) 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. 821) multiplication operator.*
- **Quaternion operator\*= **(const Quaternion &qt)****  
*Multiplication operator.*
- **Quaternion operator+ **(const Quaternion &\_qt) const****  
*Addition operator.*
- **Quaternion operator+= **(const Quaternion &\_qt)****  
*Addition operator.*
- **Quaternion operator- **(const Quaternion &\_qt) const****  
*Substraction operator.*
- **Quaternion operator- **() const****  
*Unary minus operator.*
- **Quaternion operator-= **(const Quaternion &\_qt)****  
*Substraction operator.*
- **Quaternion & operator= **(const Quaternion &\_qt)****  
*Equal operator.*
- bool **operator== **(const Quaternion &\_qt) const****  
*Equal to operator.*
- **Vector3 RotateVector **(const Vector3 &\_vec) const****  
*Rotate a vector using the quaternion.*
- **Vector3 RotateVectorReverse **(Vector3 \_vec) const****  
*Do the reverse rotation of a vector by this quaternion.*
- void **Round **(int \_precision)****  
*Round all values to \_precision decimal places.*
- void **Scale **(double \_scale)****  
*Scale a Quaternionion.*
- void **Set **(double \_u, double \_x, double \_y, double \_z)****  
*Set this quaternion from 4 floating numbers.*
- void **SetFromAxis **(double \_x, double \_y, double \_z, double \_a)****  
*Set the quaternion from an axis and angle.*
- void **SetFromAxis **(const Vector3 &\_axis, double \_a)****  
*Set the quaternion from an axis and angle.*
- void **SetFromEuler **(const Vector3 &\_vec)****  
*Set the quaternion from Euler angles.*
- void **SetFromEuler **(double \_roll, double \_pitch, double \_yaw)****  
*Set the quaternion from Euler angles.*
- void **SetTolIdentity **()****  
*Set the quatern to the identity.*

### Static Public Member Functions

- static **Quaternion EulerToQuaternion **(const Vector3 &\_vec)****  
*Convert euler angles to quatern.*
- static **Quaternion EulerToQuaternion **(double \_x, double \_y, double \_z)****  
*Convert euler angles to quatern.*
- static **Quaternion Slerp **(double \_fT, const Quaternion &\_rkP, const Quaternion &\_rkQ, bool \_shortest-Path=false)****

*Spherical linear interpolation between 2 quaternions, given the ends and an interpolation parameter between 0 and 1.*

- static **Quaternion Squad** (double *\_t*, const **Quaternion** &*\_rkP*, const **Quaternion** &*\_rkA*, const **Quaternion** &*\_rkB*, const **Quaternion** &*\_rkQ*, bool *\_shortestPath*=false)

*Spherical quadratic interpolation given the ends and an interpolation parameter between 0 and 1.*

## Public Attributes

- double **w**  
*Attributes of the quaternion.*
- double **x**  
*Attributes of the quaternion.*
- double **y**  
*Attributes of the quaternion.*
- double **z**  
*Attributes of the quaternion.*

## Friends

- std::ostream & **operator**<< (std::ostream &*\_out*, const gazebo::math::Quaternion &*\_q*)  
*Stream insertion operator.*
- std::istream & **operator**>> (std::istream &*\_in*, gazebo::math::Quaternion &*\_q*)  
*Stream extraction operator.*

## 10.119.1 Detailed Description

**A** (p. 107) quaternion class.

## 10.119.2 Constructor & Destructor Documentation

### 10.119.2.1 gazebo::math::Quaternion::Quaternion ( )

Default Constructor.

Referenced by operator\*().

### 10.119.2.2 gazebo::math::Quaternion::Quaternion ( const double & *\_w*, const double & *\_x*, const double & *\_y*, const double & *\_z* )

Constructor.

#### Parameters

<i>in</i>	<i>_w</i>	W param
<i>in</i>	<i>_x</i>	X param
<i>in</i>	<i>_y</i>	Y param
<i>in</i>	<i>_z</i>	Z param

10.119.2.3 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

10.119.2.4 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

10.119.2.5 gazebo::math::Quaternion::Quaternion ( const Vector3 & *\_rpy* )

Constructor.

#### Parameters

in	<i>_rpy</i>	euler angles
----	-------------	--------------

10.119.2.6 gazebo::math::Quaternion::Quaternion ( const Quaternion & *\_qt* )

Copy constructor.

#### Parameters

<i>qt</i>	<b>Quaternion</b> (p. 598) to copy
-----------	------------------------------------

10.119.2.7 gazebo::math::Quaternion::~~Quaternion ( )

Destructor.

### 10.119.3 Member Function Documentation

10.119.3.1 void gazebo::math::Quaternion::Correct ( ) [inline]

Correct any nan.

References gazebo::math::equal(), w, x, y, and z.

Referenced by gazebo::math::Pose::Correct().

10.119.3.2 `double gazebo::math::Quaternion::Dot ( const Quaternion & _q ) const`

Dot product.

Parameters

<code>in</code>	<code>_q</code>	the other quaternion
-----------------	-----------------	----------------------

Returns

the product

10.119.3.3 `static Quaternion gazebo::math::Quaternion::EulerToQuaternion ( const Vector3 & _vec ) [static]`

Convert euler angles to quatern.

Parameters

<code>in</code>		
-----------------	--	--

10.119.3.4 `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

10.119.3.5 `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

10.119.3.6 `Vector3 gazebo::math::Quaternion::GetAsEuler ( ) const`

Return the rotation in Euler angles.

Returns

This quaternion as an Euler vector

10.119.3.7 **Matrix3** gazebo::math::Quaternion::GetAsMatrix3 ( ) const

Get the quaternion as a 3x3 matrix.

10.119.3.8 **Matrix4** gazebo::math::Quaternion::GetAsMatrix4 ( ) const

Get the quaternion as a 4x4 matrix.

Returns

a 4x4 matrix

10.119.3.9 **Quaternion** gazebo::math::Quaternion::GetExp ( ) const

Return the exponent.

Returns

the exp

10.119.3.10 **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().

10.119.3.11 **Quaternion** gazebo::math::Quaternion::GetLog ( ) const

Return the logarithm.

Returns

the log

10.119.3.12 **double** gazebo::math::Quaternion::GetPitch ( )

Get the Euler pitch angle in radians.

Returns

the pitch



10.119.3.13 `double gazebo::math::Quaternion::GetRoll ( )`

Get the Euler roll angle in radians.

Returns

the roll

10.119.3.14 `Vector3 gazebo::math::Quaternion::GetXAxis ( ) const`

Return the X axis.

Returns

the vector

10.119.3.15 `double gazebo::math::Quaternion::GetYaw ( )`

Get the Euler yaw angle in radians.

Returns

the yaw

10.119.3.16 `Vector3 gazebo::math::Quaternion::GetYAxis ( ) const`

Return the Y axis.

Returns

the vector

10.119.3.17 `Vector3 gazebo::math::Quaternion::GetZAxis ( ) const`

Return the Z axis.

Returns

the vector

10.119.3.18 `void gazebo::math::Quaternion::Invert ( )`

Invert the quaternion.

10.119.3.19 `bool gazebo::math::Quaternion::IsFinite ( ) const`

See if a quatern is finite (e.g., not nan)

Returns

True if quatern is finite

10.119.3.20 `void gazebo::math::Quaternion::Normalize ( )`

Normalize the quaternion.

Referenced by `gazebo::math::Pose::CoordRotationSub()`.

10.119.3.21 `bool gazebo::math::Quaternion::operator!=( const Quaternion & _qt ) const`

Not equal to operator.

#### Parameters

in	_qt	<b>Quaternion</b> (p. 598) for comparison
----	-----	---

#### Returns

True if not equal

10.119.3.22 `Quaternion gazebo::math::Quaternion::operator*( const Quaternion & _q ) const` `[inline]`

Multiplication operator.

#### Parameters

in	_qt	<b>Quaternion</b> (p. 598) for multiplication
----	-----	---

#### Returns

This quaternion multiplied by the parameter

References `Quaternion()`, w, x, y, and z.

10.119.3.23 `Quaternion gazebo::math::Quaternion::operator*( const double & _f ) const`

Multiplication operator.

#### Parameters

in	_f	factor
----	----	--------

#### Returns

quaternion multiplied by \_f

10.119.3.24 `Vector3 gazebo::math::Quaternion::operator*( const Vector3 & _v ) const`

**Vector3** (p. 821) multiplication operator.

#### Parameters

in	_v	vector to multiply
----	----	--------------------

## 10.119.3.25 Quaternion gazebo::math::Quaternion::operator\*=( const Quaternion &amp; qt )

Multiplication operator.

## Parameters

in	_qt	Quaternion (p. 598) for multiplication
----	-----	--

## Returns

This quatern multiplied by the parameter

## 10.119.3.26 Quaternion gazebo::math::Quaternion::operator+( const Quaternion &amp; \_qt ) const

Addition operator.

## Parameters

in	_qt	quaternion for addition
----	-----	-------------------------

## Returns

this quaternion + \_qt

## 10.119.3.27 Quaternion gazebo::math::Quaternion::operator+=( const Quaternion &amp; \_qt )

Addition operator.

## Parameters

in	_qt	quaternion for addition
----	-----	-------------------------

## Returns

this quaternion + qt

## 10.119.3.28 Quaternion gazebo::math::Quaternion::operator-( const Quaternion &amp; \_qt ) const

Substraction operator.

## Parameters

in	_qt	quaternion to subtract
----	-----	------------------------

## Returns

this quaternion - \_qt

### 10.119.3.29 Quaternion gazebo::math::Quaternion::operator- ( ) const

Unary minus operator.

#### Returns

negates each component of the quaternion

### 10.119.3.30 Quaternion gazebo::math::Quaternion::operator-= ( const Quaternion & \_qt )

Substraction operator.

#### Parameters

in	_qt	<b>Quaternion</b> (p. 598) for subtraction
----	-----	--

#### Returns

This quatern - qt

### 10.119.3.31 Quaternion& gazebo::math::Quaternion::operator= ( const Quaternion & \_qt )

Equal operator.

#### Parameters

in	_qt	<b>Quaternion</b> (p. 598) to copy
----	-----	------------------------------------

### 10.119.3.32 bool gazebo::math::Quaternion::operator==( const Quaternion & \_qt ) const

Equal to operator.

#### Parameters

in	_qt	<b>Quaternion</b> (p. 598) for comparison
----	-----	---

#### Returns

True if equal

### 10.119.3.33 Vector3 gazebo::math::Quaternion::RotateVector ( const Vector3 & \_vec ) const [inline]

Rotate a vector using the quaternion.

#### Parameters

in	_vec	vector to rotate
----	------	------------------

**Returns**

the rotated vector

References [GetInverse\(\)](#), [gazebo::math::Vector3::x](#), [gazebo::math::Vector3::y](#), [gazebo::math::Vector3::z](#), and [z](#).

**10.119.3.34 Vector3 gazebo::math::Quaternion::RotateVectorReverse ( Vector3 \_vec ) const**

Do the reverse rotation of a vector by this quaternion.

**Parameters**

<i>in</i>	<i>_vec</i>	the vector
-----------	-------------	------------

**Returns**

the

**10.119.3.35 void gazebo::math::Quaternion::Round ( int \_precision )**

Round all values to *\_precision* decimal places.

**Parameters**

<i>in</i>	<i>_precision</i>	the precision
-----------	-------------------	---------------

**10.119.3.36 void gazebo::math::Quaternion::Scale ( double \_scale )**

Scale a Quaternionion.

**Parameters**

<i>in</i>	<i>_scale</i>	Amount to scale this rotation
-----------	---------------	-------------------------------

**10.119.3.37 void gazebo::math::Quaternion::Set ( double \_u, double \_x, double \_y, double \_z )**

Set this quaternion from 4 floating numbers.

**Parameters**

<i>in</i>	<i>_u</i>	<i>u</i>
<i>in</i>	<i>_x</i>	<i>x</i>
<i>in</i>	<i>_y</i>	<i>y</i>
<i>in</i>	<i>_z</i>	<i>z</i>

**10.119.3.38 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>	<b>Angle</b> (p. 114) in radians

10.119.3.39 `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>	<b>Angle</b> (p. 114) in radians

10.119.3.40 `void gazebo::math::Quaternion::SetFromEuler ( const Vector3 & _vec )`

Set the quaternion from Euler angles.

## Parameters

in	<code>vec</code>	Euler angle
----	------------------	-------------

10.119.3.41 `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).

10.119.3.42 `void gazebo::math::Quaternion::SetToIdentity ( )`

Set the quatern to the identity.

10.119.3.43 `static Quaternion gazebo::math::Quaternion::Slerp ( double _fT, const Quaternion & _rkP, const Quaternion & _rkQ, bool _shortestPath = false ) [static]`

Spherical linear interpolation between 2 quaternions, given the ends and an interpolation parameter between 0 and 1.

## Parameters

in	<code>_fT</code>	the interpolation parameter
in	<code>_rkP</code>	the beginning quaternion
in	<code>_rkQ</code>	the end quaternion
in	<code>_shortestPath</code>	when true, the rotation may be inverted to get to minimize rotation

10.119.3.44 `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	<i>_ft</i>	the interpolation parameter
in	<i>_rkP</i>	the beginning quaternion
in	<i>_rkA</i>	first intermediate quaternion
in	<i>_rkB</i>	second intermediate quaternion
in	<i>_rkQ</i>	the end quaternion
in	<i>_shortestPath</i>	when true, the rotation may be inverted to get to minimize rotation

### 10.119.4 Friends And Related Function Documentation

10.119.4.1 `std::ostream& operator<< ( std::ostream & _out, const gazebo::math::Quaternion & _q ) [friend]`

Stream insertion operator.

#### Parameters

in	<i>_out</i>	output stream
in	<i>_q</i>	quaternion to output

#### Returns

the stream

10.119.4.2 `std::istream& operator>> ( std::istream & _in, gazebo::math::Quaternion & _q ) [friend]`

Stream extraction operator.

#### Parameters

in	<i>_in</i>	input stream
in	<i>_q</i>	<b>Quaternion</b> (p. 598) to read values into

#### Returns

The istream

### 10.119.5 Member Data Documentation

10.119.5.1 `double gazebo::math::Quaternion::w`

Attributes of the quaternion.

Referenced by `Correct()`, `GetInverse()`, and `operator*()`.

**10.119.5.2 double gazebo::math::Quaternion::x**

Attributes of the quaternion.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), GetInverse(), operator\*(), and RotateVector().

**10.119.5.3 double gazebo::math::Quaternion::y**

Attributes of the quaternion.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), GetInverse(), operator\*(), and RotateVector().

**10.119.5.4 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**

**10.120 gazebo::math::Rand Class Reference**

Random number generator class.

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

**Static Public Member Functions**

- static double **GetDbfNormal** (double \_mean=0, double \_sigma=1)  
*Get a double from a normal distribution.*
- static double **GetDbfUniform** (double \_min=0, double \_max=1)  
*Get a double from a uniform distribution.*
- static int **GetIntNormal** (int \_mean, int \_sigma)  
*Get a double from a normal distribution.*
- static int **GetIntUniform** (int \_min, int \_max)  
*Get a integer from a uniform distribution.*
- static uint32\_t **GetSeed** ()  
*Get the seed value.*
- static void **SetSeed** (uint32\_t \_seed)  
*Set the seed value.*

**10.120.1 Detailed Description**

Random number generator class.



## 10.120.2 Member Function Documentation

10.120.2.1 `static double gazebo::math::Rand::GetDbfNormal ( double _mean = 0, double _sigma = 1 ) [static]`

Get a double from a normal distribution.

### Parameters

in	<i>_mean</i>	Mean value for the distribution
in	<i>_sigma</i>	Sigma value for the distribution

10.120.2.2 `static double gazebo::math::Rand::GetDbfUniform ( double _min = 0, double _max = 1 ) [static]`

Get a double from a uniform distribution.

### Parameters

in	<i>_min</i>	Minimum bound for the random number
in	<i>_max</i>	Maximum bound for the random number

10.120.2.3 `static int gazebo::math::Rand::GetIntNormal ( int _mean, int _sigma ) [static]`

Get a double from a normal distribution.

### Parameters

in	<i>_mean</i>	Mean value for the distribution
in	<i>_sigma</i>	Sigma value for the distribution

10.120.2.4 `static int gazebo::math::Rand::GetIntUniform ( int _min, int _max ) [static]`

Get a integer from a uniform distribution.

### Parameters

in	<i>_min</i>	Minimum bound for the random number
in	<i>_max</i>	Maximum bound for the random number

10.120.2.5 `static uint32_t gazebo::math::Rand::GetSeed ( ) [static]`

Get the seed value.

### Returns

The seed value used to initialize the random number generator.

10.120.2.6 `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:

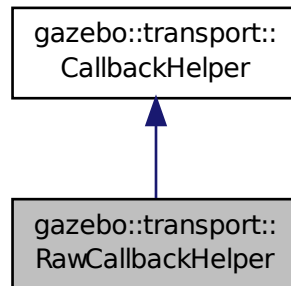
- **Rand.hh**

## 10.121 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

#### 10.121.1 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.

## 10.121.2 Constructor & Destructor Documentation

10.121.2.1 `gazebo::transport::RawCallbackHelper::RawCallbackHelper ( const boost::function< void(const std::string &)> & _cb, bool _latching = false ) [inline]`

Constructor.

### Parameters

in	<i>_cb</i>	boost function to call on incoming messages
in	<i>_latching</i>	Set to true to make the callback helper latching.

## 10.121.3 Member Function Documentation

10.121.3.1 `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. 155).

10.121.3.2 `virtual bool gazebo::transport::RawCallbackHelper::HandleData ( const std::string & _newdata ) [inline],[virtual]`

Process new incoming data.

### Parameters

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

### Returns

true if successfully processed; false otherwise

Implements `gazebo::transport::CallbackHelper` (p. 155).

10.121.3.3 `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. 155).

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

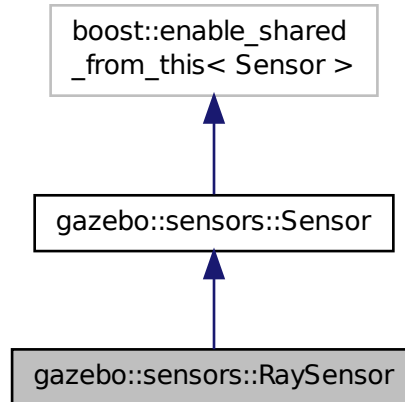
- `CallbackHelper.hh`

## 10.122 gazebo::sensors::RaySensor Class Reference

**Sensor** (p. 672) with one or more rays.

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

Inheritance diagram for gazebo::sensors::RaySensor:



### Public Member Functions

- **RaySensor** ()  
*Constructor.*
- virtual **~RaySensor** ()  
*Destructor.*
- **math::Angle GetAngleMax** () const  
*Get the maximum angle.*
- **math::Angle GetAngleMin** () const  
*Get the minimum angle.*
- double **GetAngleResolution** () const  
*Get the angle in radians between each range.*
- int **GetFiducial** (int \_index)  
*Get detected fiducial value for a ray.*
- **physics::MultiRayShapePtr GetLaserShape** () const  
*Returns a pointer to the internal **physics::MultiRayShape** (p. 507).*
- 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

#### 10.122.1 Detailed Description

**Sensor** (p. 672) 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).

#### 10.122.2 Constructor & Destructor Documentation

##### 10.122.2.1 gazebo::sensors::RaySensor::RaySensor ( )

Constructor.

10.122.2.2 `virtual gazebo::sensors::RaySensor::~~RaySensor ( ) [virtual]`

Destructor.

### 10.122.3 Member Function Documentation

10.122.3.1 `virtual void gazebo::sensors::RaySensor::Fini ( ) [protected],[virtual]`

Finalize the sensor.

Reimplemented from `gazebo::sensors::Sensor` (p. 676).

10.122.3.2 `math::Angle gazebo::sensors::RaySensor::GetAngleMax ( ) const`

Get the maximum angle.

#### Returns

the maximum angle object

10.122.3.3 `math::Angle gazebo::sensors::RaySensor::GetAngleMin ( ) const`

Get the minimum angle.

#### Returns

The minimum angle object

10.122.3.4 `double gazebo::sensors::RaySensor::GetAngleResolution ( ) const`

Get the angle in radians between each range.

#### Returns

Resolution of the angle

10.122.3.5 `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

### 10.122.3.6 `physics::MultiRayShapePtr gazebo::sensors::RaySensor::GetLaserShape ( ) const` [inline]

Returns a pointer to the internal `physics::MultiRayShape` (p. 507).

**Returns**

Pointer to ray shape

### 10.122.3.7 `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>_index</code>	Index of specific ray
-----------------	---------------------	-----------------------

**Returns**

Returns `DBL_MAX` for no detection.

### 10.122.3.8 `int gazebo::sensors::RaySensor::GetRangeCount ( ) const`

Get the range count.

**Returns**

The number of ranges

### 10.122.3.9 `double gazebo::sensors::RaySensor::GetRangeMax ( ) const`

Get the maximum range.

**Returns**

The maximum range

10.122.3.10 `double gazebo::sensors::RaySensor::GetRangeMin ( ) const`

Get the minimum range.

#### Returns

The minimum range

10.122.3.11 `double gazebo::sensors::RaySensor::GetRangeResolution ( ) const`

Get the range resolution.

#### Returns

Resolution of the range

10.122.3.12 `void gazebo::sensors::RaySensor::GetRanges ( std::vector< double > & _ranges )`

Get all the ranges.

#### Parameters

<code>_ranges</code>	A (p. 107) vector that will contain all the range data
----------------------	--

10.122.3.13 `int gazebo::sensors::RaySensor::GetRayCount ( ) const`

Get the ray count.

#### Returns

The number of rays

10.122.3.14 `double gazebo::sensors::RaySensor::GetRetro ( int _index )`

Get detected retro (intensity) value for a ray.

Warning: If you are accessing all the ray data in a loop it's possible that the Ray will update in the middle of your access loop. This means some data will come from one scan, and some from another scan. You can solve this problem by using `SetActive(false)` <your accessor loop> `SetActive(true)`.

#### Parameters

<code>in</code>	<code>_index</code>	Index of specific ray
-----------------	---------------------	-----------------------



**Returns**

Retro (intensity) value for ray

10.122.3.15 `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. 677).

10.122.3.16 `math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMax ( ) const`

Get the vertical scan line top angle.

**Returns**

The Maximum angle of the scan block

10.122.3.17 `math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMin ( ) const`

Get the vertical scan bottom angle.

**Returns**

The minimum angle of the scan block

10.122.3.18 `int gazebo::sensors::RaySensor::GetVerticalRangeCount ( ) const`

Get the vertical scan line count.

**Returns**

The number of scan lines vertically

10.122.3.19 `int gazebo::sensors::RaySensor::GetVerticalRayCount ( ) const`

Get the vertical scan line count.

**Returns**

The number of scan lines vertically

10.122.3.20 `virtual void gazebo::sensors::RaySensor::Init ( )` [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 678).

10.122.3.21 `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. 678).

10.122.3.22 `virtual void gazebo::sensors::RaySensor::Load ( const std::string & _worldName ) [virtual]`

Load the sensor with default parameters.

#### Parameters

in	<code>_worldName</code>	Name of world to load from.
----	-------------------------	-----------------------------

Reimplemented from `gazebo::sensors::Sensor` (p. 678).

10.122.3.23 `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. 679).

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

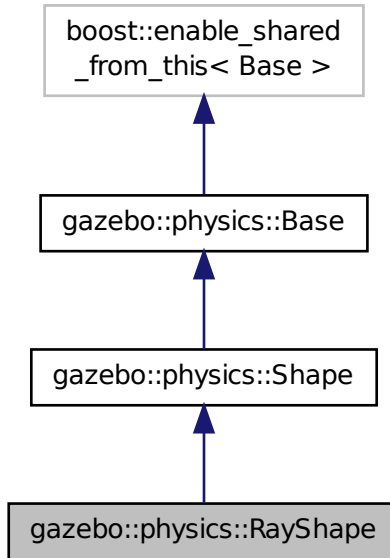
- `RaySensor.hh`

## 10.123 gazebo::physics::RayShape Class Reference

**Base** (p. 133) 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

#### 10.123.1 Detailed Description

**Base** (p. 133) class for Ray collision geometry.

#### 10.123.2 Constructor & Destructor Documentation

##### 10.123.2.1 gazebo::physics::RayShape::RayShape ( PhysicsEnginePtr \_physicsEngine ) [explicit]

Constructor for a global ray.

#### Parameters

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

10.123.2.2 `gazebo::physics::RayShape::RayShape ( CollisionPtr _parent ) [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	<b>Collision</b> (p. 190) parent of the shape.
----	----------------------	--

10.123.2.3 `virtual gazebo::physics::RayShape::~~RayShape ( ) [virtual]`

Destructor.

### 10.123.3 Member Function Documentation

10.123.3.1 `void gazebo::physics::RayShape::FillMsg ( msgs::Geometry & _msg ) [virtual]`

Fill a message with data from this object.

Parameters

out	<code>_msg</code>	Message to fill. Implement this function.
-----	-------------------	---

Implements **gazebo::physics::Shape** (p. 695).

10.123.3.2 `int gazebo::physics::RayShape::GetFiducial ( ) const`

Get the fiducial id detected by this ray.

Returns

Fiducial id detected.

10.123.3.3 `virtual void gazebo::physics::RayShape::GetGlobalPoints ( math::Vector3 & _posA, math::Vector3 & _posB ) [virtual]`

Get the global starting and ending points.

Parameters

out	<code>_posA</code>	Returns the starting point.
out	<code>_posB</code>	Returns the ending point.

10.123.3.4 `virtual void gazebo::physics::RayShape::GetIntersection ( double & _dist, std::string & _entity ) [pure virtual]`

Get the nearest intersection.

## Parameters

out	<code>_dist</code>	Distance to the intersection.
out	<code>_entity</code>	Name of the entity the ray intersected with.

10.123.3.5 `double gazebo::physics::RayShape::GetLength ( ) const`

Get the length of the ray.

## Returns

The ray length.

10.123.3.6 `virtual void gazebo::physics::RayShape::GetRelativePoints ( math::Vector3 & _posA, math::Vector3 & _posB ) [virtual]`

Get the relative starting and ending points.

## Parameters

in	<code>_posA</code>	Returns the starting point.
in	<code>_posB</code>	Returns the ending point.

10.123.3.7 `float gazebo::physics::RayShape::GetRetro ( ) const`

Get the retro-reflectivness detected by this ray.

## Returns

Retro reflectance value.

10.123.3.8 `virtual void gazebo::physics::RayShape::Init ( ) [virtual]`

In the ray.

Implements **`gazebo::physics::Shape`** (p. 695).

10.123.3.9 `virtual void gazebo::physics::RayShape::ProcessMsg ( const msgs::Geometry & _msg ) [virtual]`

Update this shape from a message.

## Parameters

in	<code>_msg</code>	Message to update from. Implement this function.
----	-------------------	--

Implements **`gazebo::physics::Shape`** (p. 695).

10.123.3.10 `void gazebo::physics::RayShape::SetFiducial ( int _fid )`

Set the fiducial id detected by this ray.

#### Parameters

in	<i>_fid</i>	Fiducial id detected by this ray.
----	-------------	-----------------------------------

10.123.3.11 `virtual void gazebo::physics::RayShape::SetLength ( double _len ) [virtual]`

Set the length of the ray.

#### Parameters

in	<i>_len</i>	Length of the array.
----	-------------	----------------------

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

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

#### Parameters

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

10.123.3.13 `void gazebo::physics::RayShape::SetRetro ( float _retro )`

Set the retro-reflectivness detected by this ray.

#### Parameters

in	<i>_retro</i>	Retro reflectance value.
----	---------------	--------------------------

10.123.3.14 `virtual void gazebo::physics::RayShape::Update ( ) [pure virtual]`

Update the ray collision.

Reimplemented from `gazebo::physics::Base` (p. 143).

## 10.123.4 Member Data Documentation

10.123.4.1 `int gazebo::physics::RayShape::contactFiducial [protected]`

Fiducial ID value.

10.123.4.2 `double gazebo::physics::RayShape::contactLen` [protected]

Length of the ray.

10.123.4.3 `double gazebo::physics::RayShape::contactRetro` [protected]

Retro reflectance value.

10.123.4.4 `math::Vector3 gazebo::physics::RayShape::globalEndPos` [protected]

End position of the ray in global cs.

10.123.4.5 `math::Vector3 gazebo::physics::RayShape::globalStartPos` [protected]

Start position of the ray in global cs.

10.123.4.6 `math::Vector3 gazebo::physics::RayShape::relativeEndPos` [protected]

End position of the ray, relative to the body.

10.123.4.7 `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**

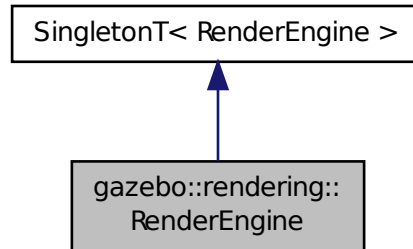
## 10.124 gazebo::rendering::RenderEngine Class Reference

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. 103) 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. 103). Load must happen before Init.*
- void **Load** ()  
*Load the parameters for **Ogre** (p. 103). 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

### 10.124.1 Detailed Description

Adaptor to Ogre3d.

Provides the interface to load, initialize the rendering engine.

### 10.124.2 Member Enumeration Documentation

#### 10.124.2.1 `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.

### 10.124.3 Member Function Documentation

#### 10.124.3.1 `void gazebo::rendering::RenderEngine::AddResourcePath ( const std::string & _uri )`

Add a new path for **Ogre** (p. 103) to search for resources.

Parameters

<code>in</code>	<code>_uri</code>	URI of the path. The uri should be of the form <code>file://</code> or <code>model://</code>
-----------------	-------------------	--

10.124.3.2 `ScenePtr gazebo::rendering::RenderEngine::CreateScene ( const std::string & _name, bool _enableVisualizations )`

Create a scene.

## Parameters

<code>in</code>	<code><i>_name</i></code>	The name of the scene.
<code>in</code>	<code><i>_enable-Visualizations</i></code>	True enables visualization elements such as laser lines.

10.124.3.3 `void gazebo::rendering::RenderEngine::Fini ( )`

Tears down the rendering engine.

10.124.3.4 `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`

10.124.3.5 `ScenePtr gazebo::rendering::RenderEngine::GetScene ( const std::string & _name )`

Get a scene by name.

## Parameters

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

## Returns

**A** (p. 107) pointer to the **Scene** (p. 651), or NULL if the scene doesn't exist.

10.124.3.6 `ScenePtr gazebo::rendering::RenderEngine::GetScene ( unsigned int _index )`

Get a scene by index.

The index should be between 0 and **GetSceneCount()** (p. 632).

## Parameters

<code>in</code>	<code><i>_index</i></code>	The index of the scene.
-----------------	----------------------------	-------------------------

## Returns

**A** (p. 107) pointer to a **Scene** (p. 651), or NULL if the index was invalid.

10.124.3.7 `unsigned int gazebo::rendering::RenderEngine::GetSceneCount ( ) const`

Get the number of scenes.

#### Returns

The number of scenes created by the **RenderEngine** (p. 628).

10.124.3.8 `void gazebo::rendering::RenderEngine::Init ( )`

Initialize **Ogre** (p. 103). Load must happen before Init.

10.124.3.9 `void gazebo::rendering::RenderEngine::Load ( )`

Load the parameters for **Ogre** (p. 103). Load must happen before Init.

10.124.3.10 `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.
-----------------	--------------------	----------------------------------

### 10.124.4 Member Data Documentation

10.124.4.1 `void* gazebo::rendering::RenderEngine::dummyContext` [protected]

GLX context used to render the scenes.Used for gui-less operation.

10.124.4.2 `void* gazebo::rendering::RenderEngine::dummyDisplay` [protected]

Pointer to the dummy display.Used for gui-less operation.

10.124.4.3 `uint64_t gazebo::rendering::RenderEngine::dummyWindowId` [protected]

ID for a dummy window. Used for gui-less operation.

10.124.4.4 `Ogre::Root* gazebo::rendering::RenderEngine::root`

Pointer to the root scene node.

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

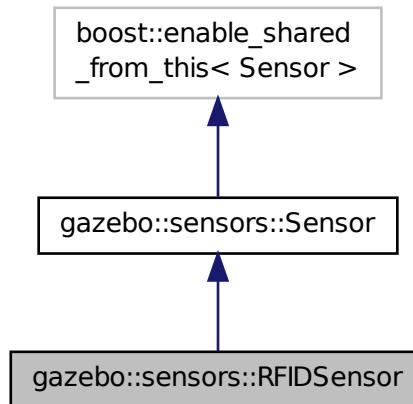
- **RenderEngine.hh**

## 10.125 gazebo::sensors::RFIDSensor Class Reference

**Sensor** (p. 672) 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

### 10.125.1 Detailed Description

**Sensor** (p. 672) class for RFID type of sensor.

### 10.125.2 Constructor & Destructor Documentation

#### 10.125.2.1 gazebo::sensors::RFIDSensor::RFIDSensor ( )

Constructor.

#### 10.125.2.2 virtual gazebo::sensors::RFIDSensor::~~RFIDSensor ( ) [virtual]

Destructor.

### 10.125.3 Member Function Documentation

#### 10.125.3.1 void gazebo::sensors::RFIDSensor::AddTag ( RFIDTag \* *tag* )

#### 10.125.3.2 virtual void gazebo::sensors::RFIDSensor::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 676).

#### 10.125.3.3 virtual void gazebo::sensors::RFIDSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 678).

#### 10.125.3.4 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 <b>Sensor</b> (p. 672) parameters.
in	<i>_worldName</i>	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 678).

#### 10.125.3.5 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. 678).

### 10.125.3.6 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. 679).

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

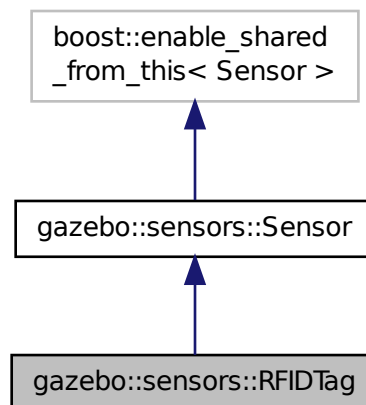
- **RFIDSensor.hh**

## 10.126 gazebo::sensors::RFIDTag Class Reference

**RFIDTag** (p. 635) to interact with RFIDTagSensors.

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

Inheritance diagram for gazebo::sensors::RFIDTag:



## Public Member Functions

- **RFIDTag** ()  
*Constructor.*
- virtual **~RFIDTag** ()  
*Destructor.*
- virtual void **Fini** ()  
*Finalize the sensor.*
- **math::Pose GetTagPose** () const  
*Returns pose of tag in world coordinate.*
- virtual void **Init** ()  
*Initialize the sensor.*
- virtual void **Load** (const std::string &\_worldName, **sdf::ElementPtr** &\_sdf)
- virtual void **Load** (const std::string &\_worldName)  
*Load the sensor with default parameters.*

## Protected Member Functions

- virtual void **UpdateImpl** (bool \_force)  
*This gets overwritten by derived sensor types.*

## Additional Inherited Members

### 10.126.1 Detailed Description

**RFIDTag** (p. 635) to interact with RFIDTagSensors.

### 10.126.2 Constructor & Destructor Documentation

#### 10.126.2.1 gazebo::sensors::RFIDTag::RFIDTag ( )

Constructor.

#### 10.126.2.2 virtual gazebo::sensors::RFIDTag::~~RFIDTag ( ) [virtual]

Destructor.

### 10.126.3 Member Function Documentation

#### 10.126.3.1 virtual void gazebo::sensors::RFIDTag::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 676).



10.126.3.2 `math::Pose gazebo::sensors::RFIDTag::GetTagPose ( ) const` `[inline]`

Returns pose of tag in world coordinate.

#### Returns

Pose of object.

References `gazebo::physics::Entity::GetWorldPose()`.

10.126.3.3 `virtual void gazebo::sensors::RFIDTag::Init ( )` `[virtual]`

Initialize the sensor.

Reimplemented from `gazebo::sensors::Sensor` (p. 678).

10.126.3.4 `virtual void gazebo::sensors::RFIDTag::Load ( const std::string & _worldName, sdf::ElementPtr & _sdf )`  
`[virtual]`

10.126.3.5 `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. 678).

10.126.3.6 `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. 679).

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

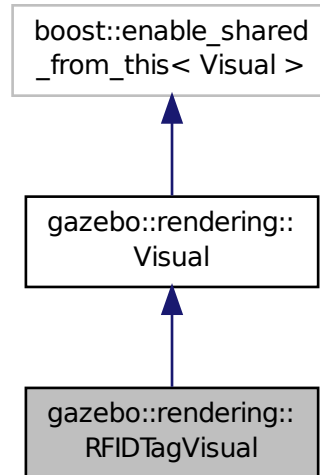
- `RFIDTag.hh`

## 10.127 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

### 10.127.1 Detailed Description

Visualization for RFID tags sensor.

### 10.127.2 Constructor & Destructor Documentation

10.127.2.1 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. 633)

10.127.2.2 virtual gazebo::rendering::RFIDTagVisual::~~RFIDTagVisual ( ) [virtual]

Destructor.

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

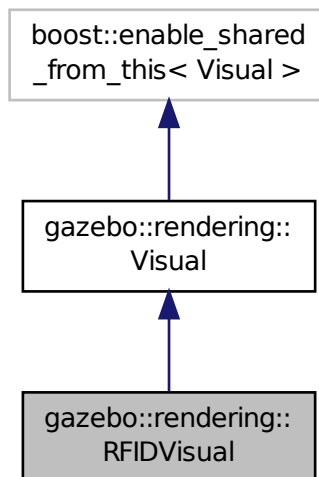
- **RFIDTagVisual.hh**

## 10.128 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

### 10.128.1 Detailed Description

Visualization for RFID sensor.

### 10.128.2 Constructor & Destructor Documentation

10.128.2.1 `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 <b>Visual</b> (p. 850).
<code>in</code>	<code><i>_vis</i></code>	Parent <b>Visual</b> (p. 850).
<code>in</code>	<code><i>_topicName</i></code>	Name of the topic which publishes RFID data.

10.128.2.2 `virtual gazebo::rendering::RFIDVisual::~RFIDVisual ( ) [virtual]`

Destructor.

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

- **RFIDVisual.hh**

## 10.129 Road Class Reference

Used to render a strip of road.

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

### 10.129.1 Detailed Description

Used to render a strip of road.

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

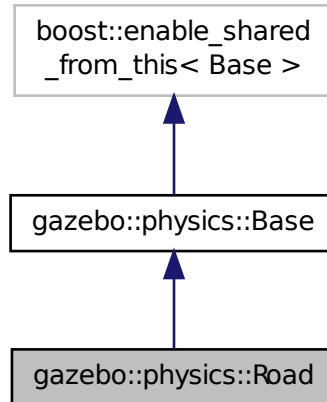
- **Road2d.hh**

## 10.130 gazebo::physics::Road Class Reference

for building a **Road** (p. 640) 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

### 10.130.1 Detailed Description

for building a **Road** (p. 640) from SDF

### 10.130.2 Constructor & Destructor Documentation

#### 10.130.2.1 gazebo::physics::Road::Road ( **BasePtr** \_parent ) [explicit]

Constructor.

#### Parameters

in	_parent	Parent of this road object.
----	---------	-----------------------------

10.130.2.2 virtual gazebo::physics::Road::~~Road ( ) [virtual]

Destructor.

### 10.130.3 Member Function Documentation

10.130.3.1 virtual void gazebo::physics::Road::Init ( ) [virtual]

Initialize the road.

Reimplemented from **gazebo::physics::Base** (p. 141).

10.130.3.2 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. 141).

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

- **Road.hh**

## 10.131 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.*

### 10.131.1 Constructor & Destructor Documentation

10.131.1.1 gazebo::rendering::Road2d::Road2d ( )

Constructor.

10.131.1.2 virtual gazebo::rendering::Road2d::~~Road2d ( ) [virtual]

Destructor.

## 10.131.2 Member Function Documentation

### 10.131.2.1 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**

## 10.132 gazebo::math::RotationSpline Class Reference

**Spline** (p. 725) 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*

### 10.132.1 Detailed Description

**Spline** (p. 725) for rotations.

### 10.132.2 Constructor & Destructor Documentation

#### 10.132.2.1 gazebo::math::RotationSpline::RotationSpline ( )

Constructor. Sets the autoCalc to true.

#### 10.132.2.2 gazebo::math::RotationSpline::~~RotationSpline ( )

Destructor. Nothing is done.

### 10.132.3 Member Function Documentation

#### 10.132.3.1 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
----	-----------------	---------------

#### 10.132.3.2 void gazebo::math::RotationSpline::Clear ( )

Clears all the points in the spline.

#### 10.132.3.3 unsigned int gazebo::math::RotationSpline::GetNumPoints ( ) const

Gets the number of control points in the spline.

#### Returns

the count

#### 10.132.3.4 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)

### 10.132.3.5 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

### 10.132.3.6 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

### 10.132.3.7 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.

**10.132.3.8 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.
----	------------------	--

**10.132.3.9 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

**10.132.4 Member Data Documentation****10.132.4.1 bool gazebo::math::RotationSpline::autoCalc [protected]**

Automatic recalculation of tangents when control points are updated.

**10.132.4.2 std::vector<Quaternion> gazebo::math::RotationSpline::points [protected]**

the control points

**10.132.4.3 std::vector<Quaternion> gazebo::math::RotationSpline::tangents [protected]**

the tangents

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

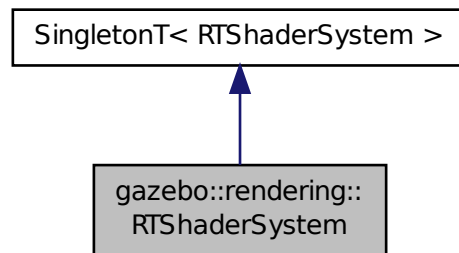
- [RotationSpline.hh](#)

## 10.133 gazebo::rendering::RTShaderSystem Class Reference

Implements **Ogre** (p. 103)'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. 103) 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

#### 10.133.1 Detailed Description

Implements **Ogre** (p. 103)'s Run-Time Shader system.

This class allows Gazebo to generate per-pixel shaders for every material at run-time.

#### 10.133.2 Member Enumeration Documentation

##### 10.133.2.1 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.

#### 10.133.3 Member Function Documentation

##### 10.133.3.1 void gazebo::rendering::RTShaderSystem::AddScene ( **ScenePtr** \_scene )

Add a scene manager.

## Parameters

in	<code>_scene</code>	The scene to process
----	---------------------	----------------------

10.133.3.2 `void gazebo::rendering::RTShaderSystem::ApplyShadows ( ScenePtr _scene )`

Apply shadows to a scene.

## Parameters

in	<code>_scene</code>	The scene to receive shadows.
----	---------------------	-------------------------------

10.133.3.3 `void gazebo::rendering::RTShaderSystem::AttachEntity ( Visual * vis )`

Set an Ogre::Entity to use RT shaders.

## Parameters

in	<code>_vis</code>	<b>Visual</b> (p. 850) that will use the <b>RTShaderSystem</b> (p. 647).
----	-------------------	--

10.133.3.4 `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.

10.133.3.5 `void gazebo::rendering::RTShaderSystem::Clear ( )`

Clear the shader system.

10.133.3.6 `void gazebo::rendering::RTShaderSystem::DetachEntity ( Visual * _vis )`

Remove and entity.

## Parameters

in	<code>_vis</code>	Remove this visual.
----	-------------------	---------------------

10.133.3.7 `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.

10.133.3.8 void gazebo::rendering::RTShaderSystem::Fini ( )

Finalize the shader system.

10.133.3.9 void gazebo::rendering::RTShaderSystem::GenerateShaders ( Visual \* *\_vis* )

Generate shaders for an entity.

## Parameters

in	<i>_vis</i>	The visual to generate shaders for.
----	-------------	-------------------------------------

10.133.3.10 Ogre::PSSMShadowCameraSetup\* gazebo::rendering::RTShaderSystem::GetPSSMShadowCameraSetup ( ) const

Get the **Ogre** (p. 103) PSSM Shadows camera setup.

## Returns

The **Ogre** (p. 103) PSSM Shadows camera setup.

10.133.3.11 void gazebo::rendering::RTShaderSystem::Init ( )

Init the run time shader system.

10.133.3.12 void gazebo::rendering::RTShaderSystem::RemoveScene ( ScenePtr *\_scene* )

Remove a scene.

## Parameters

in	<i>The</i>	scene to remove
----	------------	-----------------

10.133.3.13 void gazebo::rendering::RTShaderSystem::RemoveShadows ( ScenePtr *\_scene* )

Remove shadows from a scene.

## Parameters

in	<i>_scene</i>	The scene to remove shadows from.
----	---------------	-----------------------------------

10.133.3.14 void gazebo::rendering::RTShaderSystem::SetPerPixelLighting ( bool *\_set* )

Set the lighting model to per pixel or per vertex.

#### Parameters

<i>in</i>	<i>_set</i>	True means to use per-pixel shaders.
-----------	-------------	--------------------------------------

10.133.3.15 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:

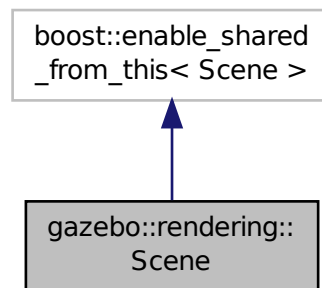
- **RTShaderSystem.hh**

## 10.134 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. 651).

- **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.*
- **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. 651).*
- `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.*

### 10.134.1 Detailed Description

Representation of an entire scene graph.

Maintains all the Visuals, Lights, and Cameras for a World.

### 10.134.2 Constructor & Destructor Documentation

10.134.2.1 gazebo::rendering::Scene::Scene ( const std::string & \_name, bool \_enableVisualizations = false )

Constructor.

#### Parameters

in	<code>_name</code>	Name of the scene.
in	<code>_enable- Visualizations</code>	True to enable visualizations, this should be set to true for user interfaces, and false for sensor generation.

10.134.2.2 virtual gazebo::rendering::Scene::~Scene ( ) [virtual]

Destructor.

### 10.134.3 Member Function Documentation

10.134.3.1 `void gazebo::rendering::Scene::AddVisual ( VisualPtr _vis )`

Add a visual to the scene.

Parameters

<code>in</code>	<code>_vis</code>	<b>Visual</b> (p. 850) to add.
-----------------	-------------------	--------------------------------

10.134.3.2 `void gazebo::rendering::Scene::Clear ( )`

Clear **rendering::Scene** (p. 651).

10.134.3.3 `VisualPtr gazebo::rendering::Scene::CloneVisual ( const std::string & _visualName, const std::string & _newName )`

Clone a visual.

Parameters

<code>in</code>	<code>_visualName</code>	Name of the visual to clone.
<code>in</code>	<code>_newName</code>	New name of the visual.

Returns

Pointer to the cloned visual.

10.134.3.4 `CameraPtr gazebo::rendering::Scene::CreateCamera ( const std::string & _name, bool _autoRender = true )`

Create a camera.

Parameters

<code>in</code>	<code>_name</code>	Name of the new camera.
<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 camera.

10.134.3.5 `DepthCameraPtr gazebo::rendering::Scene::CreateDepthCamera ( const std::string & _name, bool _autoRender = true )`

Create depth camera.

Parameters

<code>in</code>	<code>_name</code>	Name of the new camera.
<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 camera.

10.134.3.6 `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><i>_cellCount</i></code>	Number of grid cells in one direction.
<code>in</code>	<code><i>_cellLength</i></code>	Length of one grid cell.
<code>in</code>	<code><i>_lineWidth</i></code>	Width of the grid lines.
<code>in</code>	<code><i>_color</i></code>	Color of the grid lines.

10.134.3.7 `UserCameraPtr gazebo::rendering::Scene::CreateUserCamera ( const std::string & _name )`

Create a user camera.

**A** (p. 107) user camera is one design for use with a GUI.

**Parameters**

<code>in</code>	<code><i>_name</i></code>	Name of the <b>UserCamera</b> (p. 796).
-----------------	---------------------------	---

**Returns**

**A** (p. 107) pointer to the new **UserCamera** (p. 796).

10.134.3.8 `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><i>_start</i></code>	Start position of the line.
<code>in</code>	<code><i>_end</i></code>	End position of the line.
<code>in</code>	<code><i>_name</i></code>	Name of the line.

10.134.3.9 `common::Color gazebo::rendering::Scene::GetAmbientColor ( ) const`

Get the ambient color.

**Returns**

The scene's ambient color.

10.134.3.10 `common::Color gazebo::rendering::Scene::GetBackgroundColor ( ) const`

Get the background color.

## Returns

The background color.

10.134.3.11 `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. 657).

## 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.

10.134.3.12 `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.

10.134.3.13 `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.

10.134.3.14 `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.

10.134.3.15 `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. 658).

#### Parameters

in	<code>_index</code>	Index of the grid.
----	---------------------	--------------------

10.134.3.16 `uint32_t gazebo::rendering::Scene::GetGridCount ( ) const`

Get the number of grids.

#### Returns

The number of grids.

10.134.3.17 `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.

10.134.3.18 `Heightmap* gazebo::rendering::Scene::GetHeightmap ( ) const`

Get a pointer to the heightmap.

**Returns**

Pointer to the heightmap, NULL if no heightmap.

10.134.3.19 `uint32_t gazebo::rendering::Scene::GetId ( ) const`

Get the scene ID.

**Returns**

The ID of the scene.

10.134.3.20 `std::string gazebo::rendering::Scene::GetIdString ( ) const`

Get the scene Id as a string.

**Returns**

The ID as a string.

10.134.3.21 `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.

10.134.3.22 `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. 659).

**Parameters**

<code>in</code>	<code>_index</code>	Index of the light.
-----------------	---------------------	---------------------

**Returns**

Pointer to the **Light** (p. 397) or NULL if index was invalid.

10.134.3.23 `uint32_t gazebo::rendering::Scene::GetLightCount ( ) const`

Get the count of the lights.

**Returns**

The number of lights.

#### 10.134.3.24 `Ogre::SceneManager* gazebo::rendering::Scene::GetManager ( ) const`

Get the OGRE scene manager.

**Returns**

Pointer to the **Ogre** (p. 103) SceneManager.

#### 10.134.3.25 `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.

#### 10.134.3.26 `std::string gazebo::rendering::Scene::GetName ( ) const`

Get the name of the scene.

**Returns**

Name of the scene.

#### 10.134.3.27 `VisualPtr gazebo::rendering::Scene::GetSelectedVisual ( ) const`

Get the currently selected visual.

**Returns**

Pointer to the currently selected visual, or NULL if nothing is selected.

#### 10.134.3.28 `bool gazebo::rendering::Scene::GetShadowsEnabled ( ) const`

Get whether shadows are on or off.

**Returns**

True if shadows are enabled.



### 10.134.3.29 `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. 661).

#### Parameters

in	_index	Index of the <code>UserCamera</code> (p. 796) to get.
----	--------	---

#### Returns

Pointer to the `UserCamera` (p. 796), or NULL if the index was invalid.

### 10.134.3.30 `uint32_t gazebo::rendering::Scene::GetUserCameraCount ( ) const`

Get the number of user cameras in this scene.

#### Returns

The number of user cameras.

### 10.134.3.31 `VisualPtr gazebo::rendering::Scene::GetVisual ( const std::string & _name ) const`

Get a visual by name.

### 10.134.3.32 `VisualPtr gazebo::rendering::Scene::GetVisualAt ( CameraPtr _camera, const math::Vector2i & _mousePos, std::string & _mod )`

Get an entity at a pixel location using a camera.

Used for mouse picking.

#### Parameters

in	_camera	The ogre camera, used to do mouse picking
in	_mousePos	The position of the mouse in screen coordinates
out	_mod	Used for object manipulation

#### Returns

The selected entity, or NULL

### 10.134.3.33 `VisualPtr gazebo::rendering::Scene::GetVisualAt ( CameraPtr _camera, const math::Vector2i & _mousePos )`

Get a visual at a mouse position.

#### Parameters

in	_camera	Pointer to the camera used to project the mouse position.
in	_mousePos	The 2d position of the mouse in pixels.

**Returns**

Pointer to the visual, NULL if none found.

#### 10.134.3.34 **VisualPtr** gazebo::rendering::Scene::GetVisualBelow ( const std::string & *\_visualName* )

Get the closest visual below a given visual.

**Parameters**

in	<i>_visualName</i>	Name of the visual to search below.
----	--------------------	-------------------------------------

**Returns**

Pointer to the visual below, or NULL if no visual.

#### 10.134.3.35 void gazebo::rendering::Scene::GetVisualsBelowPoint ( const math::Vector3 & *\_pt*, std::vector< **VisualPtr** > & *\_visuals* )

Get a visual directly below a point.

**Parameters**

in	<i>_pt</i>	3D point to get the visual below.
out	<i>_visuals</i>	The visuals below the point order in proximity.

#### 10.134.3.36 **VisualPtr** gazebo::rendering::Scene::GetWorldVisual ( ) const

Get the top level world visual.

**Returns**

Pointer to the world visual.

#### 10.134.3.37 void gazebo::rendering::Scene::Init ( )

Init **rendering::Scene** (p. 651).

#### 10.134.3.38 void gazebo::rendering::Scene::Load ( sdf::ElementPtr *\_scene* )

Load the scene from a set of parameters.

**Parameters**

in	<i>_scene</i>	SDF scene element to load.
----	---------------	----------------------------

10.134.3.39 void gazebo::rendering::Scene::Load ( )

Load the scene with default parameters.

10.134.3.40 void gazebo::rendering::Scene::PreRender ( )

Process all received messages.

10.134.3.41 void gazebo::rendering::Scene::PrintSceneGraph ( )

Print the scene graph to std\_out.

10.134.3.42 void gazebo::rendering::Scene::RemoveVisual ( VisualIPtr \_vis )

Remove a visual from the scene.

#### Parameters

in	_vis	<b>Visual</b> (p. 850) to remove.
----	------	-----------------------------------

10.134.3.43 void gazebo::rendering::Scene::SelectVisual ( const std::string & \_name, const std::string & \_mode )

Select a visual by name.

#### Parameters

in	_name	Name of the visual to select.
in	_mode	Selection mode (normal, or move).

10.134.3.44 void gazebo::rendering::Scene::SetAmbientColor ( const common::Color & \_color )

Set the ambient color.

#### Parameters

in	_color	The ambient color to use.
----	--------	---------------------------

10.134.3.45 void gazebo::rendering::Scene::SetBackgroundColor ( const common::Color & \_color )

Set the background color.

#### Parameters

in	_color	The background color.
----	--------	-----------------------

10.134.3.46 `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	<i>_type</i>	Type of fog: "linear", "exp", or "exp2".
in	<i>_color</i>	Color of the fog.
in	<i>_density</i>	Fog density.
in	<i>_start</i>	Distance from camera to start the fog.
in	<i>_end</i>	Distance from camera at which the fog is at max density.

10.134.3.47 `void gazebo::rendering::Scene::SetGrid ( bool _enabled )`

Set the grid on or off.

#### Parameters

in	<i>_enabled</i>	Set to true to turn on the grid
----	-----------------	---------------------------------

10.134.3.48 `void gazebo::rendering::Scene::SetShadowsEnabled ( bool _value )`

Set whether shadows are on or off.

#### Parameters

in	<i>_value</i>	True to enable shadows, False to disable
----	---------------	--

10.134.3.49 `void gazebo::rendering::Scene::SetTransparent ( bool _show )`

Enable or disable transparency for all visuals.

#### Parameters

in	<i>_show</i>	True to enable transparency for all visuals.
----	--------------	--

10.134.3.50 `void gazebo::rendering::Scene::SetVisible ( const std::string & _name, bool _visible )`

Hide or show a visual.

#### Parameters

in	<i>_name</i>	Name of the visual to change.
in	<i>_visible</i>	True to make visual visible, False to make it invisible.

10.134.3.51 void gazebo::rendering::Scene::ShowCollisions ( bool *\_show* )

Enable or disable collision visualization.

Parameters

in	<i>_show</i>	True to enable collision visualization.
----	--------------	---

10.134.3.52 void gazebo::rendering::Scene::ShowCOMs ( bool *\_show* )

Enable or disable center of mass visualization.

Parameters

in	<i>_show</i>	True to enable center of mass visualization.
----	--------------	--

10.134.3.53 void gazebo::rendering::Scene::ShowContacts ( bool *\_show* )

Enable or disable contact visualization.

Parameters

in	<i>_show</i>	True to enable contact visualization.
----	--------------	---------------------------------------

10.134.3.54 void gazebo::rendering::Scene::ShowJoints ( bool *\_show* )

Enable or disable joint visualization.

Parameters

in	<i>_show</i>	True to enable joint visualization.
----	--------------	-------------------------------------

10.134.3.55 void gazebo::rendering::Scene::SnapVisualToNearestBelow ( const std::string & *\_visualName* )

Move the visual to be ontop of the nearest visual below it.

Parameters

in	<i>_visualName</i>	Name of the visual to move.
----	--------------------	-----------------------------

10.134.3.56 std::string gazebo::rendering::Scene::StripSceneName ( const std::string & *\_name* ) const

Remove the name of scene from a string.

Parameters

in	<i>_name</i>	Name to string the scene name from.
----	--------------	-------------------------------------

**Returns**

The stripped name.

**10.134.4 Member Data Documentation****10.134.4.1 SkyX::SkyX\* gazebo::rendering::Scene::skyx**

Pointer to the sky.

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

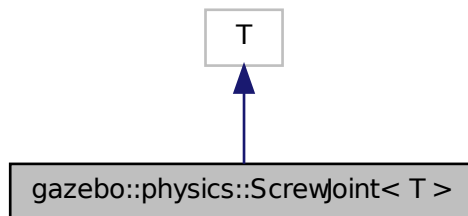
- **Scene.hh**

**10.135 gazebo::physics::ScrewJoint< T > Class Template Reference**

**A** (p. 107) 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 void **Load** (**sdf::ElementPtr** \_sdf)  
*Load a **ScrewJoint** (p. 666).*
- 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.*

## 10.135.1 Detailed Description

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

**A** (p. 107) screw joint, which has both prismatic and rotational DOFs.

## 10.135.2 Constructor & Destructor Documentation

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

Constructor.

### Parameters

<i>in</i>	<i>_parent</i>	Parent of the joint.
-----------	----------------	----------------------

References gazebo::physics::Base::SCREW\_JOINT.

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

Destructor.

## 10.135.3 Member Function Documentation

10.135.3.1 `template<class T > math::Vector3 gazebo::physics::ScrewJoint< T >::GetAnchor ( int _index ) const [virtual]`

Get the anchor.

### Parameters

<i>in</i>	<i>_index</i>	Index of the axis. Not Used.
-----------	---------------	------------------------------

### Returns

Anchor for the joint.

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

10.135.3.3 `template<class T > virtual void gazebo::physics::ScrewJoint< T >::Load ( sdf::ElementPtr _sdf )`  
`[inline], [virtual]`

Load a **ScrewJoint** (p. 666).

#### Parameters

in	<code>_sdf</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`.

10.135.3.4 `template<class T > void gazebo::physics::ScrewJoint< T >::SetAnchor ( int _index, const math::Vector3 & _anchor )` `[virtual]`

Set the anchor.

#### Parameters

in	<code>_index</code>	Index of the axis. Not Used.
in	<code>_anchor</code>	Anchor value for the joint.

10.135.3.5 `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.

### 10.135.4 Member Data Documentation

10.135.4.1 `template<class T > math::Vector3 gazebo::physics::ScrewJoint< T >::fakeAnchor` `[protected]`

The anchor value is not used internally.

10.135.4.2 `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**



## 10.136 sdf::SDF Class Reference

Base **SDF** (p. 669) 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. 669) 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**

#### 10.136.1 Detailed Description

Base **SDF** (p. 669) class.

#### 10.136.2 Constructor & Destructor Documentation

10.136.2.1 sdf::SDF::SDF ( )

#### 10.136.3 Member Function Documentation

10.136.3.1 void sdf::SDF::PrintDescription ( )

10.136.3.2 void sdf::SDF::PrintDoc ( )

10.136.3.3 void sdf::SDF::PrintValues ( )

10.136.3.4 void sdf::SDF::PrintWiki ( )

10.136.3.5 void sdf::SDF::SetFromString ( const std::string & *\_sdfData* )

Set **SDF** (p. 669) values from a string.

10.136.3.6 `std::string sdf::SDF::ToString ( ) const`

10.136.3.7 `void sdf::SDF::Write ( const std::string & _filename )`

## 10.136.4 Member Data Documentation

10.136.4.1 `ElementPtr sdf::SDF::root`

10.136.4.2 `std::string sdf::SDF::version [static]`

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

- **SDF.hh**

## 10.137 gazebo::rendering::SelectionObj Class Reference

**A** (p. 107) 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. 670) object.*
- `std::string` **GetVisualName** () const  
*Get the name of the visual the selection obj is attached to.*
- void **Init** ()  
*Initialize the **rendering::SelectionObj** (p. 670) 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.*

### 10.137.1 Detailed Description

**A** (p. 107) graphical selection object.

Used to draw a visual around a selected object.

## 10.137.2 Constructor & Destructor Documentation

### 10.137.2.1 gazebo::rendering::SelectionObj::SelectionObj ( Scene \* *\_scene* )

Constructor.

#### Parameters

<i>in</i>	<i>_scene</i>	<b>Scene</b> (p. 651) to use.
-----------	---------------	-------------------------------

### 10.137.2.2 virtual gazebo::rendering::SelectionObj::~~SelectionObj ( ) [virtual]

Destructor.

## 10.137.3 Member Function Documentation

### 10.137.3.1 void gazebo::rendering::SelectionObj::Attach ( VisualIPtr *\_visual* )

Set the position of the node.

#### Parameters

<i>in</i>	<i>This</i>	draws the selection object around the passed in visual.
-----------	-------------	---

### 10.137.3.2 void gazebo::rendering::SelectionObj::Clear ( )

Clear the **rendering::SelectionObj** (p. 670) object.

### 10.137.3.3 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.

### 10.137.3.4 void gazebo::rendering::SelectionObj::Init ( )

Initialize the **rendering::SelectionObj** (p. 670) object.

### 10.137.3.5 bool gazebo::rendering::SelectionObj::IsActive ( ) const

Return true if the user is move the selection obj.

#### Returns

True if something is selected.

10.137.3.6 void gazebo::rendering::SelectionObj::SetActive ( bool *\_active* )

Set true if the user is moving the selection obj.

#### Parameters

in	<i>_active</i>	True if the user is interacting with the selection object.
----	----------------	--

10.137.3.7 void gazebo::rendering::SelectionObj::SetHighlight ( const std::string & *\_mod* )

Highlight the selection object based on a modifier.

#### Parameters

in	<i>_mod</i>	Modifier used when highlighting the selection object.
----	-------------	---

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

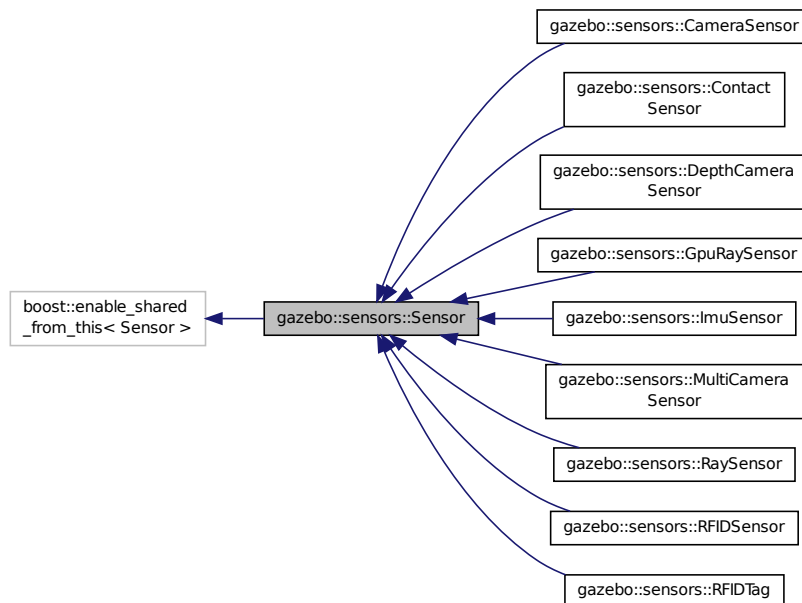
- SelectionObj.hh

## 10.138 gazebo::sensors::Sensor Class Reference

Base class for sensors.

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

Inheritance diagram for gazebo::sensors::Sensor:



## Public Member Functions

- **Sensor** ()  
*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.*
- **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.*
- 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. 679).*
- **gazebo::physics::WorldPtr world**  
*Pointer to the world.*

### 10.138.1 Detailed Description

Base class for sensors.

## 10.138.2 Constructor & Destructor Documentation

### 10.138.2.1 gazebo::sensors::Sensor::Sensor ( )

Constructor.

### 10.138.2.2 virtual gazebo::sensors::Sensor::~Sensor ( ) [virtual]

Destructor.

## 10.138.3 Member Function Documentation

### 10.138.3.1 template<typename T > event::ConnectionPtr gazebo::sensors::Sensor::ConnectUpdated ( T *\_subscriber* ) [inline]

Connect a signal that is triggered when the sensor is updated.

#### Parameters

<i>in</i>	<i>_subscriber</i>	Callback that receives the signal.
-----------	--------------------	------------------------------------

#### Returns

**A** (p. 107) pointer to the connection. This must be kept in scope.

#### See Also

**Sensor::DisconnectUpdated** (p. 675)

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

### 10.138.3.2 void gazebo::sensors::Sensor::DisconnectUpdated ( event::ConnectionPtr & *\_c* ) [inline]

Disconnect from a the updated signal.

#### Parameters

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

#### See Also

**Sensor::ConnectUpdated** (p. 675)

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

### 10.138.3.3 void gazebo::sensors::Sensor::FillMsg ( msgs::Sensor & *\_msg* )

fills a msgs::Sensor message.

## Parameters

out	<code>_msg</code>	Message to fill.
-----	-------------------	------------------

## 10.138.3.4 virtual void gazebo::sensors::Sensor::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented in **gazebo::sensors::MultiCameraSensor** (p. 504), **gazebo::sensors::CameraSensor** (p. 184), **gazebo::sensors::GpuRaySensor** (p. 324), **gazebo::sensors::ContactSensor** (p. 236), **gazebo::sensors::DepthCameraSensor** (p. 252), **gazebo::sensors::RFIDSensor** (p. 634), **gazebo::sensors::RaySensor** (p. 618), **gazebo::sensors::RFIDTag** (p. 636), and **gazebo::sensors::ImuSensor** (p. 359).

## 10.138.3.5 common::Time gazebo::sensors::Sensor::GetLastMeasurementTime ( )

Return last measurement time.

## Returns

Time of last measurement.

## 10.138.3.6 common::Time gazebo::sensors::Sensor::GetLastUpdateTime ( )

Return last update time.

## Returns

Time of last update.

## 10.138.3.7 std::string gazebo::sensors::Sensor::GetName ( ) const

Get name.

## Returns

Name of sensor.

## 10.138.3.8 std::string gazebo::sensors::Sensor::GetParentName ( ) const

Returns the name of the sensor parent.

The parent name is set by **Sensor::SetParent** (p. 679).

## Returns

Name of Parent.



10.138.3.9 `virtual math::Pose gazebo::sensors::Sensor::GetPose ( ) const [virtual]`

Get the current pose.

Returns

Current pose of the sensor.

10.138.3.10 `std::string gazebo::sensors::Sensor::GetScopedName ( ) const`

Get fully scoped name of the sensor.

Returns

world\_name::parent\_name::sensor\_name.

10.138.3.11 `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. 621), **`gazebo::sensors::CameraSensor`** (p. 185), and **`gazebo::sensors::MultiCameraSensor`** (p. 506).

10.138.3.12 `std::string gazebo::sensors::Sensor::GetType ( ) const`

Get sensor type.

Returns

Type of sensor.

10.138.3.13 `double gazebo::sensors::Sensor::GetUpdateRate ( )`

Get the update rate of the sensor.

Returns

\_hz update rate of sensor. Returns 0 if unthrottled.

10.138.3.14 `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.

10.138.3.15 `std::string gazebo::sensors::Sensor::GetWorldName ( ) const`

Returns the name of the world the sensor is in.

#### Returns

Name of the world.

10.138.3.16 `virtual void gazebo::sensors::Sensor::Init ( ) [virtual]`

Initialize the sensor.

Reimplemented in **`gazebo::sensors::GpuRaySensor`** (p. 329), **`gazebo::sensors::ContactSensor`** (p. 237), **`gazebo::sensors::CameraSensor`** (p. 185), **`gazebo::sensors::DepthCameraSensor`** (p. 252), **`gazebo::sensors::RFID-Sensor`** (p. 634), **`gazebo::sensors::RaySensor`** (p. 621), **`gazebo::sensors::RFIDTag`** (p. 637), **`gazebo::sensors::MultiCameraSensor`** (p. 506), and **`gazebo::sensors::ImuSensor`** (p. 359).

10.138.3.17 `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. 622), **`gazebo::sensors::CameraSensor`** (p. 186), and **`gazebo::sensors::ContactSensor`** (p. 237).

10.138.3.18 `virtual void gazebo::sensors::Sensor::Load ( const std::string & _worldName, sdf::ElementPtr _sdf ) [virtual]`

Load the sensor with SDF parameters.

#### Parameters

<code>in</code>	<code>_sdf</code>	SDF <b><code>Sensor</code></b> (p. 672) parameters.
<code>in</code>	<code>_worldName</code>	Name of world to load from.

Reimplemented in **`gazebo::sensors::ContactSensor`** (p. 238), **`gazebo::sensors::CameraSensor`** (p. 186), **`gazebo::sensors::RFIDSensor`** (p. 634), and **`gazebo::sensors::ImuSensor`** (p. 360).

10.138.3.19 `virtual void gazebo::sensors::Sensor::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 in **`gazebo::sensors::GpuRaySensor`** (p. 329), **`gazebo::sensors::ContactSensor`** (p. 238), **`gazebo::sensors::CameraSensor`** (p. 186), **`gazebo::sensors::DepthCameraSensor`** (p. 253), **`gazebo::sensors::RFID-Sensor`** (p. 634), **`gazebo::sensors::RaySensor`** (p. 622), **`gazebo::sensors::RFIDTag`** (p. 637), **`gazebo::sensors-`**

::MultiCameraSensor (p. 506), and gazebo::sensors::ImuSensor (p. 360).

10.138.3.20 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. 253).

10.138.3.21 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. 186), and gazebo::sensors::DepthCameraSensor (p. 253).

10.138.3.22 void gazebo::sensors::Sensor::SetUpdateRate ( double *\_hz* )

Set the update rate of the sensor.

#### Parameters

in	<i>_hz</i>	update rate of sensor.
----	------------	------------------------

10.138.3.23 void gazebo::sensors::Sensor::Update ( bool *\_force* )

Update the sensor.

#### Parameters

in	<i>_force</i>	True to force update, false otherwise.
----	---------------	--

10.138.3.24 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. 507), [gazebo::sensors::CameraSensor](#) (p. 187), [gazebo::sensors::GpuRaySensor](#) (p. 330), [gazebo::sensors::ContactSensor](#) (p. 238), [gazebo::sensors::DepthCameraSensor](#) (p. 253), [gazebo::sensors::RFIDSensor](#) (p. 635), [gazebo::sensors::RaySensor](#) (p. 622), [gazebo::sensors::RFIDTag](#) (p. 637), and [gazebo::sensors::ImuSensor](#) (p. 360).

#### 10.138.4 Member Data Documentation

10.138.4.1 `bool gazebo::sensors::Sensor::active` [protected]

True if sensor generation is active.

10.138.4.2 `std::vector<event::ConnectionPtr> gazebo::sensors::Sensor::connections` [protected]

All event connections.

10.138.4.3 `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.

10.138.4.4 `common::Time gazebo::sensors::Sensor::lastUpdateTime` [protected]

Time of the last update.

10.138.4.5 `transport::NodePtr gazebo::sensors::Sensor::node` [protected]

Node for communication.

10.138.4.6 `std::string gazebo::sensors::Sensor::parentName` [protected]

Name of the parent.

10.138.4.7 `std::vector<SensorPluginPtr> gazebo::sensors::Sensor::plugins` [protected]

All the plugins for the sensor.

10.138.4.8 `math::Pose gazebo::sensors::Sensor::pose` [protected]

Pose of the sensor.

10.138.4.9 `transport::SubscriberPtr gazebo::sensors::Sensor::poseSub` [protected]

Subscribe to pose updates.

10.138.4.10 `sdf::ElementPtr gazebo::sensors::Sensor::sdf` [protected]

Pointer to the SDF element for the sensor.

10.138.4.11 `common::Time gazebo::sensors::Sensor::updatePeriod` [protected]

Desired time between updates, set indirectly by `Sensor::SetUpdateRate` (p. 679).

10.138.4.12 `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`

## 10.139 SensorFactor Class Reference

The sensor factory; the class is just for namespacing purposes.

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

### 10.139.1 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`

## 10.140 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).*

### 10.140.1 Member Function Documentation

10.140.1.1 `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
---------------	--

10.140.1.2 `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. 672)

10.140.1.3 `static void gazebo::sensors::SensorFactory::RegisterAll ( ) [static]`

Register all known sensors.

- **sensors::CameraSensor** (p. 183)
- **sensors::DepthCameraSensor** (p. 250)
- **sensors::GpuRaySensor** (p. 320)
- **sensors::RaySensor** (p. 616)
- **sensors::ContactSensor** (p. 234)
- **sensors::RFIDSensor** (p. 633)
- **sensors::RFIDTag** (p. 635)

10.140.1.4 `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:

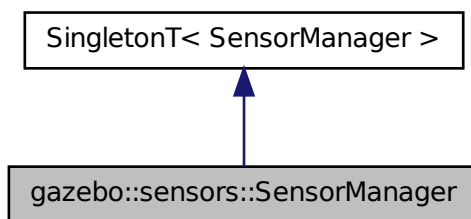
- **SensorFactory.hh**

## 10.141 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)`  
*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 Run ()`  
*Run the sensor manager update in a new thread.*
- `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

### 10.141.1 Detailed Description

Class to manage and update all sensors.

### 10.141.2 Member Function Documentation

10.141.2.1 `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><i>_elem</i></code>	The SDF element that describes the sensor
<code>in</code>	<code><i>_worldName</i></code>	Name of the world in which to create the sensor
<code>in</code>	<code><i>_parentName</i></code>	The name of the parent link which the sensor is attached to.

#### Returns

The name of the sensor

10.141.2.2 `void gazebo::sensors::SensorManager::Fini ( )`

Finalize all the sensors.

10.141.2.3 `SensorPtr gazebo::sensors::SensorManager::GetSensor ( const std::string & _name )`

Get a sensor.

#### Parameters

<code>in</code>	<code><i>_name</i></code>	The name of a sensor to find.
-----------------	---------------------------	-------------------------------



**Returns**

**A** (p. 107) pointer to the sensor. NULL if not found.

10.141.2.4 **Sensor\_V** gazebo::sensors::SensorManager::GetSensors ( ) const

Get all the sensors.

**Returns**

Vector of all the sensors.

## 10.141.2.5 void gazebo::sensors::SensorManager::GetSensorTypes ( std::vector&lt; std::string &gt; &amp; \_types ) const

Get all the sensor types.

**Parameters**

out	<i>All</i>	the sensor types.
-----	------------	-------------------

## 10.141.2.6 void gazebo::sensors::SensorManager::Init ( )

Init all the sensors.

## 10.141.2.7 void gazebo::sensors::SensorManager::RemoveSensor ( const std::string &amp; \_name )

Remove a sensor.

**Parameters**

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

## 10.141.2.8 void gazebo::sensors::SensorManager::RemoveSensors ( )

Remove all sensors.

## 10.141.2.9 void gazebo::sensors::SensorManager::Run ( )

Run the sensor manager update in a new thread.

## 10.141.2.10 bool gazebo::sensors::SensorManager::SensorsInitialized ( )

True if SensorManager::initSensors queue is empty i.e.  
all sensors managed by **SensorManager** (p. 683) have been initialized

10.141.2.11 void gazebo::sensors::SensorManager::Stop ( )

Stop the run thread.

10.141.2.12 void gazebo::sensors::SensorManager::Update ( bool *\_force* = false )

Update all the sensors.

Checks to see if any sensor need to be initialized first, then updates all sensors once.

#### Parameters

in	<i>_force</i>	True force update, false if not
----	---------------	---------------------------------

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

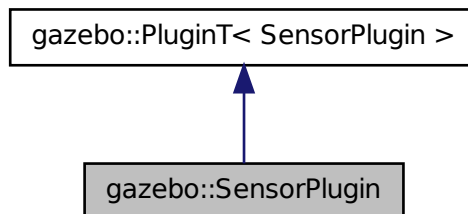
- **SensorManager.hh**

## 10.142 gazebo::SensorPlugin Class Reference

**A** (p. 107) 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

### 10.142.1 Detailed Description

**A** (p. 107) plugin with access to physics::Sensor.

See [reference](#).

### 10.142.2 Constructor & Destructor Documentation

#### 10.142.2.1 gazebo::SensorPlugin::SensorPlugin ( ) [inline]

Constructor.

References gazebo::SENSOR\_PLUGIN, and gazebo::PluginT< SensorPlugin >::type.

#### 10.142.2.2 virtual gazebo::SensorPlugin::~~SensorPlugin ( ) [inline],[virtual]

Destructor.

### 10.142.3 Member Function Documentation

#### 10.142.3.1 virtual void gazebo::SensorPlugin::Init ( ) [inline],[virtual]

Override this method for custom plugin initialization behavior.

#### 10.142.3.2 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.

#### 10.142.3.3 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**

## 10.143 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")
- bool **LoadString** (const std::string &\_sdfString)
- bool **ParseArgs** (int argc, char \*\*argv)
- void **PrintUsage** ()
- void **Run** ()
- void **SetParams** (const **common::StrStr\_M** &params)
- void **Stop** ()

### Public Attributes

- int **systemPluginsArgc**
- char \*\* **systemPluginsArgv**

### 10.143.1 Constructor & Destructor Documentation

10.143.1.1 gazebo::Server::Server ( )

10.143.1.2 virtual gazebo::Server::~~Server ( ) [virtual]

### 10.143.2 Member Function Documentation

10.143.2.1 void gazebo::Server::Fini ( )

10.143.2.2 bool gazebo::Server::GetInitialized ( ) const

10.143.2.3 void gazebo::Server::Init ( )

10.143.2.4 bool gazebo::Server::LoadFile ( const std::string & *filename* = "worlds/empty.world" )

10.143.2.5 bool gazebo::Server::LoadString ( const std::string & *sdfString* )

10.143.2.6 bool gazebo::Server::ParseArgs ( int *argc*, char \*\* *argv* )

10.143.2.7 void gazebo::Server::PrintUsage ( )

10.143.2.8 void gazebo::Server::Run ( )

10.143.2.9 void gazebo::Server::SetParams ( const **common::StrStr\_M** & *params* )

10.143.2.10 void gazebo::Server::Stop ( )

### 10.143.3 Member Data Documentation

10.143.3.1 int gazebo::Server::systemPluginsArgc

10.143.3.2 char\*\* gazebo::Server::systemPluginsArgv

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

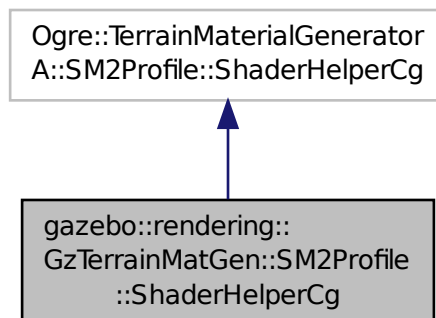
- [Server.hh](#)

## 10.144 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)

### 10.144.1 Detailed Description

Keeping the CG shader for reference.

Utility class to help with generating shaders for Cg / HLSL.

### 10.144.2 Member Function Documentation

- 10.144.2.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::defaultVpParams ( const **SM2Profile** \*\_prof, const Ogre::Terrain \*\_terrain, TechniqueType \_tt, const Ogre::HighLevelGpuProgramPtr & \_prog )  
[protected], [virtual]
- 10.144.2.2 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateFragmentProgram ( const **SM2Profile** \*\_prof, const Ogre::Terrain \*\_terrain, TechniqueType \_tt )  
[virtual]
- 10.144.2.3 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgram ( const **SM2Profile** \*\_prof, const Ogre::Terrain \*\_terrain, TechniqueType \_tt )  
[virtual]
- 10.144.2.4 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgramSource ( const **SM2Profile** \*\_prof, const Ogre::Terrain \*\_terrain, TechniqueType \_tt, Ogre::StringUtil::StrStreamType & \_outStream )  
[protected], [virtual]
- 10.144.2.5 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpDynamicShadows ( const **SM2Profile** \*\_prof, const Ogre::Terrain \*\_terrain, TechniqueType \_tt, Ogre::StringUtil::StrStreamType & \_outStream )  
[protected], [virtual]
- 10.144.2.6 virtual unsigned int gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpDynamicShadowsParams ( unsigned int \_texCoordStart, const **SM2Profile** \*\_prof, const Ogre::Terrain \*\_terrain, TechniqueType \_tt, Ogre::StringUtil::StrStreamType & \_outStream ) [protected], [virtual]
- 10.144.2.7 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpFooter ( const **SM2Profile** \*\_prof, const Ogre::Terrain \*\_terrain, TechniqueType \_tt, Ogre::StringUtil::StrStreamType & \_outStream )  
[protected], [virtual]
- 10.144.2.8 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:

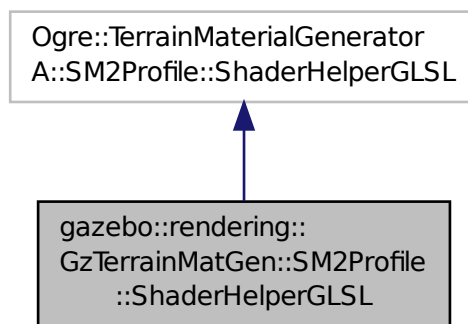
- [Heightmap.hh](#)

## 10.145 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)

### 10.145.1 Detailed Description

Utility class to help with generating shaders for GLSL.

### 10.145.2 Member Function Documentation

- 10.145.2.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::defaultVpParams ( const **SM2Profile** \*\_prof, const Ogre::Terrain \*\_terrain, TechniqueType \_tt, const Ogre::HighLevelGpuProgramPtr & \_prog ) [protected], [virtual]
- 10.145.2.2 void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadows ( const **SM2Profile** \*\_prof, const Ogre::Terrain \*\_terrain, TechniqueType \_tt, Ogre::StringUtil::StrStreamType & \_outStream ) [protected]
- 10.145.2.3 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadowsHelpers ( const **SM2Profile** \*\_prof, const Ogre::Terrain \*\_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & \_outStream ) [protected], [virtual]
- 10.145.2.4 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]
- 10.145.2.5 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpFooter ( const **SM2Profile** \*\_prof, const Ogre::Terrain \*\_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & \_outStream ) [protected], [virtual]
- 10.145.2.6 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpHeader ( const **SM2Profile** \*\_prof, const Ogre::Terrain \*\_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & \_outStream ) [protected], [virtual]



- 10.145.2.7 `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]`
- 10.145.2.8 `virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFragmentProgram ( const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType tt ) [virtual]`
- 10.145.2.9 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFragmentProgramSource ( const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & _outStream ) [protected], [virtual]`
- 10.145.2.10 `virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVertexProgram ( const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType tt ) [virtual]`
- 10.145.2.11 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVertexProgramSource ( const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & _outStream ) [protected], [virtual]`
- 10.145.2.12 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpDynamicShadows ( const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & _outStream ) [protected], [virtual]`
- 10.145.2.13 `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]`
- 10.145.2.14 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpFooter ( const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & _outStream ) [protected], [virtual]`
- 10.145.2.15 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpHeader ( const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & _outStream ) [protected], [virtual]`
- 10.145.2.16 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::updateParams ( const SM2Profile * _prof, const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain, bool _compositeMap ) [virtual]`
- 10.145.2.17 `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:

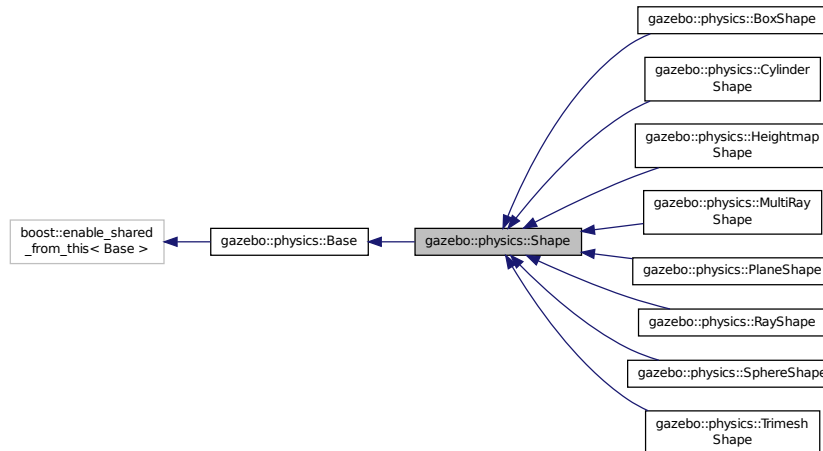
- **Heightmap.hh**

## 10.146 gazebo::physics::Shape Class Reference

**Base** (p. 133) 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

### 10.146.1 Detailed Description

**Base** (p. 133) class for all shapes.

### 10.146.2 Constructor & Destructor Documentation

10.146.2.1 `gazebo::physics::Shape::Shape ( CollisionPtr _parent ) [explicit]`

Constructor.

Parameters

in	_parent	Parent of the shape.
----	---------	----------------------

10.146.2.2 `virtual gazebo::physics::Shape::~~Shape ( ) [virtual]`

Destructor.

### 10.146.3 Member Function Documentation

10.146.3.1 `virtual void gazebo::physics::Shape::FillMsg ( msgs::Geometry & _msg ) [pure virtual]`

Fill in the values for a geometry message.

Parameters

out	_msg	The geometry message to fill.
-----	------	-------------------------------

Implemented in `gazebo::physics::MultiRayShape` (p.511), `gazebo::physics::RayShape` (p.625), `gazebo::physics::HeightmapShape` (p.346), `gazebo::physics::PlaneShape` (p.568), `gazebo::physics::TrimeshShape` (p.791), `gazebo::physics::CylinderShape` (p.244), `gazebo::physics::SphereShape` (p.724), and `gazebo::physics::BoxShape` (p.151).

10.146.3.2 `virtual void gazebo::physics::Shape::Init ( ) [pure virtual]`

Initialize the shape.

Reimplemented from `gazebo::physics::Base` (p.141).

Implemented in `gazebo::physics::RayShape` (p.626), `gazebo::physics::HeightmapShape` (p.347), `gazebo::physics::TrimeshShape` (p.792), `gazebo::physics::MultiRayShape` (p.514), `gazebo::physics::PlaneShape` (p.568), `gazebo::physics::SphereShape` (p.724), `gazebo::physics::BoxShape` (p.151), and `gazebo::physics::CylinderShape` (p.245).

10.146.3.3 `virtual void gazebo::physics::Shape::ProcessMsg ( const msgs::Geometry & _msg ) [pure virtual]`

Process a geometry message.

Parameters

in	_msg	The message to set values from.
----	------	---------------------------------

Implemented in `gazebo::physics::MultiRayShape` (p.514), `gazebo::physics::RayShape` (p.626), `gazebo::physics::HeightmapShape` (p.348), `gazebo::physics::PlaneShape` (p.568), `gazebo::physics::TrimeshShape` (p.792), `gazebo::physics::CylinderShape` (p.245), `gazebo::physics::SphereShape` (p.724), and `gazebo::physics::BoxShape` (p.151).

#### 10.146.4 Member Data Documentation

##### 10.146.4.1 CollisionPtr gazebo::physics::Shape::collisionParent [protected]

This shape's collision parent.

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

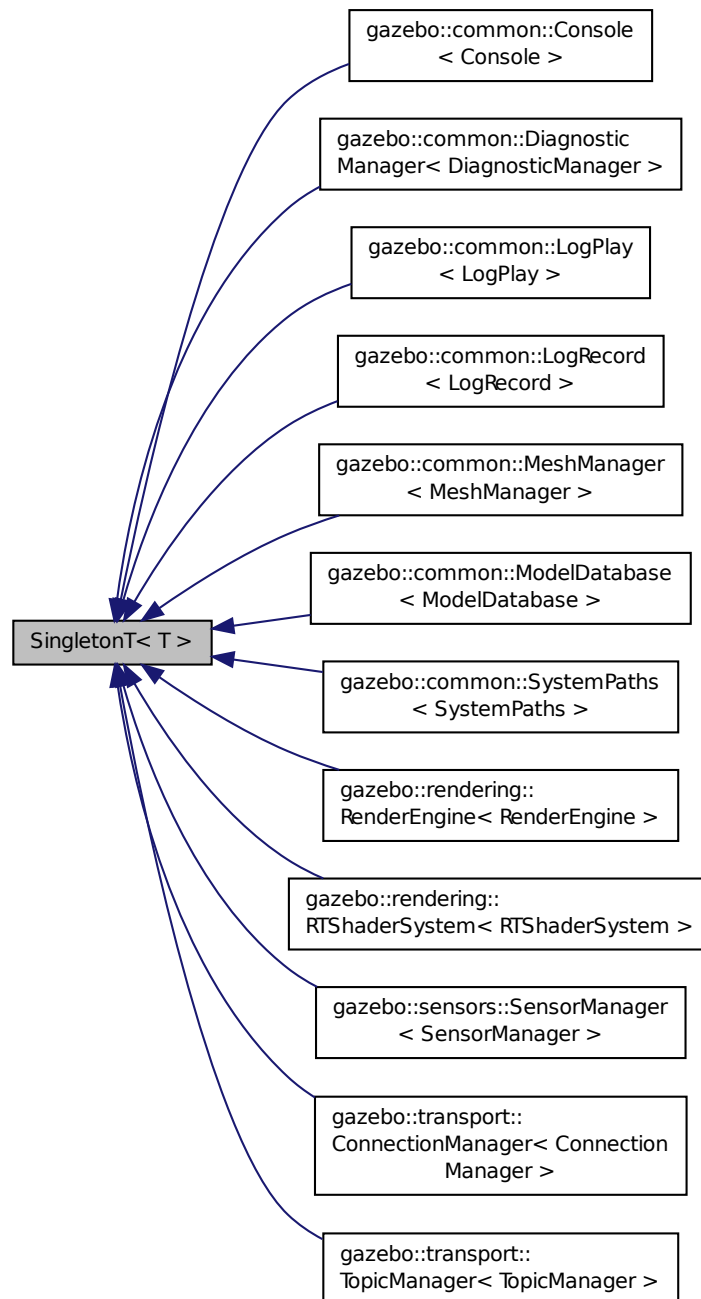
- **Shape.hh**

#### 10.147 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.*

### 10.147.1 Detailed Description

```
template<class T>class SingletonT< T >
```

Singleton template class.

### 10.147.2 Constructor & Destructor Documentation

10.147.2.1 `template<class T> SingletonT< T >::SingletonT ( )` [inline],[protected]

Constructor.

10.147.2.2 `template<class T> virtual SingletonT< T >::~~SingletonT ( )` [inline],[protected],[virtual]

Destructor.

### 10.147.3 Member Function Documentation

10.147.3.1 `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**

## 10.148 gazebo::common::Skeleton Class Reference

**A** (p. 107) 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*

### 10.148.1 Detailed Description

**A** (p. 107) skeleton.

### 10.148.2 Constructor & Destructor Documentation

#### 10.148.2.1 gazebo::common::Skeleton::Skeleton ( )

Constructor.

#### 10.148.2.2 gazebo::common::Skeleton::Skeleton ( **SkeletonNode** \* *\_root* )

Constructor.

##### Parameters

in	<i>_root</i>	node
----	--------------	------

#### 10.148.2.3 virtual gazebo::common::Skeleton::~Skeleton ( ) [virtual]

Destructor.

### 10.148.3 Member Function Documentation

#### 10.148.3.1 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
----	--------------	----------------------



10.148.3.2 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)

10.148.3.3 void gazebo::common::Skeleton::BuildNodeMap ( ) [protected]

Initializes the handle numbers for each node in the map using breadth first traversal.

10.148.3.4 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

10.148.3.5 math::Matrix4 gazebo::common::Skeleton::GetBindShapeTransform ( )

Return bind pose skeletal transform.

#### Returns

a matrix

10.148.3.6 SkeletonNode\* gazebo::common::Skeleton::GetNodeByHandle ( unsigned int *\_handle* )

Find or create node with handle.

#### Parameters

in	<i>_handle</i>	
----	----------------	--

#### Returns

the node. **A** (p. 107) new node is created if it didn't exist

10.148.3.7 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

### 10.148.3.8 `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

### 10.148.3.9 `NodeMap` `gazebo::common::Skeleton::GetNodes ( )`

Get a copy of the node dictionary.

### 10.148.3.10 `unsigned int` `gazebo::common::Skeleton::GetNumAnimations ( )`

Returns the number of animations.

## Returns

the count

### 10.148.3.11 `unsigned int` `gazebo::common::Skeleton::GetNumJoints ( )`

Returns the number of joints.

## Returns

the count

### 10.148.3.12 `unsigned int` `gazebo::common::Skeleton::GetNumNodes ( )`

Returns the node count.

## Returns

the count

10.148.3.13 `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

10.148.3.14 `SkeletonNode* gazebo::common::Skeleton::GetRootNode ( )`

Return the root.

#### Returns

the root

10.148.3.15 `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

10.148.3.16 `void gazebo::common::Skeleton::PrintTransforms ( )`

Outputs the transforms to `std::err` stream.

10.148.3.17 `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
-----------------	-------------------------	----------------

10.148.3.18 `void gazebo::common::Skeleton::SetBindShapeTransform ( math::Matrix4 _trans )`

Set the bind pose skeletal transform.

## Parameters

in	<i>_trans</i>	the transform
----	---------------	---------------

10.148.3.19 void gazebo::common::Skeleton::SetNumVertAttached ( unsigned int *\_vertices* )

Resizes the raw node weight array.

## Parameters

in	<i>_vertices</i>	the new size
----	------------------	--------------

10.148.3.20 void gazebo::common::Skeleton::SetRootNode ( **SkeletonNode** \* *\_node* )

Change the root node.

## Parameters

in	<i>_node</i>	the new node
----	--------------	--------------

## 10.148.4 Member Data Documentation

10.148.4.1 **std::vector<SkeletonAnimation\*>** gazebo::common::Skeleton::anim [protected]

the array of animations

10.148.4.2 **math::Matrix4** gazebo::common::Skeleton::bindShapeTransform [protected]

the bind pose skeletal transform

10.148.4.3 **NodeMap** gazebo::common::Skeleton::nodes [protected]

The dictionary of nodes, indexed by name.

10.148.4.4 **RawNodeWeights** gazebo::common::Skeleton::rawNW [protected]

the node weight table

10.148.4.5 **SkeletonNode\*** gazebo::common::Skeleton::root [protected]

the root node

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

- **Skeleton.hh**

## 10.149 gazebo::common::SkeletonAnimation Class Reference

**Skeleton** (p. 698) 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*

### 10.149.1 Detailed Description

**Skeleton** (p. 698) animation.

### 10.149.2 Constructor & Destructor Documentation

#### 10.149.2.1 gazebo::common::SkeletonAnimation::SkeletonAnimation ( const std::string & *\_name* )

The Constructor.

##### Parameters

in	<i>_name</i>	the name of the animation
----	--------------	---------------------------

#### 10.149.2.2 gazebo::common::SkeletonAnimation::~~SkeletonAnimation ( )

The destructor.

Clears the list without destroying the animations

### 10.149.3 Member Function Documentation

#### 10.149.3.1 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

#### 10.149.3.2 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 <b>math::Pose</b> (p. 573)

#### 10.149.3.3 double gazebo::common::SkeletonAnimation::GetLength ( ) const

Returns the duration of the animations.

**Returns**

the duration in seconds

10.149.3.4 `std::string gazebo::common::SkeletonAnimation::GetName ( ) const`

Returns the name.

**Returns**

the name

10.149.3.5 `unsigned int gazebo::common::SkeletonAnimation::GetNodeCount ( ) const`

Returns the number of animation nodes.

**Returns**

the count

10.149.3.6 `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

10.149.3.7 `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

**10.149.3.8** `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)

**10.149.3.9** `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

**10.149.3.10** `void gazebo::common::SkeletonAnimation::Scale ( const double _scale )`

Scales every animation in the animations list.

**Parameters**

in	<i>_scale</i>	the scaling factor
----	---------------	--------------------

**10.149.3.11** `void gazebo::common::SkeletonAnimation::SetName ( const std::string & _name )`

Changes the name.

**Parameters**

in	<i>_name</i>	the new name
----	--------------	--------------

**10.149.4 Member Data Documentation**

**10.149.4.1** `std::map<std::string, NodeAnimation*> gazebo::common::SkeletonAnimation::animations` `[protected]`

a dictionary of node animations



10.149.4.2 double gazebo::common::SkeletonAnimation::length [protected]

the duration of the longest animation

10.149.4.3 std::string gazebo::common::SkeletonAnimation::name [protected]

the node name

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

- **SkeletonAnimation.hh**

## 10.150 gazebo::common::SkeletonNode Class Reference

**A** (p. 107) 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*

### 10.150.1 Detailed Description

**A** (p. 107) skeleton node.

### 10.150.2 Member Enumeration Documentation

#### 10.150.2.1 enum gazebo::common::SkeletonNode::SkeletonNodeType

enumeration of node types

Enumerator:

***NODE***

***JOINT***

### 10.150.3 Constructor & Destructor Documentation

#### 10.150.3.1 gazebo::common::SkeletonNode::SkeletonNode ( **SkeletonNode** \* *\_parent* )

Constructor.

Parameters

<code>in</code>	<code><i>_parent</i></code>	The parent node
-----------------	-----------------------------	-----------------

10.150.3.2 `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

10.150.3.3 `virtual gazebo::common::SkeletonNode::~~SkeletonNode ( ) [virtual]`

Destructor.

### 10.150.4 Member Function Documentation

10.150.4.1 `void gazebo::common::SkeletonNode::AddChild ( SkeletonNode * _child )`

Add a new child.

#### Parameters

in	<code>_child</code>	a child
----	---------------------	---------

10.150.4.2 `void gazebo::common::SkeletonNode::AddRawTransform ( NodeTransform _t )`

Add a raw transform.

#### Parameters

in	<code>_t</code>	the transform
----	-----------------	---------------

10.150.4.3 `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

10.150.4.4 `SkeletonNode* gazebo::common::SkeletonNode::GetChildById ( std::string _id )`

Get child by string id.

## Parameters

in	_id	the string id
----	-----	---------------

## Returns

the child skeleton or NULL if not found

#### 10.150.4.5 SkeletonNode\* gazebo::common::SkeletonNode::GetChildByName ( std::string \_name )

Get child by name.

## Parameters

in	_name	the name of the child skeleton
----	-------	--------------------------------

## Returns

the skeleton, or NULL if not found

#### 10.150.4.6 unsigned int gazebo::common::SkeletonNode::GetChildCount ( )

Returns the children count.

## Returns

the count

#### 10.150.4.7 unsigned int gazebo::common::SkeletonNode::GetHandle ( )

Get the handle index.

## Returns

the handle index

#### 10.150.4.8 std::string gazebo::common::SkeletonNode::GetId ( )

Returns the index.

## Returns

the id string

#### 10.150.4.9 math::Matrix4 gazebo::common::SkeletonNode::GetInverseBindTransform ( )

Retrieve the inverse of the bind pose skeletal transform.

## Returns

the transform

10.150.4.10 `math::Matrix4 gazebo::common::SkeletonNode::GetModelTransform ( )`

Retrieve the model transform.

**Returns**

the transform

10.150.4.11 `std::string gazebo::common::SkeletonNode::GetName ( )`

Returns the name.

**Returns**

the name

10.150.4.12 `unsigned int gazebo::common::SkeletonNode::GetNumRawTrans ( )`

Return the raw transformations count.

**Returns**

the count

10.150.4.13 `SkeletonNode* gazebo::common::SkeletonNode::GetParent ( )`

Returns the parent node.

**Returns**

the parent

10.150.4.14 `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

10.150.4.15 `std::vector<NodeTransform> gazebo::common::SkeletonNode::GetRawTransforms ( )`

Retrieve the raw transformations.

**Returns**

an array of transformations

10.150.4.16 `math::Matrix4 gazebo::common::SkeletonNode::GetTransform ( )`

Get transform relative to parent.

10.150.4.17 `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)

10.150.4.18 `bool gazebo::common::SkeletonNode::IsJoint ( )`

Is a joint query.

**Returns**

true if the skeleton type is a joint, false otherwise

10.150.4.19 `bool gazebo::common::SkeletonNode::IsRootNode ( )`

Queries whether a node has no parent parent.

**Returns**

true if the node has no parent, false otherwise

10.150.4.20 `void gazebo::common::SkeletonNode::Reset ( bool _resetChildren )`

Reset the transformation to the initial transformation.

**Parameters**

<code>in</code>	<code><i>_resetChildren</i></code>	when true, performs the operation for every node in the tree
-----------------	------------------------------------	--

10.150.4.21 `void gazebo::common::SkeletonNode::SetHandle ( unsigned int _h )`

Assign a handle number.

**Parameters**

<code>in</code>	<code><i>_h</i></code>	the handle
-----------------	------------------------	------------

10.150.4.22 void gazebo::common::SkeletonNode::SetId ( std::string *\_id* )

Change the id string.

Parameters

in	<i>_id</i>	the new id string
----	------------	-------------------

10.150.4.23 void gazebo::common::SkeletonNode::SetInitialTransform ( math::Matrix4 *\_tras* )

Sets the initial transformation.

Parameters

in	<i>_tras</i>	the transformation matrix
----	--------------	---------------------------

10.150.4.24 void gazebo::common::SkeletonNode::SetInverseBindTransform ( math::Matrix4 *\_invBM* )

Assign the inverse of the bind pose skeletal transform.

Parameters

in	<i>_invBM</i>	the transform
----	---------------	---------------

10.150.4.25 void gazebo::common::SkeletonNode::SetModelTransform ( math::Matrix4 *\_trans*, bool *\_updateChildren* = true )

Set the model transformation.

Parameters

in	<i>_trans</i>	the transformation
in	<i>_updateChildren</i>	when true the UpdateChildrenTransforms operation is performed

10.150.4.26 void gazebo::common::SkeletonNode::SetName ( std::string *\_name* )

Change the name.

Parameters

in	<i>_name</i>	the new name
----	--------------	--------------

10.150.4.27 void gazebo::common::SkeletonNode::SetParent ( SkeletonNode \* *\_parent* )

Set the parent node.

Parameters

in	<i>_parent</i>	the new parent
----	----------------	----------------



10.150.4.28 void gazebo::common::SkeletonNode::SetTransform ( math::Matrix4 *\_trans*, bool *\_updateChildren = true* )

Set a transformation.

#### Parameters

in	<i>_trans</i>	the transformation
in	<i>_updateChildren</i>	when true the UpdateChildrenTransforms operation is performed

10.150.4.29 void gazebo::common::SkeletonNode::SetType ( SkeletonNodeType *\_type* )

Change the skeleton node type.

#### Parameters

in	<i>_type</i>	the new type
----	--------------	--------------

10.150.4.30 void gazebo::common::SkeletonNode::UpdateChildrenTransforms ( )

Apply model transformations in order for each node in the tree.

## 10.150.5 Member Data Documentation

10.150.5.1 std::vector<SkeletonNode\*> gazebo::common::SkeletonNode::children [protected]

the children nodes

10.150.5.2 unsigned int gazebo::common::SkeletonNode::handle [protected]

handle index number

10.150.5.3 std::string gazebo::common::SkeletonNode::id [protected]

a string identifier

10.150.5.4 math::Matrix4 gazebo::common::SkeletonNode::initialTransform [protected]

the initial transformation

10.150.5.5 math::Matrix4 gazebo::common::SkeletonNode::invBindTransform [protected]

the inverse of the bind pose skeletal transform

10.150.5.6 math::Matrix4 gazebo::common::SkeletonNode::modelTransform [protected]

the model transformation

10.150.5.7 `std::string gazebo::common::SkeletonNode::name` [protected]

the name of the skeletal node

10.150.5.8 `SkeletonNode* gazebo::common::SkeletonNode::parent` [protected]

the parent node

10.150.5.9 `std::vector<NodeTransform> gazebo::common::SkeletonNode::rawTransforms` [protected]

the raw transformation

10.150.5.10 `math::Matrix4 gazebo::common::SkeletonNode::transform` [protected]

the transform

10.150.5.11 `SkeletonNodeType gazebo::common::SkeletonNode::type` [protected]

the type fo node

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

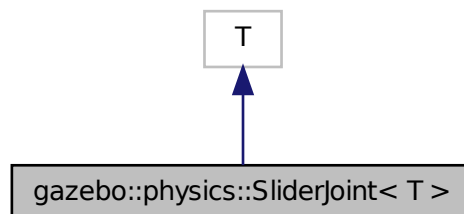
- **Skeleton.hh**

## 10.151 gazebo::physics::SliderJoint< T > Class Template Reference

**A** (p. 107) 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. 718).*

- virtual void `SetAnchor (int _index, const math::Vector3 &_anchor)`

*Set the anchor.*

## Protected Attributes

- `math::Vector3 fakeAnchor`

*The anchor value is not used internally.*

### 10.151.1 Detailed Description

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

**A** (p. 107) slider joint.

### 10.151.2 Constructor & Destructor Documentation

10.151.2.1 `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`.

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

Destructor.

### 10.151.3 Member Function Documentation

10.151.3.1 `template<class T > math::Vector3 gazebo::physics::SliderJoint< T >::GetAnchor ( int _index ) const [virtual]`

Get the anchor.

## Parameters

in	<code>_index</code>	Index of the axis. Not used.
----	---------------------	------------------------------

## Returns

Anchor for the joint.

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

10.151.3.3 `template<class T > virtual void gazebo::physics::SliderJoint< T >::Load ( sdf::ElementPtr _sdf )`  
`[inline], [virtual]`

Load a **SliderJoint** (p. 718).

## Parameters

in	<code>_sdf</code>	SDF values to load from
----	-------------------	-------------------------

10.151.3.4 `template<class T > void gazebo::physics::SliderJoint< T >::SetAnchor ( int _index, const math::Vector3 & _anchor )`  
`[virtual]`

Set the anchor.

## Parameters

in	<code>_index</code>	Index of the axis. Not used.
in	<code>_anchor</code>	Anchor for the axis.

## 10.151.4 Member Data Documentation

10.151.4.1 `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:

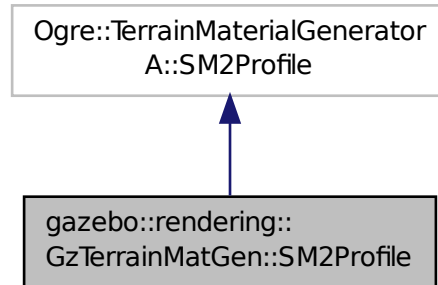
- **SliderJoint.hh**

## 10.152 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)

### 10.152.1 Detailed Description

Shader model 2 profile target.

### 10.152.2 Constructor & Destructor Documentation

10.152.2.1 `gazebo::rendering::GzTerrainMatGen::SM2Profile::SM2Profile ( Ogre::TerrainMaterialGenerator * _parent, const Ogre::String & _name, const Ogre::String & _desc )`

Constructor.

10.152.2.2 `virtual gazebo::rendering::GzTerrainMatGen::SM2Profile::~SM2Profile ( ) [virtual]`

Destructor.

### 10.152.3 Member Function Documentation

10.152.3.1 `virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::addTechnique ( const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain, TechniqueType _tt ) [protected],[virtual]`

10.152.3.2 `Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generate ( const Ogre::Terrain * _terrain )`

10.152.3.3 `Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generateForCompositeMap ( const Ogre::Terrain * _terrain )`

10.152.3.4 `void gazebo::rendering::GzTerrainMatGen::SM2Profile::UpdateParams ( const Ogre::MaterialPtr & _mat, const Ogre::Terrain * _terrain )`

10.152.3.5 `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:

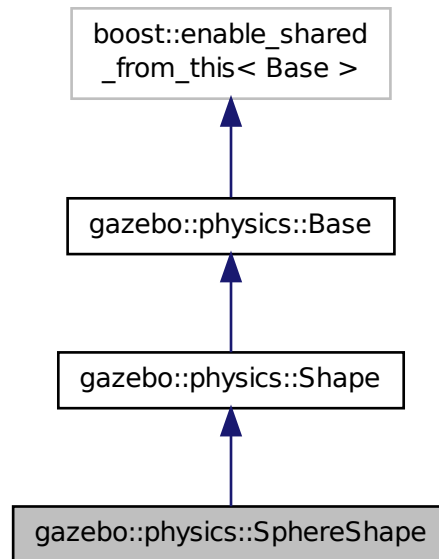
- **Heightmap.hh**

## 10.153 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

### 10.153.1 Detailed Description

Sphere collision shape.

### 10.153.2 Constructor & Destructor Documentation

10.153.2.1 `gazebo::physics::SphereShape::SphereShape ( CollisionPtr _parent ) [explicit]`

Constructor.

#### Parameters

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

10.153.2.2 `virtual gazebo::physics::SphereShape::~~SphereShape ( ) [virtual]`

Destructor.

### 10.153.3 Member Function Documentation

10.153.3.1 `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. 695).

10.153.3.2 `double gazebo::physics::SphereShape::GetRadius ( ) const`

Get the sphere's radius.

#### Returns

Radius of the sphere.

10.153.3.3 `virtual void gazebo::physics::SphereShape::Init ( ) [virtual]`

Initialize the sphere.

Implements **`gazebo::physics::Shape`** (p. 695).

10.153.3.4 `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. 695).



10.153.3.5 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**

## 10.154 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.*

### 10.154.1 Detailed Description

Splines.

### 10.154.2 Constructor & Destructor Documentation

#### 10.154.2.1 gazebo::math::Spline::Spline ( )

constructor

#### 10.154.2.2 gazebo::math::Spline::~~Spline ( )

destructor

### 10.154.3 Member Function Documentation

#### 10.154.3.1 void gazebo::math::Spline::AddPoint ( const Vector3 & \_pt )

Adds a control point to the end of the spline.

Parameters

in	_pt	point to add
----	-----	--------------

#### 10.154.3.2 void gazebo::math::Spline::Clear ( )

Clears all the points in the spline.

#### 10.154.3.3 Vector3 gazebo::math::Spline::GetPoint ( unsigned int \_index ) const

Gets the detail of one of the control points of the spline.

## Parameters

in	<i>_index</i>	the control point index
----	---------------	-------------------------

## Returns

the control point, or [0,0,0] and a message on the error stream

## 10.154.3.4 unsigned int gazebo::math::Spline::GetPointCount ( ) const

Gets the number of control points in the spline.

## Returns

the count

10.154.3.5 Vector3 gazebo::math::Spline::GetTangent ( unsigned int *\_index* ) const

Get the tangent value for a point.

## Parameters

in	<i>_index</i>	the control point index
----	---------------	-------------------------

## 10.154.3.6 double gazebo::math::Spline::GetTension ( ) const

Get the tension value.

## Returns

The value of the tension, which is between 0.0 and 1.0

10.154.3.7 Vector3 gazebo::math::Spline::Interpolate ( double *\_t* ) const

Returns an interpolated point based on a parametric value over the whole series.

## Parameters

in	<i>_t</i>	parameter (range 0 to 1)
----	-----------	--------------------------

10.154.3.8 Vector3 gazebo::math::Spline::Interpolate ( unsigned int *\_fromIndex*, double *\_t* ) const

Interpolates a single segment of the spline given a parametric value.

## Parameters

in	<i>_fromIndex</i>	The point index to treat as t = 0. fromIndex + 1 is deemed to be t = 1
in	<i>_t</i>	Parametric value

### 10.154.3.9 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.

### 10.154.3.10 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.
----	------------------	--

### 10.154.3.11 void gazebo::math::Spline::SetTension ( double *\_t* )

Set the tension parameter.

**A** (p. 107) value of 0 = Catmull-Rom spline.

#### Parameters

in	<i>_t</i>	Tension value between 0.0 and 1.0
----	-----------	-----------------------------------

### 10.154.3.12 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

## 10.154.4 Member Data Documentation

10.154.4.1 `bool gazebo::math::Spline::autoCalc` [protected]

when true, the tangents are recalculated when the control point change

10.154.4.2 `Matrix4 gazebo::math::Spline::coeffs` [protected]

Matrix of coefficients.

10.154.4.3 `std::vector<Vector3> gazebo::math::Spline::points` [protected]

control points

10.154.4.4 `std::vector<Vector3> gazebo::math::Spline::tangents` [protected]

tangents

10.154.4.5 `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:

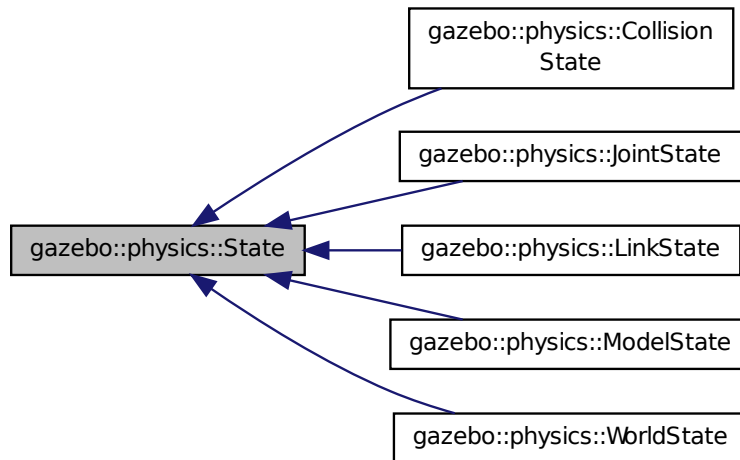
- `Spline.hh`

## 10.155 gazebo::physics::State Class Reference

**State** (p. 729) 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. 729).*
- **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. 729).*

## Protected Attributes

- `std::string name`  
Name associated with this **State** (p. 729).
- `common::Time realTime`
- `common::Time simTime`
- `common::Time wallTime`  
Times for the state data.

### 10.155.1 Detailed Description

**State** (p. 729) of an entity.

This is the base class for all **State** (p. 729) information.

### 10.155.2 Constructor & Destructor Documentation

#### 10.155.2.1 gazebo::physics::State::State ( )

Default constructor.

#### 10.155.2.2 gazebo::physics::State::State ( const std::string & \_name, const common::Time & \_realTime, const common::Time & \_simTime )

Constructor.

Construct a **State** (p. 729) object using some basic information.

#### Parameters

<code>_name</code>	Name associated with the <b>State</b> (p. 729) information. This is typically the name of an <b>Entity</b> (p. 274). <code>_realTime</code> Clock time since simulation started.
<code>_simTime</code>	Simulation time associated with this <b>State</b> (p. 729) info.

#### 10.155.2.3 virtual gazebo::physics::State::~~State ( ) [virtual]

Destructor.

### 10.155.3 Member Function Documentation

#### 10.155.3.1 std::string gazebo::physics::State::GetName ( ) const

Get the name associated with this **State** (p. 729).

#### Returns

Name associated with this state information. Typically a name of an **Entity** (p. 274).

### 10.155.3.2 `common::Time gazebo::physics::State::GetRealTime ( ) const`

Get the real time when this state was generated.

#### Returns

Clock time since simulation was stated.

### 10.155.3.3 `common::Time gazebo::physics::State::GetSimTime ( ) const`

Get the sim time when this state was generated.

#### Returns

Simulation time when the data was recorded.

### 10.155.3.4 `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. 729) data was recorded.

### 10.155.3.5 `virtual void gazebo::physics::State::Load ( const sdf::ElementPtr _elem ) [virtual]`

Load state from SDF element.

Populates the **State** (p. 729) information from data stored in an SDF::Element

#### Parameters

<code>_elem</code>	Pointer to the SDF::Element
--------------------	-----------------------------

Reimplemented in **gazebo::physics::ModelState** (p. 491), **gazebo::physics::LinkState** (p. 426), **gazebo::physics::WorldState** (p. 892), **gazebo::physics::CollisionState** (p. 201), and **gazebo::physics::JointState** (p. 390).

### 10.155.3.6 `State gazebo::physics::State::operator-( const State & _state ) const`

Subtraction operator.

#### Parameters

<code>in</code>	<code>_pt</code>	<b>A</b> (p. 107) state to subtract.
-----------------	------------------	--------------------------------------

#### Returns

The resulting state.



10.155.3.7 `State& gazebo::physics::State::operator= ( const State & _state )`

Assignment operator.

## Parameters

<code>in</code>	<code><i>_state</i></code>	<b>State</b> (p. 729) value
-----------------	----------------------------	-----------------------------

## Returns

`this`

10.155.3.8 `void gazebo::physics::State::SetName ( const std::string & _name )`

Set the name associated with this **State** (p. 729).

## Parameters

<code>in</code>	<code><i>_name</i></code>	Name associated with this state information. Typically the name of an <b>Entity</b> (p. 274).
-----------------	---------------------------	---

## 10.155.4 Member Data Documentation

10.155.4.1 `std::string gazebo::physics::State::name` [protected]

Name associated with this **State** (p. 729).

10.155.4.2 `common::Time gazebo::physics::State::realTime` [protected]10.155.4.3 `common::Time gazebo::physics::State::simTime` [protected]10.155.4.4 `common::Time gazebo::physics::State::wallTime` [protected]

Times for the state data.

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

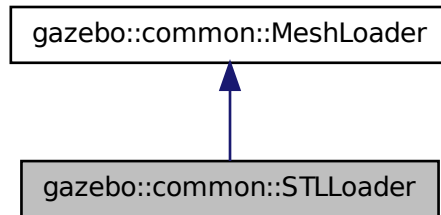
- **State.hh**

## 10.156 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.*

### 10.156.1 Detailed Description

Class used to load STL mesh files.

### 10.156.2 Constructor & Destructor Documentation

#### 10.156.2.1 gazebo::common::STLLoader::STLLoader ( )

Constructor.

#### 10.156.2.2 virtual gazebo::common::STLLoader::~~STLLoader ( ) [virtual]

Destructor.

### 10.156.3 Member Function Documentation

#### 10.156.3.1 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. 464).

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

- **STLloader.hh**

## 10.157 gazebo::common::SubMesh Class Reference

**A** (p. 107) 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.*

### 10.157.1 Detailed Description

**A** (p. 107) child mesh.

### 10.157.2 Member Enumeration Documentation

#### 10.157.2.1 enum gazebo::common::SubMesh::PrimitiveType

An enumeration of the geometric mesh primitives.

Enumerator:

**POINTS**  
**LINES**  
**LINESTRIPS**  
**TRIANGLES**  
**TRIFANS**  
**TRISTRIPS**

### 10.157.3 Constructor & Destructor Documentation

#### 10.157.3.1 gazebo::common::SubMesh::SubMesh ( )

Constructor.

#### 10.157.3.2 virtual gazebo::common::SubMesh::~~SubMesh ( ) [virtual]

Destructor.

### 10.157.4 Member Function Documentation

#### 10.157.4.1 void gazebo::common::SubMesh::AddIndex ( unsigned int \_i )

Add an index to the mesh.

## Parameters

in	<code>_i</code>	the new vertex index
----	-----------------	----------------------

10.157.4.2 `void gazebo::common::SubMesh::AddNodeAssignment ( unsigned int _vertex, unsigned int _node, float _weight )`

Add a vertex - skeleton node assignment.

## Parameters

in	<code>_vertex</code>	the vertex index
in	<code>_node</code>	the node index
in	<code>_weight</code>	the weight (between 0 and 1)

10.157.4.3 `void gazebo::common::SubMesh::AddNormal ( const math::Vector3 & _n )`

Add a normal to the mesh.

## Parameters

in	<code>_n</code>	the normal
----	-----------------	------------

10.157.4.4 `void gazebo::common::SubMesh::AddNormal ( double _x, double _y, double _z )`

Add a normal to the mesh.

## Parameters

in	<code>_x</code>	position along x
in	<code>_y</code>	position along y
in	<code>_z</code>	position along z

10.157.4.5 `void gazebo::common::SubMesh::AddTexCoord ( double _u, double _v )`

Add a texture coord to the mesh.

## Parameters

in	<code>_u</code>	position along u
in	<code>_v</code>	position along v

10.157.4.6 `void gazebo::common::SubMesh::AddVertex ( const math::Vector3 & _v )`

Add a vertex to the mesh.

## Parameters

in	<code>_v</code>	the new position
----	-----------------	------------------

10.157.4.7 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

10.157.4.8 void gazebo::common::SubMesh::CopyNormals ( const std::vector< math::Vector3 > & *\_norms* )

Copy normals from a vector.

Parameters

in	<i>_norms</i>	to copy from
----	---------------	--------------

10.157.4.9 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
----	---------------	---------------------------

10.157.4.10 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>	

10.157.4.11 void gazebo::common::SubMesh::GenSphericalTexCoord ( const math::Vector3 & *\_center* )

Generate texture coordinates using spherical projection from center.

Parameters

in	<i>_center</i>	
----	----------------	--

10.157.4.12 unsigned int gazebo::common::SubMesh::GetIndex ( unsigned int *\_i* ) const

Get an index.

## Parameters

in	<i>_i</i>	
----	-----------	--

10.157.4.13 `unsigned int gazebo::common::SubMesh::GetIndexCount ( ) const`

Return the number of indices.

10.157.4.14 `unsigned int gazebo::common::SubMesh::GetMaterialIndex ( ) const`

Get the material index.

10.157.4.15 `math::Vector3 gazebo::common::SubMesh::GetMax ( ) const`

Get the maximum X, Y, Z values.

## Returns

10.157.4.16 `unsigned int gazebo::common::SubMesh::GetMaxIndex ( ) const`

Get the highest index value.

10.157.4.17 `math::Vector3 gazebo::common::SubMesh::GetMin ( ) const`

Get the minimum X, Y, Z values.

## Returns

10.157.4.18 `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
----	-----------	-----------------------------

10.157.4.19 `unsigned int gazebo::common::SubMesh::GetNodeAssignmentsCount ( ) const`

Return the number of vertex - skeleton node assignments.

10.157.4.20 `math::Vector3 gazebo::common::SubMesh::GetNormal ( unsigned int _i ) const`

Get a normal.



## Parameters

in	<i>_i</i>	the normal index
----	-----------	------------------

## Returns

the orientation of the normal, or throws an exception

10.157.4.21 `unsigned int gazebo::common::SubMesh::GetNormalCount ( ) const`

Return the number of normals.

10.157.4.22 `PrimitiveType gazebo::common::SubMesh::GetPrimitiveType ( ) const`

Get the primitive type.

## Returns

the primitive type

10.157.4.23 `math::Vector2d gazebo::common::SubMesh::GetTexCoord ( unsigned int _i ) const`

Get a tex coord.

## Parameters

in	<i>_i</i>	the texture index
----	-----------	-------------------

## Returns

the texture coordinates

10.157.4.24 `unsigned int gazebo::common::SubMesh::GetTexCoordCount ( ) const`

Return the number of texture coordinates.

10.157.4.25 `math::Vector3 gazebo::common::SubMesh::GetVertex ( unsigned int _i ) const`

Get a vertex.

## Parameters

in	<i>_i</i>	the vertex index
----	-----------	------------------

## Returns

the position or throws an exception

10.157.4.26 `unsigned int gazebo::common::SubMesh::GetVertexCount ( ) const`

Return the number of vertices.

10.157.4.27 `unsigned int gazebo::common::SubMesh::GetVertexIndex ( const math::Vector3 & _v ) const`

Get the index of the vertex.

**Parameters**

<code>in</code>	<code>_v</code>	
-----------------	-----------------	--

10.157.4.28 `bool gazebo::common::SubMesh::HasVertex ( const math::Vector3 & _v ) const`

Return true if this submesh has the vertex.

**Parameters**

<code>in</code>	<code>_v</code>	
-----------------	-----------------	--

10.157.4.29 `void gazebo::common::SubMesh::RecalculateNormals ( )`

Recalculate all the normals.

10.157.4.30 `void gazebo::common::SubMesh::Scale ( double _factor )`

Scale all vertices by `_factor`.

**Parameters**

<code>in</code>	<code>_factor</code>	Scaling factor
-----------------	----------------------	----------------

10.157.4.31 `void gazebo::common::SubMesh::SetIndexCount ( unsigned int _count )`

Resize the index array.

**Parameters**

<code>in</code>	<code>_count</code>	the new size of the array
-----------------	---------------------	---------------------------

10.157.4.32 `void gazebo::common::SubMesh::SetMaterialIndex ( unsigned int _index )`

Set the material index.

Relates to the parent mesh material list

## Parameters

in	<i>_index</i>	
----	---------------	--

10.157.4.33 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

10.157.4.34 void gazebo::common::SubMesh::SetNormalCount ( unsigned int *\_count* )

Resize the normal array.

## Parameters

in	<i>_count</i>	the new size of the array
----	---------------	---------------------------

10.157.4.35 void gazebo::common::SubMesh::SetPrimitiveType ( PrimitiveType *\_type* )

Set the primitive type.

## Parameters

in	<i>_type</i>	the type
----	--------------	----------

10.157.4.36 void gazebo::common::SubMesh::SetScale ( const math::Vector3 & *\_factor* )

Scale all vertices by the *\_factor* vector.

## Parameters

in	<i>_factor</i>	Scaling vector
----	----------------	----------------

10.157.4.37 void gazebo::common::SubMesh::SetSubMeshCenter ( math::Vector3 *\_center* )

Reset mesh center to geometric center.

## Parameters

in	<i>_center</i>	
----	----------------	--

10.157.4.38 void gazebo::common::SubMesh::SetTexCoord ( unsigned int *\_i*, const math::Vector2d & *\_t* )

Set a tex coord.

## Parameters

in	<code>_i</code>	
in	<code>_t</code>	

10.157.4.39 void gazebo::common::SubMesh::SetTexCoordCount ( unsigned int *\_count* )

Resize the texture coordinate array.

## Parameters

in	<code>_count</code>	
----	---------------------	--

10.157.4.40 void gazebo::common::SubMesh::SetVertex ( unsigned int *\_i*, const math::Vector3 & *\_v* )

Set a vertex.

## Parameters

in	<code>_i</code>	the index
in	<code>_v</code>	the position

10.157.4.41 void gazebo::common::SubMesh::SetVertexCount ( unsigned int *\_count* )

Resize the vertex array.

## Parameters

in	<code>_count</code>	the new size of the array
----	---------------------	---------------------------

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

- **Mesh.hh**

## 10.158 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.*

### 10.158.1 Detailed Description

Options for a subscription.

### 10.158.2 Constructor & Destructor Documentation

10.158.2.1 **gazebo::transport::SubscribeOptions::SubscribeOptions** ( ) `[inline]`

Constructor.

### 10.158.3 Member Function Documentation

10.158.3.1 **bool gazebo::transport::SubscribeOptions::GetLatching** ( ) `const [inline]`

Are we latching?

#### Returns

true if we're latching the latest message, false otherwise

10.158.3.2 **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

10.158.3.3 **NodePtr gazebo::transport::SubscribeOptions::GetNode** ( ) `const [inline]`

Get the node we're subscribed to.

#### Returns

The associated node

10.158.3.4 `std::string gazebo::transport::SubscribeOptions::GetTopic ( ) const` `[inline]`

Get the topic we're subscribed to.

#### Returns

The topic we're subscribed to

10.158.3.5 `template<class M> void gazebo::transport::SubscribeOptions::Init ( const std::string & _topic, NodePtr _node, bool _latching )` `[inline]`

Initialize the options.

#### Parameters

<code>in</code>	<code>_topic</code>	Topic we're subscribing to
<code>in, out</code>	<code>_node</code>	The associated node
<code>in</code>	<code>_latching</code>	If true, latch the latest message; if false, don't latch

References `gzthrow`, and `NULL`.

Referenced by `gazebo::transport::Node::Subscribe()`.

10.158.3.6 `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**

## 10.159 gazebo::transport::Subscriber Class Reference

**A** (p. 107) 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.*

## 10.159.1 Detailed Description

**A** (p. 107) subscriber to a topic.

## 10.159.2 Constructor & Destructor Documentation

### 10.159.2.1 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

### 10.159.2.2 virtual gazebo::transport::Subscriber::~Subscriber ( ) [virtual]

Destructor.

## 10.159.3 Member Function Documentation

### 10.159.3.1 unsigned int gazebo::transport::Subscriber::GetCallbackId ( ) const

### 10.159.3.2 std::string gazebo::transport::Subscriber::GetTopic ( ) const

Get the topic name.

Returns

The topic name

### 10.159.3.3 void gazebo::transport::Subscriber::SetCallbackId ( unsigned int *\_id* )

Referenced by gazebo::transport::Node::Subscribe().

### 10.159.3.4 void gazebo::transport::Subscriber::Unsubscribe ( ) const

Unsubscribe from the topic.

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

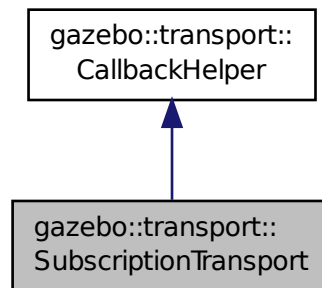
- **Subscriber.hh**

## 10.160 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

#### 10.160.1 Detailed Description

transport/transport.hh

Handles sending data over the wire to remote subscribers



## 10.160.2 Constructor & Destructor Documentation

### 10.160.2.1 gazebo::transport::SubscriptionTransport::SubscriptionTransport ( )

Constructor.

### 10.160.2.2 virtual gazebo::transport::SubscriptionTransport::~~SubscriptionTransport ( ) [virtual]

Destructor.

## 10.160.3 Member Function Documentation

### 10.160.3.1 const ConnectionPtr& gazebo::transport::SubscriptionTransport::GetConnection ( ) const

Get the connection we're using.

#### Returns

Pointer to the connection we're using

### 10.160.3.2 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. 155).

### 10.160.3.3 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

### 10.160.3.4 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. 155).

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

- **SubscriptionTransport.hh**

**10.161 gazebo::physics::SurfaceParams Class Reference**

**SurfaceParams** (p. 750) 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. 753)) of the friction pyramid.*
- double **kd**  
*spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 752) and **SurfaceParams::erp** (p. 752).*
- double **kp**  
*spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 752) and **SurfaceParams::erp** (p. 752).*
- 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.*

### 10.161.1 Detailed Description

**SurfaceParams** (p. 750) defines various Surface contact parameters.

These parameters defines the properties of a **physics::Contact** (p. 229) constraint.

### 10.161.2 Constructor & Destructor Documentation

#### 10.161.2.1 gazebo::physics::SurfaceParams::SurfaceParams ( )

Constructor.

#### 10.161.2.2 virtual gazebo::physics::SurfaceParams::~~SurfaceParams ( ) [virtual]

Destructor.

### 10.161.3 Member Function Documentation

#### 10.161.3.1 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.
----	-------------------	--

#### 10.161.3.2 virtual void gazebo::physics::SurfaceParams::Load ( sdf::ElementPtr \_sdf ) [virtual]

Load the contact params.

Parameters

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

#### 10.161.3.3 virtual void gazebo::physics::SurfaceParams::ProcessMsg ( const msgs::Surface & \_msg ) [virtual]

### 10.161.4 Member Data Documentation

#### 10.161.4.1 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](http://www.ode.org/ode-latest-userguide.html#sec_7_3_7)

#### 10.161.4.2 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](http://www.ode.org/ode-latest-userguide.html#sec_7_3_7)

#### 10.161.4.3 double gazebo::physics::SurfaceParams::cfm

Constraint Force Mixing parameter.

See for example [http://www.ode.org/ode-latest-userguide.html#sec\\_3\\_8\\_0](http://www.ode.org/ode-latest-userguide.html#sec_3_8_0) for more details.

#### 10.161.4.4 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](http://www.ode.org/ode-latest-userguide.html#sec_3_8_0) for more details.

#### 10.161.4.5 math::Vector3 gazebo::physics::SurfaceParams::fdir1

Primary friction direction for dry friction coefficient (**SurfaceParams::mu1** (p. 753)) 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](http://www.ode.org/ode-latest-userguide.html#sec_7_3_7)

#### 10.161.4.6 double gazebo::physics::SurfaceParams::kd

spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 752) and **SurfaceParams::erp** (p. 752).

##### See Also

See for example [http://www.ode.org/ode-latest-userguide.html#sec\\_3\\_8\\_2](http://www.ode.org/ode-latest-userguide.html#sec_3_8_2) for more details.

#### 10.161.4.7 double gazebo::physics::SurfaceParams::kp

spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p.752) and **SurfaceParams::erp** (p.752).

#### See Also

See for example [http://www.ode.org/ode-latest-userguide.html#sec\\_3\\_8\\_2](http://www.ode.org/ode-latest-userguide.html#sec_3_8_2) for more details.

#### 10.161.4.8 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](http://www.ode.org/ode-latest-userguide.html#sec_5_2_0))

#### 10.161.4.9 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](http://www.ode.org/ode-latest-userguide.html#sec_5_2_0))

#### 10.161.4.10 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](http://www.ode.org/ode-latest-userguide.html#sec_7_3_7)

#### 10.161.4.11 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](http://www.ode.org/ode-latest-userguide.html#sec_7_3_7)

#### 10.161.4.12 `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](http://www.ode.org/ode-latest-userguide.html#sec_7_3_7)

#### 10.161.4.13 `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](http://www.ode.org/ode-latest-userguide.html#sec_7_3_7)

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

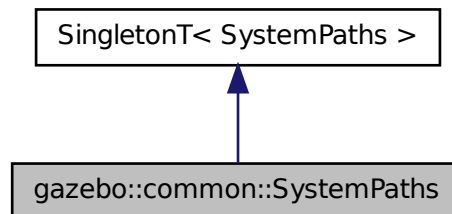
- **SurfaceParams.hh**

## 10.162 `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 **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 **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. 757)*
- bool **modelPathsFromEnv**  
*if true, call UpdateGazeboPaths() within GetGazeboPaths() (p. 757)*
- bool **ogrePathsFromEnv**  
*if true, call UpdateOgrePaths() within GetOgrePaths() (p. 758)*
- bool **pluginPathsFromEnv**  
*if true, call UpdatePluginPaths() within GetPluginPaths() (p. 758)*

## Additional Inherited Members

### 10.162.1 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. 103) RenderSystem\_GL.so et. al.
- SystemPaths::pluginPaths - plugin library paths for common::WorldPlugin

## 10.162.2 Member Function Documentation

10.162.2.1 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
----	--------------	----------------------

10.162.2.2 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
----	--------------	----------------------

10.162.2.3 void gazebo::common::SystemPaths::AddPluginPaths ( const std::string & *\_path* )

Add colon delimited paths to plugins.

### Parameters

in	<i>_path</i>	the directory to add
----	--------------	----------------------

10.162.2.4 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
----	----------------	-------------------

10.162.2.5 void gazebo::common::SystemPaths::ClearGazeboPaths ( )

clear out SystemPaths::gazeboPaths

10.162.2.6 void gazebo::common::SystemPaths::ClearOgrePaths ( )

clear out SystemPaths::ogrePaths

10.162.2.7 void gazebo::common::SystemPaths::ClearPluginPaths ( )

clear out SystemPaths::pluginPaths



10.162.2.8 `std::string gazebo::common::SystemPaths::FindFile ( const std::string & _filename, bool _searchLocalPath = true )`

Find a file in the gazebo paths.

#### Parameters

<code>in</code>	<code>_filename</code>	Name of the file to find.
<code>in</code>	<code>_searchLocal-Path</code>	True to search in the current working directory.

#### Returns

Returns full path name to file

10.162.2.9 `std::string gazebo::common::SystemPaths::FindFileURI ( const std::string & _uri )`

Find a file or path using a URI.

#### Parameters

<code>in</code>	<code>_uri</code>	the uniform resource identifier
-----------------	-------------------	---------------------------------

#### Returns

Returns full path name to file

10.162.2.10 `const std::list<std::string>& gazebo::common::SystemPaths::GetGazeboPaths ( )`

Get the gazebo install paths.

#### Returns

a list of paths

10.162.2.11 `std::string gazebo::common::SystemPaths::GetLogPath ( ) const`

Get the log path.

#### Returns

the path

10.162.2.12 `const std::list<std::string>& gazebo::common::SystemPaths::GetModelPaths ( )`

Get the model paths.

#### Returns

a list of paths

10.162.2.13 `const std::list<std::string>& gazebo::common::SystemPaths::GetOgrePaths ( )`

Get the ogre install paths.

Returns

a list of paths

10.162.2.14 `const std::list<std::string>& gazebo::common::SystemPaths::GetPluginPaths ( )`

Get the plugin paths.

Returns

a list of paths

10.162.2.15 `std::string gazebo::common::SystemPaths::GetWorldPathExtension ( )`

Returns the world path extension.

Returns

Right now, it just returns "/worlds"

### 10.162.3 Member Data Documentation

10.162.3.1 `bool gazebo::common::SystemPaths::gazeboPathsFromEnv`

if true, call `UpdateGazeboPaths()` within **GetGazeboPaths()** (p. 757)

10.162.3.2 `bool gazebo::common::SystemPaths::modelPathsFromEnv`

if true, call `UpdateGazeboPaths()` within **GetGazeboPaths()** (p. 757)

10.162.3.3 `bool gazebo::common::SystemPaths::ogrePathsFromEnv`

if true, call `UpdateOgrePaths()` within **GetOgrePaths()** (p. 758)

10.162.3.4 `bool gazebo::common::SystemPaths::pluginPathsFromEnv`

if true, call `UpdatePluginPaths()` within **GetPluginPaths()** (p. 758)

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

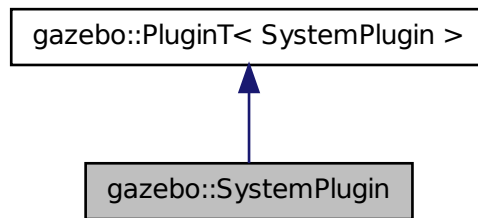
- **SystemPaths.hh**

## 10.163 gazebo::SystemPlugin Class Reference

**A** (p. 107) 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

#### 10.163.1 Detailed Description

**A** (p. 107) plugin loaded within the gzserver on startup.

See [reference](#).

**Todo** how to make doxygen reference to the file gazebo.cc::g\_plugins?

#### 10.163.2 Constructor & Destructor Documentation

##### 10.163.2.1 gazebo::SystemPlugin::SystemPlugin ( ) [inline]

Constructor.

References gazebo::SYSTEM\_PLUGIN, and gazebo::PluginT< SystemPlugin >::type.

10.163.2.2 virtual gazebo::SystemPlugin::~~SystemPlugin ( ) [inline],[virtual]

Destructor.

### 10.163.3 Member Function Documentation

10.163.3.1 virtual void gazebo::SystemPlugin::Init ( ) [inline],[virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.163.3.2 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.

10.163.3.3 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**

## 10.164 gazebo::common::Time Class Reference

**A** (p. 107) **Time** (p. 760) class, can be used to hold wall- or sim-time.

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

### Public Member Functions

- **Time** ()  
*Constructors.*
- **Time** (const **Time** &\_time)  
*Copy constructor.*
- **Time** (const struct timeval &\_tv)  
*Constructor.*
- **Time** (const struct timespec &\_tv)  
*Constructor.*

- **Time** (int32\_t \_sec, int32\_t \_nsec)  
*Constructor.*
- **Time** (double \_time)  
*Constructor.*
- virtual  $\sim$ **Time** ()  
*Destructor.*
- double **Double** () const  
*Get the time as a double.*
- float **Float** () const  
*Get the time as a float.*
- bool **operator!=** (const struct timeval &\_tv) const  
*Equal to operator.*
- bool **operator!=** (const struct timespec &\_tv) const  
*Equal to operator.*
- bool **operator!=** (const **Time** &\_time) const  
*Equal to operator.*
- bool **operator!=** (double \_time) const  
*Equal to operator.*
- **Time operator\*** (const struct timeval &\_tv) const  
*Multiplication operator.*
- **Time operator\*** (const struct timespec &\_tv) const  
*Multiplication operator.*
- **Time operator\*** (const **Time** &\_time) const  
*Multiplication operators.*
- const **Time & operator\*=** (const struct timeval &\_tv)  
*Multiplication assignment operator.*
- const **Time & operator\*=** (const struct timespec &\_tv)  
*Multiplication assignment operator.*
- const **Time & operator\*=** (const **Time** &\_time)  
*Multiplication operators.*
- **Time operator+** (const struct timeval &\_tv) const  
*Addition operators.*
- **Time operator+** (const struct timespec &\_tv) const  
*Addition operators.*
- **Time operator+** (const **Time** &\_time) const  
*Addition operators.*
- const **Time & operator+=** (const struct timeval &\_tv)  
*Addition assignment operator.*
- const **Time & operator+=** (const struct timespec &\_tv)  
*Addition assignment operator.*
- const **Time & operator+=** (const **Time** &\_time)  
*Addition assignment operator.*
- **Time operator-** (const struct timeval &\_tv) const  
*Subtraction operator.*
- **Time operator-** (const struct timespec &\_tv) const  
*Subtraction operator.*
- **Time operator-** (const **Time** &\_time) const

- Subtraction operator.*

  - const **Time & operator-=** (const struct timeval &\_tv)
- Subtraction assignment operator.*

  - const **Time & operator-=** (const struct timespec &\_tv)
- Subtraction assignment operator.*

  - const **Time & operator-=** (const **Time** &\_time)
- Subtraction assignment operator.*

  - **Time operator/** (const struct timeval &\_tv) const
- Division operator.*

  - **Time operator/** (const struct timespec &\_tv) const
- Division operator.*

  - **Time operator/** (const **Time** &\_time) const
- Division operator.*

  - const **Time & operator/=** (const struct timeval &\_tv)
- Division assignment operator.*

  - const **Time & operator/=** (const struct timespec &\_tv)
- Division assignment operator.*

  - const **Time & operator/=** (const **Time** &time)
- Division assignment operator.*

  - bool **operator<** (const struct timeval &\_tv) const
- Less than operator.*

  - bool **operator<** (const struct timespec &\_tv) const
- Less than operator.*

  - bool **operator<** (const **Time** &\_time) const
- Less than operator.*

  - bool **operator<** (double \_time) const
- Less than operator.*

  - bool **operator<=** (const struct timeval &\_tv) const
- Less than or equal to operator.*

  - bool **operator<=** (const struct timespec &\_tv) const
- Less than or equal to operator.*

  - bool **operator<=** (const **Time** &\_time) const
- Less than or equal to operator.*

  - bool **operator<=** (double \_time) const
- Less than or equal to operator.*

  - **Time & operator=** (const struct timeval &\_tv)
- Assignment operator.*

  - **Time & operator=** (const struct timespec &\_tv)
- Assignment operator.*

  - **Time & operator=** (const **Time** &\_time)
- Assignment operator.*

  - bool **operator==** (const struct timeval &\_tv) const
- Equal to operator.*

  - bool **operator==** (const struct timespec &\_tv) const
- Equal to operator.*

  - bool **operator==** (const **Time** &\_time) const
- Equal to operator.*

- bool **operator==** (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 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)  
*Nano sleep.*
- static double **SecToNano** (double \_sec)  
*Convert seconds to nanoseconds.*

### Public Attributes

- int32\_t **nsec**  
*Microseconds.*
- int32\_t **sec**  
*Seconds.*

## 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.*

### 10.164.1 Detailed Description

**A** (p. 107) **Time** (p. 760) class, can be used to hold wall- or sim-time. stored as sec and nano-sec.

### 10.164.2 Constructor & Destructor Documentation

#### 10.164.2.1 gazebo::common::Time::Time ( )

Constructors.

#### 10.164.2.2 gazebo::common::Time::Time ( const Time & \_time )

Copy constructor.

##### Parameters

in	<i>time</i>	<b>Time</b> (p. 760) to copy
----	-------------	------------------------------

#### 10.164.2.3 gazebo::common::Time::Time ( const struct timeval & \_tv )

Constructor.

##### Parameters

in	<i>_tv</i>	<b>Time</b> (p. 760) to initialize to
----	------------	---------------------------------------

#### 10.164.2.4 gazebo::common::Time::Time ( const struct timespec & \_tv )

Constructor.

##### Parameters

in	<i>_tv</i>	<b>Time</b> (p. 760) to initialize to
----	------------	---------------------------------------

#### 10.164.2.5 gazebo::common::Time::Time ( int32\_t \_sec, int32\_t \_nsec )

Constructor.



## Parameters

in	<code>_sec</code>	Seconds
in	<code>_nsec</code>	Nanoseconds

10.164.2.6 gazebo::common::Time::Time ( double *time* )

Constructor.

## Parameters

in	<code>_time</code>	<b>Time</b> (p. 760) in double format sec.nsec
----	--------------------	--

## 10.164.2.7 virtual gazebo::common::Time::~~Time ( ) [virtual]

Destructor.

## 10.164.3 Member Function Documentation

## 10.164.3.1 double gazebo::common::Time::Double ( ) const

Get the time as a double.

## Returns

**Time** (p. 760) as a double in seconds

## 10.164.3.2 float gazebo::common::Time::Float ( ) const

Get the time as a float.

## Returns

**Time** (p. 760) as a float in seconds

## 10.164.3.3 static const Time&amp; gazebo::common::Time::GetWallTime ( ) [static]

Get the wall time.

## Returns

the current time

10.164.3.4 static double gazebo::common::Time::MicToNano ( double *ms* ) [inline],[static]

Convert microseconds to nanoseconds.

## Parameters

<code>_ms</code>	microseconds
------------------	--------------

**Returns**

nanoseconds

10.164.3.5 `static double gazebo::common::Time::MilToNano ( double _ms ) [inline],[static]`

Convert milliseconds to nanoseconds.

**Parameters**

<code>in</code>	<code>_ms</code>	milliseconds
-----------------	------------------	--------------

**Returns**

nanoseconds

10.164.3.6 `static Time gazebo::common::Time::MSleep ( unsigned int _ms ) [static]`

Millisecond sleep.

**Parameters**

<code>in</code>	<code>_ms</code>	milliseconds
-----------------	------------------	--------------

10.164.3.7 `static Time gazebo::common::Time::NSleep ( unsigned int _ns ) [static]`

Nano sleep.

**Parameters**

<code>in</code>	<code>_ns</code>	nanoseconds
-----------------	------------------	-------------

10.164.3.8 `static Time gazebo::common::Time::NSleep ( Time _time ) [static]`

Nano sleep.

**Parameters**

<code>in</code>	<code>_time</code>	is a <b>Time</b> (p. 760)
-----------------	--------------------	---------------------------

10.164.3.9 `bool gazebo::common::Time::operator!= ( const struct timeval & _tv ) const`

Equal to operator.

## Parameters

in	_tv	the time to compare to
----	-----	------------------------

## Returns

true if values are the same, false otherwise

10.164.3.10 `bool gazebo::common::Time::operator!=( const struct timespec & _tv ) const`

Equal to operator.

## Parameters

in	_tv	the time to compare to
----	-----	------------------------

## Returns

true if values are the same, false otherwise

10.164.3.11 `bool gazebo::common::Time::operator!=( const Time & _time ) const`

Equal to operator.

## Parameters

in	_time	the time to compare to
----	-------	------------------------

## Returns

true if values are the same, false otherwise

10.164.3.12 `bool gazebo::common::Time::operator!=( double _time ) const`

Equal to operator.

## Parameters

in	_time	the time to compare to
----	-------	------------------------

## Returns

true if values are the same, false otherwise

10.164.3.13 `Time gazebo::common::Time::operator*( const struct timeval & _tv ) const`

Multiplication operator.

## Parameters

<code>in</code>	<code>_tv</code>	The scaling duration
-----------------	------------------	----------------------

## Returns

**Time** (p. 760) instance

10.164.3.14 **Time** gazebo::common::Time::operator\* ( const struct timespec & *\_tv* ) const

Multiplication operator.

## Parameters

<code>in</code>	<code>_tv</code>	the scaling duration
-----------------	------------------	----------------------

## Returns

**Time** (p. 760) instance

10.164.3.15 **Time** gazebo::common::Time::operator\* ( const Time & *\_time* ) const

Multiplication operators.

## Parameters

<code>in</code>	<code>_time</code>	the scaling factor
-----------------	--------------------	--------------------

## Returns

a scaled **Time** (p. 760) instance

10.164.3.16 `const Time&` gazebo::common::Time::operator\*=( const struct timeval & *\_tv* )

Multiplication assignment operator.

## Parameters

<code>in</code>	<code>_tv</code>	the scaling duration
-----------------	------------------	----------------------

## Returns

a reference to this instance

10.164.3.17 `const Time&` gazebo::common::Time::operator\*=( const struct timespec & *\_tv* )

Multiplication assignment operator.

## Parameters

in	<code>_tv</code>	the scaling duration
----	------------------	----------------------

## Returns

a reference to this instance

10.164.3.18 `const Time& gazebo::common::Time::operator*=( const Time & _time )`

Multiplication operators.

## Parameters

in	<code>_time</code>	scale factor
----	--------------------	--------------

## Returns

a scaled **Time** (p. 760) instance

10.164.3.19 `Time gazebo::common::Time::operator+( const struct timeval & _tv ) const`

Addition operators.

## Parameters

in	<code>_tv</code>	the time to add
----	------------------	-----------------

## Returns

a **Time** (p. 760) instance

10.164.3.20 `Time gazebo::common::Time::operator+( const struct timespec & _tv ) const`

Addition operators.

## Parameters

in	<code>_tv</code>	the time to add
----	------------------	-----------------

## Returns

a **Time** (p. 760) instance

10.164.3.21 `Time gazebo::common::Time::operator+( const Time & _time ) const`

Addition operators.

## Parameters

in	<i>_time</i>	The time to add
----	--------------	-----------------

## Returns

a **Time** (p. 760) instance

10.164.3.22 `const Time& gazebo::common::Time::operator+=( const struct timeval & _tv )`

Addition assignment operator.

## Parameters

in	<i>_tv</i>	the time to add
----	------------	-----------------

## Returns

a reference to this instance

10.164.3.23 `const Time& gazebo::common::Time::operator+=( const struct timespec & _tv )`

Addition assignment operator.

## Parameters

in	<i>_tv</i>	the time to add
----	------------	-----------------

## Returns

a reference to this instance

10.164.3.24 `const Time& gazebo::common::Time::operator+=( const Time & _time )`

Addition assignemtn operator.

## Parameters

in	<i>_time</i>	The time to add
----	--------------	-----------------

## Returns

a **Time** (p. 760) instance

10.164.3.25 `Time gazebo::common::Time::operator-( const struct timeval & _tv ) const`

Subtraction operator.

## Parameters

in	_tv	The time to subtract
----	-----	----------------------

## Returns

a **Time** (p. 760) instance

## 10.164.3.26 Time gazebo::common::Time::operator- ( const struct timespec &amp; \_tv ) const

Subtraction operator.

## Parameters

in	_tv	The time to subtract
----	-----	----------------------

## Returns

a **Time** (p. 760) instance

## 10.164.3.27 Time gazebo::common::Time::operator- ( const Time &amp; \_time ) const

Subtraction operator.

## Parameters

in	_time	The time to subtract
----	-------	----------------------

## Returns

a **Time** (p. 760) instance

## 10.164.3.28 const Time&amp; gazebo::common::Time::operator-= ( const struct timeval &amp; \_tv )

Subtraction assignment operator.

## Parameters

in	_tv	The time to subtract
----	-----	----------------------

## Returns

a **Time** (p. 760) instance

## 10.164.3.29 const Time&amp; gazebo::common::Time::operator-= ( const struct timespec &amp; \_tv )

Subtraction assignment operator.

## Parameters

in	<code>_tv</code>	The time to subtract
----	------------------	----------------------

## Returns

a **Time** (p. 760) instance

10.164.3.30 `const Time& gazebo::common::Time::operator-= ( const Time & _time )`

Subtraction assignment operator.

## Parameters

in	<code>_time</code>	The time to subtract
----	--------------------	----------------------

## Returns

a reference to this instance

10.164.3.31 `Time gazebo::common::Time::operator/ ( const struct timeval & _tv ) const`

Division operator.

## Parameters

in	<code>_tv</code>	a timeval divisor
----	------------------	-------------------

## Returns

a **Time** (p. 760) instance

10.164.3.32 `Time gazebo::common::Time::operator/ ( const struct timespec & _tv ) const`

Division operator.

## Parameters

in	<code>_tv</code>	a timespec divisor
----	------------------	--------------------

## Returns

a **Time** (p. 760) instance

10.164.3.33 `Time gazebo::common::Time::operator/ ( const Time & _time ) const`

Division operator.



## Parameters

<i>in</i>	<i>_time</i>	the divisor
-----------	--------------	-------------

## Returns

a **Time** (p. 760) instance

10.164.3.34 `const Time& gazebo::common::Time::operator/= ( const struct timeval & _tv )`

Division assignment operator.

## Parameters

<i>in</i>	<i>_tv</i>	a divisor
-----------	------------	-----------

## Returns

a **Time** (p. 760) instance

10.164.3.35 `const Time& gazebo::common::Time::operator/= ( const struct timespec & _tv )`

Division assignment operator.

## Parameters

<i>in</i>	<i>_tv</i>	a divisor
-----------	------------	-----------

## Returns

a **Time** (p. 760) instance

10.164.3.36 `const Time& gazebo::common::Time::operator/= ( const Time & time )`

Division assignment operator.

## Parameters

<i>in</i>	<i>time</i>	the divisor
-----------	-------------	-------------

## Returns

a **Time** (p. 760) instance

10.164.3.37 `bool gazebo::common::Time::operator< ( const struct timeval & _tv ) const`

Less than operator.

## Parameters

in	<code>_tv</code>	the time to compare with
----	------------------	--------------------------

## Returns

true if tv is shorter than this, false otherwise

10.164.3.38 `bool gazebo::common::Time::operator< ( const struct timespec & _tv ) const`

Less than operator.

## Parameters

in	<code>_tv</code>	the time to compare with
----	------------------	--------------------------

## Returns

true if tv is shorter than this, false otherwise

10.164.3.39 `bool gazebo::common::Time::operator< ( const Time & _time ) const`

Less than operator.

## Parameters

in	<code>_time</code>	the time to compare with
----	--------------------	--------------------------

## Returns

true if time is shorter than this, false otherwise

10.164.3.40 `bool gazebo::common::Time::operator< ( double _time ) const`

Less than operator.

## Parameters

in	<code>_time</code>	the time to compare with
----	--------------------	--------------------------

## Returns

true if time is shorter than this, false otherwise

10.164.3.41 `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

10.164.3.42 `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

10.164.3.43 `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

10.164.3.44 `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

10.164.3.45 `Time& gazebo::common::Time::operator= ( const struct timeval & _tv )`

Assignment operator.

## Parameters

in	<code>_tv</code>	the new time
----	------------------	--------------

## Returns

a reference to this instance

10.164.3.46 `Time& gazebo::common::Time::operator= ( const struct timespec & _tv )`

Assignment operator.

## Parameters

in	<code>_tv</code>	the new time
----	------------------	--------------

## Returns

a reference to this instance

10.164.3.47 `Time& gazebo::common::Time::operator= ( const Time & _time )`

Assignment operator.

## Parameters

in	<code>_time</code>	the new time
----	--------------------	--------------

## Returns

a reference to this instance

10.164.3.48 `bool gazebo::common::Time::operator== ( const struct timeval & _tv ) const`

Equal to operator.

## Parameters

in	<code>_tv</code>	the time to compare to
----	------------------	------------------------

## Returns

true if values are the same, false otherwise

10.164.3.49 `bool gazebo::common::Time::operator== ( const struct timespec & _tv ) const`

Equal to operator.

## Parameters

in	<code>_tv</code>	the time to compare to
----	------------------	------------------------

## Returns

true if values are the same, false otherwise

10.164.3.50 `bool gazebo::common::Time::operator==( const Time & _time ) const`

Equal to operator.

## Parameters

in	<code>_time</code>	the time to compare to
----	--------------------	------------------------

## Returns

true if values are the same, false otherwise

10.164.3.51 `bool gazebo::common::Time::operator==( double _time ) const`

Equal to operator.

## Parameters

in	<code>_time</code>	the time to compare to
----	--------------------	------------------------

## Returns

true if values are the same, false otherwise

10.164.3.52 `bool gazebo::common::Time::operator>( const struct timeval & _tv ) const`

Greater than operator.

## Parameters

in	<code>_tv</code>	the time to compare with
----	------------------	--------------------------

## Returns

true if time is greater than this, false otherwise

10.164.3.53 `bool gazebo::common::Time::operator>( const struct timespec & _tv ) const`

Greater than operator.

## Parameters

in	_tv	the time to compare with
----	-----	--------------------------

## Returns

true if time is greater than this, false otherwise

10.164.3.54 `bool gazebo::common::Time::operator> ( const Time & _time ) const`

Greater than operator.

## Parameters

in	_time	the time to compare with
----	-------	--------------------------

## Returns

true if time is greater than this, false otherwise

10.164.3.55 `bool gazebo::common::Time::operator> ( double _time ) const`

Greater than operator.

## Parameters

in	_time	the time to compare with
----	-------	--------------------------

## Returns

true if time is greater than this, false otherwise

10.164.3.56 `bool gazebo::common::Time::operator>= ( const struct timeval & _tv ) const`

Greater than or equal operator.

## Parameters

in	_tv	the time to compare with
----	-----	--------------------------

## Returns

true if tv is greater than or equal to this, false otherwise

10.164.3.57 `bool gazebo::common::Time::operator>= ( const struct timespec & _tv ) const`

Greater than or equal operator.

## Parameters

in	<code>_tv</code>	the time to compare with
----	------------------	--------------------------

## Returns

true if tv is greater than or equal to this, false otherwise

10.164.3.58 `bool gazebo::common::Time::operator>= ( const Time & _time ) const`

Greater than or equal operator.

## Parameters

in	<code>_time</code>	the time to compare with
----	--------------------	--------------------------

## Returns

true if time is greater than or equal to this, false otherwise

10.164.3.59 `bool gazebo::common::Time::operator>= ( double _time ) const`

Greater than or equal operator.

## Parameters

in	<code>_time</code>	the time to compare with
----	--------------------	--------------------------

## Returns

true if time is greater than or equal to this, false otherwise

10.164.3.60 `static double gazebo::common::Time::SecToNano ( double _sec ) [inline],[static]`

Convert seconds to nanoseconds.

## Parameters

in	<code>_sec</code>	duration in seconds
----	-------------------	---------------------

## Returns

nanoseconds

10.164.3.61 `void gazebo::common::Time::Set ( int32_t _sec, int32_t _nsec )`

Set to sec and nsec.

## Parameters

in	<code>_sec</code>	Seconds
in	<code>_nsec</code>	Nanoseconds

10.164.3.62 `void gazebo::common::Time::Set ( double _seconds )`

Set to seconds.

## Parameters

in	<code>_seconds</code>	Number of seconds
----	-----------------------	-------------------

10.164.3.63 `void gazebo::common::Time::SetToWallTime ( )`

Set the time to the wall time.

## 10.164.4 Friends And Related Function Documentation

10.164.4.1 `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

10.164.4.2 `std::istream& operator>> ( std::istream & _in, gazebo::common::Time & _time )` [*friend*]

Stream extraction operator.

## Parameters

in	<code>_in</code>	the input stream
in	<code>_time</code>	time to read from to the stream

## Returns

the input stream

## 10.164.5 Member Data Documentation

10.164.5.1 `int32_t gazebo::common::Time::nsec`

Microseconds.



10.164.5.2 int32\_t gazebo::common::Time::sec

Seconds.

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

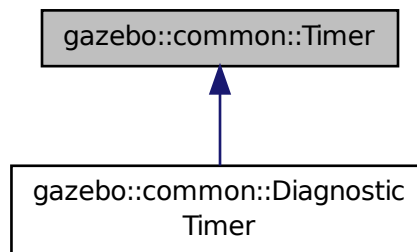
- **Time.hh**

## 10.165 gazebo::common::Timer Class Reference

**A** (p. 107) 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.*
- void **Start** ()  
*Start the timer.*

### Friends

- std::ostream & **operator**<< (std::ostream &out, const **gazebo::common::Timer** &t)  
*stream operator friendly*

### 10.165.1 Detailed Description

**A** (p. 107) timer class, used to time things in real world walltime.

## 10.165.2 Constructor & Destructor Documentation

### 10.165.2.1 gazebo::common::Timer::Timer ( )

Constructor.

### 10.165.2.2 virtual gazebo::common::Timer::~~Timer ( ) [virtual]

Destructor.

## 10.165.3 Member Function Documentation

### 10.165.3.1 Time gazebo::common::Timer::GetElapsed ( ) const

Get the elapsed time.

#### Returns

The time

### 10.165.3.2 void gazebo::common::Timer::Start ( )

Start the timer.

Referenced by gazebo::common::DiagnosticTimer::DiagnosticTimer().

## 10.165.4 Friends And Related Function Documentation

### 10.165.4.1 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:

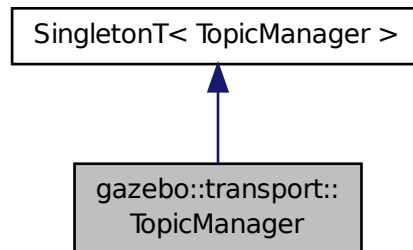
- **Timer.hh**

## 10.166 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. 107) map of string->list of **Node** (p. 515) 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)  
*Advertise on a topic.*
- void **ClearBuffers** ()  
*Clear all buffers.*
- void **ConnectPubToSub** (const std::string &\_topic, const **SubscriptionTransportPtr** &\_sublink)  
***Connection** (p. 217) a local **Publisher** (p. 595) to a remote **Subscriber** (p. 746).*
- 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. 746) to a remote **Publisher** (p. 595).*
- 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 unpause 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

### 10.166.1 Detailed Description

Manages topics and their subscriptions.

### 10.166.2 Member Typedef Documentation

10.166.2.1 typedef std::map<std::string, std::list<NodePtr> > gazebo::transport::TopicManager::SubNodeMap

**A** (p. 107) map of string->list of **Node** (p. 515) pointers.

### 10.166.3 Member Function Documentation

10.166.3.1 void gazebo::transport::TopicManager::AddNode ( NodePtr \_node )

Add a node to the manager.

#### Parameters

in, out	_node	The node to be added
---------	-------	----------------------

10.166.3.2 `template<typename M > PublisherPtr gazebo::transport::TopicManager::Advertise ( const std::string & _topic, unsigned int _queueLimit ) [inline]`

Advertise on a topic.

#### Parameters

<code>in</code>	<code>_topic</code>	The name of the topic
<code>in</code>	<code>_queueLimit</code>	The maximum number of outgoing messages to queue

#### Returns

Pointer to the newly created **Publisher** (p. 595)

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()`.

10.166.3.3 `void gazebo::transport::TopicManager::ClearBuffers ( )`

Clear all buffers.

10.166.3.4 `void gazebo::transport::TopicManager::ConnectPubToSub ( const std::string & _topic, const SubscriptionTransportPtr & _sublink )`

**Connection** (p. 217) a local **Publisher** (p. 595) to a remote **Subscriber** (p. 746).

#### Parameters

<code>in</code>	<code>_topic</code>	The topic to use
<code>in</code>	<code>_sublink</code>	The subscription transport object to use

10.166.3.5 `void gazebo::transport::TopicManager::ConnectSubscribers ( const std::string & _topic )`

Connect all subscribers on a topic to known publishers.

#### Parameters

<code>in</code>	<code>_topic</code>	The topic to be connected
-----------------	---------------------	---------------------------

10.166.3.6 `void gazebo::transport::TopicManager::ConnectSubToPub ( const msgs::Publish & _pub )`

Connect a local **Subscriber** (p. 746) to a remote **Publisher** (p. 595).

#### Parameters

<code>in</code>	<code>_pub</code>	The publish object to use
-----------------	-------------------	---------------------------

10.166.3.7 `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

10.166.3.8 `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

10.166.3.9 **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()`.

10.166.3.10 `void gazebo::transport::TopicManager::Fini ( )`

Finalize the manager.

10.166.3.11 `std::map<std::string, std::list<std::string> > gazebo::transport::TopicManager::GetAdvertisedTopics ( ) const`

Get a list of all the topics.

#### Returns

**A** (p. 107) map where keys are message types, and values are a list of topic names.

#### See Also

`transport::GetAdvertisedTopics`

10.166.3.12 `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
-----	--------------------------	---

10.166.3.13 `void gazebo::transport::TopicManager::Init ( )`

Initialize the manager.

10.166.3.14 `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

10.166.3.15 `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
----	---------------------	--

10.166.3.16 `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
----	-----------------------	---

10.166.3.17 `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. 595) 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.

10.166.3.18 `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
-----------------	--------------------	-------------------------------

10.166.3.19 `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
-----------------	------------------	----------------------------------

10.166.3.20 `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

10.166.3.21 `void gazebo::transport::TopicManager::Unadvertise ( const std::string & _topic )`

Unadvertise a topic.

## Parameters

<code>in</code>	<code>_topic</code>	The topic to be unadvertised
-----------------	---------------------	------------------------------

10.166.3.22 `void gazebo::transport::TopicManager::Unsubscribe ( const std::string & _topic, const NodePtr & _sub )`

Unsubscribe from a topic.

Use a **Subscriber** (p. 746) rather than calling this function directly



## Parameters

in	<code>_topic</code>	The topic to unsubscribe from
in	<code>_sub</code>	The node to unsubscribe

10.166.3.23 **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**

## 10.167 gazebo::physics::TrajectoryInfo Struct Reference

```
#include <Actor.hh>
```

## Public Attributes

- double **duration**
- double **endTime**
- unsigned int **id**
- double **startTime**
- bool **translated**
- std::string **type**

### 10.167.1 Member Data Documentation

10.167.1.1 double gazebo::physics::TrajectoryInfo::duration

10.167.1.2 double gazebo::physics::TrajectoryInfo::endTime

10.167.1.3 unsigned int gazebo::physics::TrajectoryInfo::id

10.167.1.4 double gazebo::physics::TrajectoryInfo::startTime

10.167.1.5 bool gazebo::physics::TrajectoryInfo::translated

10.167.1.6 `std::string gazebo::physics::TrajectoryInfo::type`

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

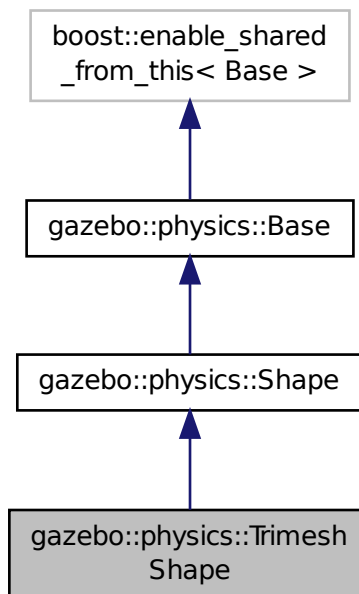
- **Actor.hh**

## 10.168 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

#### 10.168.1 Detailed Description

Triangle mesh collision shape.

#### 10.168.2 Constructor & Destructor Documentation

10.168.2.1 gazebo::physics::TrimeshShape::TrimeshShape ( CollisionPtr *\_parent* ) [explicit]

Constructor.

##### Parameters

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

10.168.2.2 virtual gazebo::physics::TrimeshShape::~~TrimeshShape ( ) [virtual]

Destructor.

#### 10.168.3 Member Function Documentation

10.168.3.1 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. 695).

10.168.3.2 `std::string gazebo::physics::TrimeshShape::GetFilename ( ) const`

Get the filename of the mesh data.

#### Returns

The filename of the mesh data.

10.168.3.3 `virtual math::Vector3 gazebo::physics::TrimeshShape::GetSize ( ) const [virtual]`

Get the size of the triangle mesh.

#### Returns

The size of the triangle mesh.

10.168.3.4 `virtual void gazebo::physics::TrimeshShape::Init ( ) [virtual]`

Initialize the shape.

Implements **gazebo::physics::Shape** (p. 695).

10.168.3.5 `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. 695).

10.168.3.6 `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.
-----------------	------------------------	---

10.168.3.7 `void gazebo::physics::TrimeshShape::SetScale ( const math::Vector3 & _scale )`

Set the scaling factor.

#### Parameters

<code>in</code>	<code>_scale</code>	Scaling factor.
-----------------	---------------------	-----------------

10.168.3.8 `virtual void gazebo::physics::TrimeshShape::Update ( ) [inline],[virtual]`

Update the tri mesh.

Reimplemented from `gazebo::physics::Base` (p. 143).

#### 10.168.4 Member Data Documentation

10.168.4.1 `const common::Mesh* gazebo::physics::TrimeshShape::mesh [protected]`

Pointer to the mesh data.

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

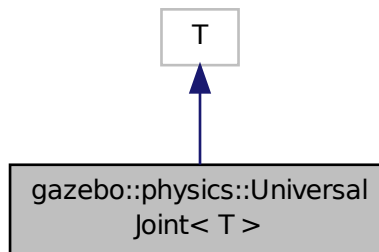
- `TrimeshShape.hh`

## 10.169 gazebo::physics::UniversalJoint< T > Class Template Reference

**A** (p. 107) 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. 793).*

### 10.169.1 Detailed Description

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

**A** (p. 107) universal joint.

### 10.169.2 Constructor & Destructor Documentation

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

Constructor.

#### Parameters

in	_parent	Parent link of the univeral joint.
----	---------	------------------------------------

References gazebo::physics::Base::UNIVERSAL\_JOINT.

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

Destuctor.

### 10.169.3 Member Function Documentation

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

10.169.3.2 `template<class T > virtual void gazebo::physics::UniversalJoint< T >::Load ( sdf::ElementPtr _sdf )`  
`[inline], [virtual]`

Load a **UniversalJoint** (p. 793).

#### Parameters

in	_sdf	SDF values to load from.
----	------	--------------------------

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

- **UniversalJoint.hh**

## 10.170 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.*

## 10.170.1 Constructor & Destructor Documentation

### 10.170.1.1 urdf2gazebo::URDF2Gazebo::URDF2Gazebo ( )

constructor

### 10.170.1.2 urdf2gazebo::URDF2Gazebo::~~URDF2Gazebo ( )

destructor

## 10.170.2 Member Function Documentation

### 10.170.2.1 TiXmlDocument urdf2gazebo::URDF2Gazebo::InitModelDoc ( TiXmlDocument \* \_xmlDoc )

convert urdf xml document string to sdf xml document

Parameters

in	_xmlDoc	a tinyxml document containing the urdf model
----	---------	--

Returns

a tinyxml document containing sdf of the model

### 10.170.2.2 TiXmlDocument urdf2gazebo::URDF2Gazebo::InitModelFile ( const std::string & \_filename )

convert urdf file to sdf xml document

Parameters

in	_urdfStr	a string containing filename of the urdf model
----	----------	--

Returns

a tinyxml document containing sdf of the model

```
10.170.2.3 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:

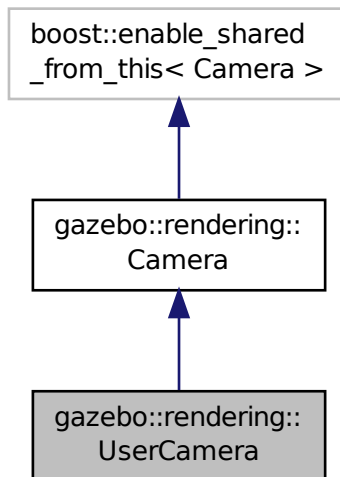
- `parser_urdf.hh`

## 10.171 gazebo::rendering::UserCamera Class Reference

**A** (p. 107) 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.*
  - 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

### 10.171.1 Detailed Description

**A** (p. 107) camera used for user visualization of a scene.

### 10.171.2 Constructor & Destructor Documentation

#### 10.171.2.1 gazebo::rendering::UserCamera::UserCamera ( const std::string & \_name, ScenePtr \_scene )

Constructor.

#### Parameters

in	<code>_name</code>	Name of the camera.
in	<code>_scene</code>	<b>Scene</b> (p. 651) to put the camera in.

#### 10.171.2.2 virtual gazebo::rendering::UserCamera::~~UserCamera ( ) [virtual]

Destructor.

### 10.171.3 Member Function Documentation

#### 10.171.3.1 virtual void gazebo::rendering::UserCamera::AnimationComplete ( ) [protected],[virtual]

Internal function used to indicate that an animation has completed.

Reimplemented from **gazebo::rendering::Camera** (p. 164).

10.171.3.2 `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

<code>in</code>	<code>_visual</code>	The visual to attach to.
<code>in</code>	<code>_inheritOrientation</code>	True if the camera should also rotate when the visual rotates.
<code>in</code>	<code>_minDist</code>	Minimum distance the camera can get to the visual.
<code>in</code>	<code>_maxDist</code>	Maximum distance the camera can get from the visual.

#### Returns

True if successfully attach to the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 165).

10.171.3.3 `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

<code>in</code>	<code>_value</code>	True to enable viewcontroller, False to disable.
-----------------	---------------------	--

10.171.3.4 `void gazebo::rendering::UserCamera::Fini ( ) [virtual]`

Finalize.

Reimplemented from **gazebo::rendering::Camera** (p. 166).

10.171.3.5 `float gazebo::rendering::UserCamera::GetAvgFPS ( ) const`

Get the average frames per second.

#### Returns

The average rendering frames per second

10.171.3.6 **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. 337).

10.171.3.7 `float gazebo::rendering::UserCamera::GetTriangleCount ( ) const`

Get the triangle count.

**Returns**

The number of triangles currently being rendered.

10.171.3.8 `std::string gazebo::rendering::UserCamera::GetViewControllerTypeString ( )`

Get current view controller type.

**Returns**

Type of the current view controller: "orbit", "fps"

10.171.3.9 `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

10.171.3.10 `VisualPtr gazebo::rendering::UserCamera::GetVisual ( const math::Vector2i & _mousePos ) const`

Get a visual at a mouse position.

**Parameters**

<code>in</code>	<code>_mousePos</code>	2D position of the mouse in pixels.
-----------------	------------------------	-------------------------------------

10.171.3.11 `void gazebo::rendering::UserCamera::HandleKeyPressEvent ( const std::string & _key )`

Handle a key press.

## Parameters

in	_key	The key pressed.
----	------	------------------

10.171.3.12 void gazebo::rendering::UserCamera::HandleKeyReleaseEvent ( const std::string & \_key )

Handle a key release.

## Parameters

in	_key	The key released.
----	------	-------------------

10.171.3.13 void gazebo::rendering::UserCamera::HandleMouseEvent ( const common::MouseEvent & \_evt )

Handle a mouse event.

## Parameters

in	_evt	The mouse event.
----	------	------------------

10.171.3.14 void gazebo::rendering::UserCamera::Init ( ) [virtual]

Initialize.

Reimplemented from **gazebo::rendering::Camera** (p. 173).

10.171.3.15 void gazebo::rendering::UserCamera::Load ( sdf::ElementPtr \_sdf ) [virtual]

Load the user camera.

## Parameters

in	_sdf	Parameters for the camera.
----	------	----------------------------

Reimplemented from **gazebo::rendering::Camera** (p. 174).

10.171.3.16 void gazebo::rendering::UserCamera::Load ( ) [virtual]

Generic load function.

Reimplemented from **gazebo::rendering::Camera** (p. 174).

10.171.3.17 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. 175)

## Parameters

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

Reimplemented from `gazebo::rendering::Camera` (p. 174).

10.171.3.18 `void gazebo::rendering::UserCamera::MoveToVisual ( VisualPtr _visual )`

Move the camera to focus on a visual.

## Parameters

in	<code>_visual</code>	<b>Visual</b> (p. 850) to move the camera to.
----	----------------------	---

10.171.3.19 `void gazebo::rendering::UserCamera::MoveToVisual ( const std::string & _visualName )`

Move the camera to focus on a visual.

## Parameters

in	<code>_visualName</code>	Name of the visual to move the camera to.
----	--------------------------	---

10.171.3.20 `virtual void gazebo::rendering::UserCamera::PostRender ( ) [virtual]`

Post render.

Reimplemented from `gazebo::rendering::Camera` (p. 175).

10.171.3.21 `void gazebo::rendering::UserCamera::Resize ( unsigned int _w, unsigned int _h )`

Resize the camera.

## Parameters

in	<code>_w</code>	Width of the camera image.
in	<code>_h</code>	Height of the camera image.

10.171.3.22 `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
----	------------------	-----------------

10.171.3.23 `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. 178).

#### 10.171.3.24 void gazebo::rendering::UserCamera::SetViewController ( const std::string & *\_type* )

Set view controller.

#### Parameters

in	<code>_type</code>	The type of view controller: "orbit", "fps"
----	--------------------	---

#### 10.171.3.25 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.

#### 10.171.3.26 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.

#### 10.171.3.27 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. 178).

#### 10.171.3.28 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	<i>_visual</i>	<b>Visual</b> (p. 850) to track.
----	----------------	----------------------------------

#### Returns

True if the camera is now tracking the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 180).

10.171.3.29 `virtual void gazebo::rendering::UserCamera::Update ( ) [virtual]`

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 180).

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

- **UserCamera.hh**

## 10.172 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.*

### 10.172.1 Detailed Description

Generic double x, y vector.

### 10.172.2 Constructor & Destructor Documentation

#### 10.172.2.1 gazebo::math::Vector2d::Vector2d ( )

Constructor.

#### 10.172.2.2 gazebo::math::Vector2d::Vector2d ( const double & \_x, const double & \_y )

Constructor.

##### Parameters

in	_x	value along x
in	_y	value along y

#### 10.172.2.3 gazebo::math::Vector2d::Vector2d ( const Vector2d & \_v )

Copy constructor.

##### Parameters

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

#### 10.172.2.4 virtual gazebo::math::Vector2d::~~Vector2d ( ) [virtual]

Destructor.

### 10.172.3 Member Function Documentation

#### 10.172.3.1 Vector2d gazebo::math::Vector2d::Cross ( const Vector2d & \_v ) const

Return the cross product of this vector and \_v.

##### Parameters

in	_v	the vector
----	----	------------

##### Returns

the cross product

#### 10.172.3.2 double gazebo::math::Vector2d::Distance ( const Vector2d & \_pt ) const

Calc distance to the given point.

## Parameters

in	_pt	The point to measure to
----	-----	-------------------------

## Returns

the distance

## 10.172.3.3 bool gazebo::math::Vector2d::IsFinite ( ) const

See if a point is finite (e.g., not nan)

## Returns

true if finite, false otherwise

## 10.172.3.4 void gazebo::math::Vector2d::Normalize ( )

Normalize the vector length.

## 10.172.3.5 bool gazebo::math::Vector2d::operator!=( const Vector2d &amp; \_v ) const

Not equal to operator.

## Returns

true if elements are of different values (tolerance 1e-6)

## 10.172.3.6 const Vector2d gazebo::math::Vector2d::operator\* ( const Vector2d &amp; \_v ) const

Multiplication operators.

## Parameters

in	_v	the vector
----	----	------------

## Returns

the result

## 10.172.3.7 const Vector2d gazebo::math::Vector2d::operator\* ( double \_v ) const

Multiplication operators.

## Parameters

in	_v	the scaling factor
----	----	--------------------

**Returns**

a scaled vector

10.172.3.8 `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

10.172.3.9 `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

10.172.3.10 `Vector2d gazebo::math::Vector2d::operator+( const Vector2d & _v ) const`

Addition operator.

**Parameters**

<code>in</code>	<code>_v</code>	vector to add
-----------------	-----------------	---------------

**Returns**

sum vector

10.172.3.11 `const Vector2d& gazebo::math::Vector2d::operator+=( const Vector2d & _v )`

Addition assignment operator.

## Parameters

in	_v	the vector to add
----	----	-------------------

## 10.172.3.12 Vector2d gazebo::math::Vector2d::operator- ( const Vector2d &amp; \_v ) const

Subtraction operator.

## Parameters

in	_v	the vector to subtract
----	----	------------------------

## Returns

the subtracted vector

## 10.172.3.13 const Vector2d&amp; gazebo::math::Vector2d::operator-= ( const Vector2d &amp; \_v )

Subtraction assignment operator.

## Parameters

in	_v	the vector to subtract
----	----	------------------------

## Returns

this

## 10.172.3.14 const Vector2d gazebo::math::Vector2d::operator/ ( const Vector2d &amp; \_v ) const

Division operator.

## Remarks

this is an element wise division

## Parameters

in	_v	a vector
----	----	----------

## Returns

a result

## 10.172.3.15 const Vector2d gazebo::math::Vector2d::operator/ ( double \_v ) const

Division operator.

**Parameters**

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

**Returns**

a vector

10.172.3.16 `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

10.172.3.17 `const Vector2d& gazebo::math::Vector2d::operator/= ( double _v )`

Division operator.

**Parameters**

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

**Returns**

a vector

10.172.3.18 `Vector2d& gazebo::math::Vector2d::operator= ( const Vector2d & _v )`

Assignment operator.

**Parameters**

<code>in</code>	<code>_v</code>	a value for x and y element
-----------------	-----------------	-----------------------------

**Returns**

this

10.172.3.19 `const Vector2d& gazebo::math::Vector2d::operator= ( double _v )`

Assignment operator.

#### Parameters

<code>in</code>	<code>_v</code>	the value for x and y element
-----------------	-----------------	-------------------------------

#### Returns

this

10.172.3.20 `bool gazebo::math::Vector2d::operator==( const Vector2d & _v ) const`

Equal to operator.

#### Parameters

<code>in</code>	<code>_v</code>	the vector to compare to
-----------------	-----------------	--------------------------

#### Returns

true if the elements of the 2 vectors are equal within a tolerance (1e-6)

10.172.3.21 `double gazebo::math::Vector2d::operator[] ( unsigned int _index ) const`

Array subscript operator.

#### Parameters

<code>in</code>	<code>_index</code>	the index
-----------------	---------------------	-----------

#### Returns

the value, or 0 if `_index` is out of bounds

10.172.3.22 `void gazebo::math::Vector2d::Set ( double _x, double _y )`

Set the contents of the vector.

#### Parameters

<code>in</code>	<code>_x</code>	value along x
<code>in</code>	<code>_y</code>	value along y

## 10.172.4 Friends And Related Function Documentation

10.172.4.1 `std::ostream& operator<< ( std::ostream & _out, const gazebo::math::Vector2d & _pt )` [*friend*]

Stream extraction operator.

#### Parameters

<code>in</code>	<code><i>_out</i></code>	output stream
<code>in</code>	<code><i>_pt</i></code>	<b>Vector2d</b> (p. 804) to output

#### Returns

The stream

10.172.4.2 `std::istream& operator>> ( std::istream & _in, gazebo::math::Vector2d & _pt )` [*friend*]

Stream extraction operator.

#### Parameters

<code>in</code>	<code><i>_in</i></code>	input stream
<code>in</code>	<code><i>_pt</i></code>	<b>Vector3</b> (p. 821) to read values into

#### Returns

The stream

### 10.172.5 Member Data Documentation

10.172.5.1 `double gazebo::math::Vector2d::x`

x data

10.172.5.2 `double gazebo::math::Vector2d::y`

y data

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

- **Vector2d.hh**

## 10.173 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.*

### 10.173.1 Detailed Description

Generic integer x, y vector.

### 10.173.2 Constructor & Destructor Documentation

#### 10.173.2.1 gazebo::math::Vector2i::Vector2i ( )

Constructor.

#### 10.173.2.2 gazebo::math::Vector2i::Vector2i ( const int & *\_x*, const int & *\_y* )

Constructor.

#### Parameters

in	<i>_x</i>	value along x
in	<i>_y</i>	value along y

#### 10.173.2.3 gazebo::math::Vector2i::Vector2i ( const Vector2i & *\_pt* )

Copy onstructor.

#### Parameters

in	<i>_pt</i>	a point
----	------------	---------

10.173.2.4 virtual gazebo::math::Vector2i::~~Vector2i ( ) [virtual]

Destructor.

### 10.173.3 Member Function Documentation

10.173.3.1 Vector2i gazebo::math::Vector2i::Cross ( const Vector2i & *\_pt* ) const

Return the cross product of this vector and *\_pt*.

#### Parameters

<i>in</i>	<i>_pt</i>	the other vector
-----------	------------	------------------

#### Returns

the product

10.173.3.2 int gazebo::math::Vector2i::Distance ( const Vector2i & *\_pt* ) const

Calc distance to the given point.

#### Parameters

<i>in</i>	<i>_pt</i>	a point
-----------	------------	---------

#### Returns

the distance

10.173.3.3 bool gazebo::math::Vector2i::IsFinite ( ) const

See if a point is finite (e.g., not nan)

#### Returns

the result

10.173.3.4 void gazebo::math::Vector2i::Normalize ( )

Normalize the vector length.

10.173.3.5 bool gazebo::math::Vector2i::operator!= ( const Vector2i & *\_v* ) const

Equality operators.

#### Parameters

<i>_v</i>	the vector to compare with
-----------	----------------------------

**Returns**

true if component have different values, false otherwise

10.173.3.6 `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

10.173.3.7 `const Vector2i gazebo::math::Vector2i::operator*( int _v ) const`

Multiplication operator.

**Parameters**

<code>in</code>	<code>_v</code>	the scaling factor
-----------------	-----------------	--------------------

**Returns**

the result

10.173.3.8 `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

### 10.173.3.9 const Vector2i& gazebo::math::Vector2i::operator\*=( int \_v )

Multiplication operator.

#### Parameters

in	_v	scaling factor
----	----	----------------

#### Returns

this

### 10.173.3.10 Vector2i gazebo::math::Vector2i::operator+ ( const Vector2i & \_v ) const

Addition operator.

#### Parameters

in	_v	the vector to add
----	----	-------------------

#### Returns

the sum vector

### 10.173.3.11 const Vector2i& gazebo::math::Vector2i::operator+=( const Vector2i & \_v )

Addition assignment operator.

#### Parameters

in	_v	the vector to add
----	----	-------------------

#### Returns

this

### 10.173.3.12 Vector2i gazebo::math::Vector2i::operator- ( const Vector2i & \_v ) const

Subtraction operator.

#### Parameters

in	_v	the vector to subtract
----	----	------------------------

#### Returns

the result vector

10.173.3.13 `const Vector2i& gazebo::math::Vector2i::operator-= ( const Vector2i & _v )`

Subtraction operators.

#### Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

#### Returns

this

10.173.3.14 `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

10.173.3.15 `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

10.173.3.16 `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

10.173.3.17 `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

10.173.3.18 `Vector2i& gazebo::math::Vector2i::operator= ( const Vector2i & _v )`

Assignment operator.

**Parameters**

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

**Returns**

this

10.173.3.19 `const Vector2i& gazebo::math::Vector2i::operator= ( int _value )`

Assignment operator.

**Parameters**

<code>in</code>	<code>_value</code>	the value for x and y
-----------------	---------------------	-----------------------

**Returns**

this

10.173.3.20 `bool gazebo::math::Vector2i::operator==( const Vector2i & _v ) const`

Equality operator.

**Parameters**

<code>_v</code>	the vector to compare with
-----------------	----------------------------

**Returns**

true if component have the same values, false otherwise

10.173.3.21 `int gazebo::math::Vector2i::operator[]( unsigned int _index ) const`

Array subscript operator.

**Parameters**

<code>in</code>	<code>_index</code>	the array index
-----------------	---------------------	-----------------

10.173.3.22 `void gazebo::math::Vector2i::Set ( int _x, int _y )`

Set the contents of the vector.

**Parameters**

<code>in</code>	<code>_x</code>	value along x
<code>in</code>	<code>_y</code>	value along y

**10.173.4 Friends And Related Function Documentation**

10.173.4.1 `std::ostream& operator<< ( std::ostream & _out, const gazebo::math::Vector2i & _pt )` [friend]

Stream insertion operator.

**Parameters**

<code>in</code>	<code>_out</code>	output stream
<code>in</code>	<code>pt</code>	<b>Vector2i</b> (p. 812) to output

**Returns**

the stream



10.173.4.2 `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>	<b>Vector3</b> (p. 821) to read values into

#### Returns

The stream

### 10.173.5 Member Data Documentation

10.173.5.1 `int gazebo::math::Vector2i::x`

x data

10.173.5.2 `int gazebo::math::Vector2i::y`

y data

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

- **Vector2i.hh**

## 10.174 gazebo::math::Vector3 Class Reference

The **Vector3** (p. 821) 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  $\sim$ **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.*

### 10.174.1 Detailed Description

The **Vector3** (p. 821) 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.

### 10.174.2 Constructor & Destructor Documentation

#### 10.174.2.1 gazebo::math::Vector3::Vector3 ( )

Constructor.

Referenced by operator-().

#### 10.174.2.2 gazebo::math::Vector3::Vector3 ( const double & *\_x*, const double & *\_y*, const double & *\_z* )

Constructor.

##### Parameters

in	<i>_x</i>	value along x
in	<i>_y</i>	value along y
in	<i>_z</i>	value along z

#### 10.174.2.3 gazebo::math::Vector3::Vector3 ( const Vector3 & *\_v* )

Copy constructor.

##### Parameters

in	<i>_v</i>	a vector
----	-----------	----------

#### 10.174.2.4 virtual gazebo::math::Vector3::~~Vector3 ( ) [virtual]

Destructor.

### 10.174.3 Member Function Documentation

10.174.3.1 void gazebo::math::Vector3::Correct ( ) [inline]

Corrects any nan values.

References x, y, and z.

Referenced by gazebo::math::Pose::Correct().

10.174.3.2 Vector3 gazebo::math::Vector3::Cross ( const Vector3 & \_pt ) const

Return the cross product of this vector and pt.

Returns

the product

10.174.3.3 double gazebo::math::Vector3::Distance ( const Vector3 & \_pt ) const

Calc distance to the given point.

Parameters

in	_pt	the point
----	-----	-----------

Returns

the distance

10.174.3.4 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

10.174.3.5 double gazebo::math::Vector3::Dot ( const Vector3 & \_pt ) const

Return the dot product of this vector and pt.

Returns

the product

10.174.3.6 `bool gazebo::math::Vector3::Equal ( const Vector3 & _v ) const`

Equality test.

#### Remarks

This is equivalent to the `==` operator

#### Parameters

<code>in</code>	<code>_v</code>	the other vector
-----------------	-----------------	------------------

#### Returns

true if the 2 vectors have the same values, false otherwise

10.174.3.7 `Vector3 gazebo::math::Vector3::GetAbs ( ) const`

Get the absolute value of the vector.

#### Returns

a vector with positive elements

10.174.3.8 `double gazebo::math::Vector3::GetDistToLine ( const Vector3 & _pt1, const Vector3 & _pt2 )`

Get distance to a line.

#### Parameters

<code>in</code>	<code>_pt1</code>	first point on the line
<code>in</code>	<code>_pt2</code>	second point on the line

#### Returns

the minimum distance from this point to the line

10.174.3.9 `double gazebo::math::Vector3::GetLength ( ) const`

Returns the length (magnitude) of the vector \ return the length.

10.174.3.10 `double gazebo::math::Vector3::GetMax ( ) const`

Get the maximum value in the vector.

#### Returns

the maximum element

10.174.3.11 `double gazebo::math::Vector3::GetMin ( ) const`

Get the minimum value in the vector.

**Returns**

the minimum element

10.174.3.12 `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

10.174.3.13 `Vector3 gazebo::math::Vector3::GetPerpendicular ( ) const`

Return a vector that is perpendicular to this one.

**Returns**

an orthogonal vector

10.174.3.14 `Vector3 gazebo::math::Vector3::GetRounded ( ) const`

Get a rounded version of this vector.

**Returns**

a rounded vector

10.174.3.15 `double gazebo::math::Vector3::GetSquaredLength ( ) const`

Return the square of the length (magnitude) of the vector.

**Returns**

the squared length

10.174.3.16 `double gazebo::math::Vector3::GetSum ( ) const`

Return the sum of the values.

**Returns**

the sum

10.174.3.17 `bool gazebo::math::Vector3::IsFinite ( ) const`

See if a point is finite (e.g., not nan)

10.174.3.18 `Vector3 gazebo::math::Vector3::Normalize ( )`

Normalize the vector length.

**Returns**

unit length vector

10.174.3.19 `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

10.174.3.20 `Vector3 gazebo::math::Vector3::operator*( const Vector3 & _p ) const`

Multiplication operator.

**Remarks**

this is an element wise multiplication, not a cross product

**Parameters**

<code>in</code>	<code>_v</code>	
-----------------	-----------------	--

10.174.3.21 `Vector3 gazebo::math::Vector3::operator*( double _v ) const`

Multiplication operators.



## Parameters

in	_v	the scaling factor
----	----	--------------------

## Returns

a scaled vector

10.174.3.22 `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

10.174.3.23 `const Vector3& gazebo::math::Vector3::operator*=( double _v )`

Multiplication operator.

## Parameters

in	_v	scaling factor
----	----	----------------

## Returns

this

10.174.3.24 `Vector3 gazebo::math::Vector3::operator+( const Vector3 & _v ) const`

Addition operator.

## Parameters

in	_v	vector to add
----	----	---------------

## Returns

the sum vector

10.174.3.25 `const Vector3& gazebo::math::Vector3::operator+= ( const Vector3 & _v )`

Addition assignment operator.

#### Parameters

<code>in</code>	<code>_v</code>	vector to add
-----------------	-----------------	---------------

10.174.3.26 `Vector3 gazebo::math::Vector3::operator- ( ) const [inline]`

Negation operator.

#### Returns

negative of this vector

References Vector3(), x, y, and z.

10.174.3.27 `Vector3 gazebo::math::Vector3::operator- ( const Vector3 & _pt ) const [inline]`

Subtraction operators.

#### Parameters

<code>in</code>	<code>_pt</code>	a vector to subtract
-----------------	------------------	----------------------

#### Returns

a vector

References Vector3(), x, y, and z.

10.174.3.28 `const Vector3& gazebo::math::Vector3::operator-= ( const Vector3 & _pt )`

Subtraction operators.

#### Parameters

<code>in</code>	<code>_pt</code>	subtrahend
-----------------	------------------	------------

10.174.3.29 `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

10.174.3.30 `const Vector3 gazebo::math::Vector3::operator/ ( double _v ) const`

Division operator.

**Remarks**

this is an element wise division

**Returns**

a vector

10.174.3.31 `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

10.174.3.32 `const Vector3& gazebo::math::Vector3::operator/= ( double _v )`

Division operator.

**Remarks**

this is an element wise division

**Returns**

this

10.174.3.33 `Vector3& gazebo::math::Vector3::operator= ( const Vector3 & _v )`

Assignment operator.

**Parameters**

<code>in</code>	<code>_v</code>	a new value
-----------------	-----------------	-------------

**Returns**

this

10.174.3.34 **Vector3&** gazebo::math::Vector3::operator= ( double *\_value* )

Assignment operator.

**Parameters**

in	<i>_value</i>	assigned to all elements
----	---------------	--------------------------

**Returns**

this

10.174.3.35 **bool** gazebo::math::Vector3::operator== ( const Vector3 & *\_pt* ) const

Equal to operator.

**Parameters**

in	<i>_pt</i>	The vector to compare against
----	------------	-------------------------------

**Returns**

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.174.3.36 **double** gazebo::math::Vector3::operator[] ( unsigned int *index* ) const

[] operator

10.174.3.37 **Vector3** gazebo::math::Vector3::Round ( )

Round to near whole number, return the result.

**Returns**

the result

10.174.3.38 **void** gazebo::math::Vector3::Round ( int *\_precision* )

Round all values to *\_precision* decimal places.

**Parameters**

in	<i>_precision</i>	the decimal places
----	-------------------	--------------------

10.174.3.39 `void gazebo::math::Vector3::Set ( double _x = 0, double _y = 0, double _z = 0 ) [inline]`

Set the contents of the vector.

#### Parameters

<code>in</code>	<code>_x</code>	value along x
<code>in</code>	<code>_y</code>	value along y
<code>in</code>	<code>_z</code>	value along z

References x, y, and z.

10.174.3.40 `void gazebo::math::Vector3::SetToMax ( const Vector3 & _v )`

Set this vector's components to the maximum of itself and the passed in vector.

#### Parameters

<code>in</code>	<code>_v</code>	the maximum clamping vector
-----------------	-----------------	-----------------------------

10.174.3.41 `void gazebo::math::Vector3::SetToMin ( const Vector3 & _v )`

Set this vector's components to the minimum of itself and the passed in vector.

#### Parameters

<code>in</code>	<code>_v</code>	the minimum clamping vector
-----------------	-----------------	-----------------------------

## 10.174.4 Friends And Related Function Documentation

10.174.4.1 `Vector3 operator* ( double _s, const Vector3 & _v ) [friend]`

Multiplication operators.

#### Parameters

<code>in</code>	<code>_v</code>	the scaling factor
-----------------	-----------------	--------------------

#### Returns

a scaled vector

10.174.4.2 `std::ostream& operator<< ( std::ostream & _out, const gazebo::math::Vector3 & _pt ) [friend]`

Stream insertion operator.

#### Parameters

<code>_out</code>	output stream
<code>_pt</code>	<b>Vector3</b> (p. 821) to output

**Returns**

the stream

10.174.4.3 `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

**10.174.5 Member Data Documentation**

10.174.5.1 `double gazebo::math::Vector3::x`

X location.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `operator-()`, `gazebo::math::Quaternion::RotateVector()`, and `Set()`.

10.174.5.2 `double gazebo::math::Vector3::y`

Y location.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `operator-()`, `gazebo::math::Quaternion::RotateVector()`, and `Set()`.

10.174.5.3 `double gazebo::math::Vector3::z`

Z location.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `operator-()`, `gazebo::math::Quaternion::RotateVector()`, and `Set()`.

10.174.5.4 `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**

**10.175 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.*

### 10.175.1 Detailed Description

double Generic x, y, z, w vector

### 10.175.2 Constructor & Destructor Documentation

#### 10.175.2.1 gazebo::math::Vector4::Vector4 ( )

Constructor.



10.175.2.2 gazebo::math::Vector4::Vector4 ( const double & *\_x*, const double & *\_y*, const double & *\_z*, const double & *\_w* )

Constructor with component values.

#### Parameters

in	<i>_x</i>	value along x axis
in	<i>_y</i>	value along y axis
in	<i>_z</i>	value along z axis
in	<i>_w</i>	value along w axis

10.175.2.3 gazebo::math::Vector4::Vector4 ( const Vector4 & *\_v* )

Copy constructor.

#### Parameters

in	<i>_v</i>	vector
----	-----------	--------

10.175.2.4 virtual gazebo::math::Vector4::~~Vector4 ( ) [virtual]

Destructor.

### 10.175.3 Member Function Documentation

10.175.3.1 double gazebo::math::Vector4::Distance ( const Vector4 & *\_pt* ) const

Calc distance to the given point.

#### Parameters

in	<i>_pt</i>	the point
----	------------	-----------

#### Returns

the distance

10.175.3.2 double gazebo::math::Vector4::GetLength ( ) const

Returns the length (magnitude) of the vector.

10.175.3.3 double gazebo::math::Vector4::GetSquaredLength ( ) const

Return the square of the length (magnitude) of the vector.

#### Returns

the length

10.175.3.4 `bool gazebo::math::Vector4::IsFinite ( ) const`

See if a point is finite (e.g., not nan)

#### Returns

true if finite, false otherwise

10.175.3.5 `void gazebo::math::Vector4::Normalize ( )`

Normalize the vector length.

10.175.3.6 `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

10.175.3.7 `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

10.175.3.8 `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`

**10.175.3.9** `const Vector4 gazebo::math::Vector4::operator* ( double _v ) const`

Multiplication operators.

**Parameters**

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

**Returns**

a scaled vector

**10.175.3.10** `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

**10.175.3.11** `const Vector4& gazebo::math::Vector4::operator*= ( double _v )`

Multiplication assignment operator.

**Parameters**

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

**Returns**

this

**10.175.3.12** `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

10.175.3.13 `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

10.175.3.14 `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

10.175.3.15 `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

10.175.3.16 `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

10.175.3.17 `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

10.175.3.18 `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

10.175.3.19 `const Vector4& gazebo::math::Vector4::operator/= ( double _v )`

Division operator.

## Parameters

in	<code>_v</code>	scaling factor
----	-----------------	----------------

## Returns

a vector

10.175.3.20 `Vector4& gazebo::math::Vector4::operator=( const Vector4 & _v )`

Assignment operator.

## Parameters

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

## Returns

a reference to this vector

10.175.3.21 `Vector4& gazebo::math::Vector4::operator=( double _value )`

Assignment operator.

## Parameters

in	<code>_value</code>	
----	---------------------	--

10.175.3.22 `bool gazebo::math::Vector4::operator==( const Vector4 & _pt ) const`

Equal to operator.

## Parameters

in	<code>_pt</code>	the other vector
----	------------------	------------------

## Returns

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.175.3.23 `double gazebo::math::Vector4::operator[]( unsigned int _index ) const`

Array subscript operator.

## Parameters

in	<code>_index</code>	
----	---------------------	--

10.175.3.24 `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>_x</code>	value along x axis
<code>in</code>	<code>_y</code>	value along y axis
<code>in</code>	<code>_z</code>	value along z axis
<code>in</code>	<code>_w</code>	value along w axis

## 10.175.4 Friends And Related Function Documentation

10.175.4.1 `std::ostream& operator<< ( std::ostream & _out, const gazebo::math::Vector4 & _pt )` [`friend`]

Stream insertion operator.

#### Parameters

<code>in</code>	<code>_out</code>	output stream
<code>in</code>	<code>_pt</code>	<b>Vector4</b> (p. 834) to output

#### Returns

The stream

10.175.4.2 `std::istream& operator>> ( std::istream & _in, gazebo::math::Vector4 & _pt )` [`friend`]

Stream extraction operator.

#### Parameters

<code>in</code>	<code>_in</code>	input stream
<code>in</code>	<code>_pt</code>	<b>Vector4</b> (p. 834) to read values into

#### Returns

the stream

## 10.175.5 Member Data Documentation

10.175.5.1 `double gazebo::math::Vector4::w`

W value.

10.175.5.2 `double gazebo::math::Vector4::x`

X value.

10.175.5.3 double gazebo::math::Vector4::y

Y value.

10.175.5.4 double gazebo::math::Vector4::z

Z value.

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

- **Vector4.hh**

## 10.176 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.*

### 10.176.1 Detailed Description

Handle video encoding and decoding using libavcodec.

### 10.176.2 Constructor & Destructor Documentation

10.176.2.1 gazebo::common::Video::Video ( )

Constructor.

10.176.2.2 virtual gazebo::common::Video::~~Video ( ) [virtual]

Destructor.



### 10.176.3 Member Function Documentation

#### 10.176.3.1 int gazebo::common::Video::GetHeight ( ) const

Get the height of the video in pixels.

##### Returns

the height

#### 10.176.3.2 bool gazebo::common::Video::GetNextFrame ( unsigned char \*\* *\_buffer* )

Get the next frame of the video.

##### Parameters

out	<i>_img</i>	<b>Image</b> (p. 352) in which the frame is stored
-----	-------------	--

##### Returns

false if HAVE\_FFmpeg is not defined, true otherwise

#### 10.176.3.3 int gazebo::common::Video::GetWidth ( ) const

Get the width of the video in pixels.

##### Returns

the width

#### 10.176.3.4 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:

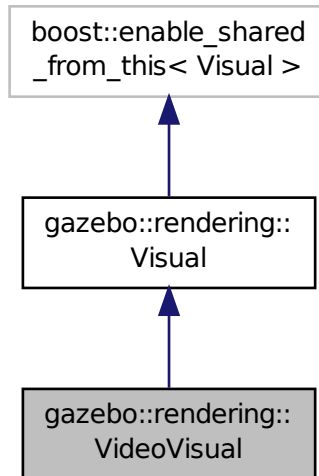
- **Video.hh**

## 10.177 gazebo::rendering::VideoVisual Class Reference

**A** (p. 107) 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

### 10.177.1 Detailed Description

**A** (p. 107) visual element that displays a video as a texture.

### 10.177.2 Constructor & Destructor Documentation

#### 10.177.2.1 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.

10.177.2.2 virtual gazebo::rendering::VideoVisual::~~VideoVisual ( ) [virtual]

Destructor.

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

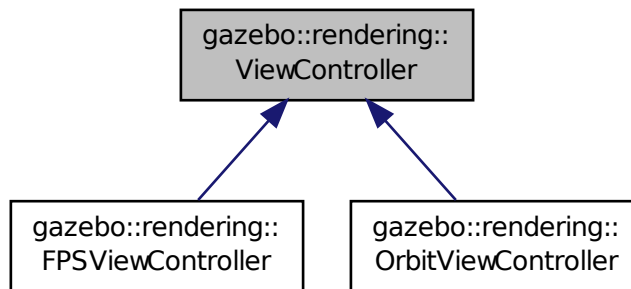
- **VideoVisual.hh**

## 10.178 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. 157).*

## Protected Attributes

- **UserCameraPtr camera**

*Pointer to the camera to control.*

- bool **enabled**

*True if enabled.*

- std::string **typeString**

*Type of view controller.*

## 10.178.1 Detailed Description

Base class for view controllers.

## 10.178.2 Constructor & Destructor Documentation

### 10.178.2.1 gazebo::rendering::ViewController::ViewController ( UserCameraPtr `_camera` )

Constructor.

#### Parameters

in	<code>_camera</code>	The user camera to controll.
----	----------------------	------------------------------

### 10.178.2.2 virtual gazebo::rendering::ViewController::~~ViewController ( ) [virtual]

Destructor.

## 10.178.3 Member Function Documentation

### 10.178.3.1 std::string gazebo::rendering::ViewController::GetTypeString ( ) const

Get the type of view controller.

#### Returns

The view controller type string.

### 10.178.3.2 virtual void gazebo::rendering::ViewController::HandleKeyPressEvent ( const std::string & `_key` ) [pure virtual]

Handle a key press event.

## Parameters

in	<code>_key</code>	The key that was pressed.
----	-------------------	---------------------------

Implemented in **gazebo::rendering::OrbitViewController** (p. 536), and **gazebo::rendering::FPSViewController** (p. 314).

**10.178.3.3** `virtual void gazebo::rendering::ViewController::HandleKeyReleaseEvent ( const std::string & _key ) [pure virtual]`

Handle a key release event.

## Parameters

in	<code>_key</code>	The key that was released.
----	-------------------	----------------------------

Implemented in **gazebo::rendering::OrbitViewController** (p. 537), and **gazebo::rendering::FPSViewController** (p. 314).

**10.178.3.4** `virtual void gazebo::rendering::ViewController::HandleMouseEvent ( const common::MouseEvent & _event ) [pure virtual]`

Handle a mouse event.

## Parameters

in	<code>_event</code>	The mouse position.
----	---------------------	---------------------

Implemented in **gazebo::rendering::OrbitViewController** (p. 537), and **gazebo::rendering::FPSViewController** (p. 314).

**10.178.3.5** `virtual void gazebo::rendering::ViewController::Init ( ) [pure virtual]`

Initialize the view controller.

Implemented in **gazebo::rendering::OrbitViewController** (p. 537), and **gazebo::rendering::FPSViewController** (p. 315).

**10.178.3.6** `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. 537).

**10.178.3.7** `void gazebo::rendering::ViewController::SetEnabled ( bool _value )`

Set whether the controller is enabled.

## Parameters

in	<code>_value</code>	True if the controller is enabled.
----	---------------------	------------------------------------

10.178.3.8 `virtual void gazebo::rendering::ViewController::Update ( )` [pure virtual]

Update the controller, which should update the position of the **Camera** (p. 157).

Implemented in `gazebo::rendering::OrbitViewController` (p. 538), and `gazebo::rendering::FPSViewController` (p. 315).

## 10.178.4 Member Data Documentation

10.178.4.1 `UserCameraPtr gazebo::rendering::ViewController::camera` [protected]

Pointer to the camera to control.

10.178.4.2 `bool gazebo::rendering::ViewController::enabled` [protected]

True if enabled.

10.178.4.3 `std::string gazebo::rendering::ViewController::typeString` [protected]

Type of view controller.

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

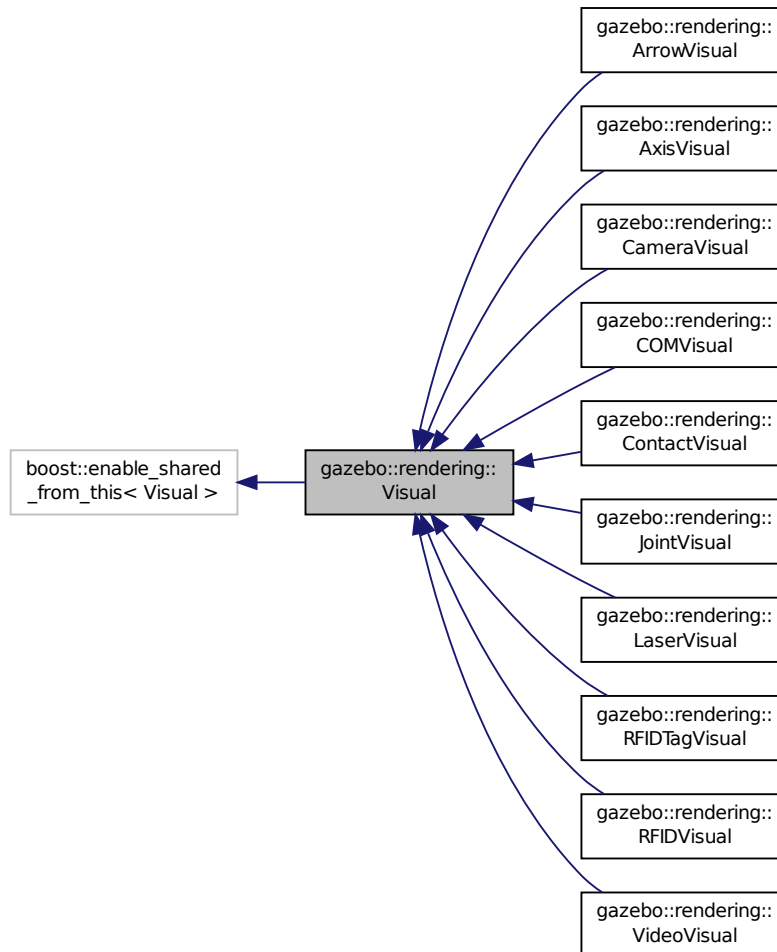
- `ViewController.hh`

## 10.179 gazebo::rendering::Visual Class Reference

**A** (p. 107) 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. 103).*
- **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. 103).*

### 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. 103).*

## 10.179.1 Detailed Description

**A** (p. 107) renderable object.

## 10.179.2 Constructor & Destructor Documentation

10.179.2.1 gazebo::rendering::Visual::Visual ( const std::string & \_name, VisualPtr \_parent, bool \_useRTShader = true )

Constructor.

### Parameters

in	<code>_name</code>	Name of the visual.
in	<code>_parent</code>	Parent of the visual.
in	<code>_useRTShader</code>	True if the visual should use the real-time shader system (RTShader).

10.179.2.2 `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>	<b>Scene</b> (p. 651) containing the visual.
in	<code>_useRTShader</code>	True if the visual should use the real-time shader system (RTShader).

10.179.2.3 `virtual gazebo::rendering::Visual::~~Visual ( ) [virtual]`

Destructor.

### 10.179.3 Member Function Documentation

10.179.3.1 `void gazebo::rendering::Visual::AttachAxes ( )`

Attach visualization axes.

10.179.3.2 `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.

10.179.3.3 `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.

10.179.3.4 `void gazebo::rendering::Visual::AttachObject ( Ogre::MovableObject * _obj )`

Attach a renewable object to the visual.

#### Parameters

in	<code>_obj</code>	<b>A</b> (p. 107) movable object to attach to the visual.
----	-------------------	---

10.179.3.5 void gazebo::rendering::Visual::AttachVisual ( VisualPtr \_vis )

Attach a visual to this visual.

#### Parameters

in	_vis	Visual (p. 850) to attach.
----	------	----------------------------

10.179.3.6 void gazebo::rendering::Visual::ClearParent ( )

Clear parents.

10.179.3.7 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. 850).
in	_newParent	Parent of the cloned Visual (p. 850).

#### Returns

The visual.

10.179.3.8 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. 107) pointer to the new dynamic line.

10.179.3.9 void gazebo::rendering::Visual::DeleteDynamicLine ( DynamicLines \* \_line )

Delete a dynamic line.

#### Parameters

in	_line	Pointer to the line to delete.
----	-------	--------------------------------

10.179.3.10 void gazebo::rendering::Visual::DetachObjects ( )

Detach all objects.

10.179.3.11 `void gazebo::rendering::Visual::DetachVisual ( VisualPtr _vis )`

Detach a visual.

Parameters

<code>in</code>	<code>_vis</code>	<b>Visual</b> (p. 850) to detach.
-----------------	-------------------	-----------------------------------

10.179.3.12 `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.
-----------------	--------------------	-------------------------------

10.179.3.13 `void gazebo::rendering::Visual::DisableTrackVisual ( )`

Disable tracking of a visual.

10.179.3.14 `void gazebo::rendering::Visual::EnableTrackVisual ( VisualPtr _vis )`

Set one visual to track/follow another.

Parameters

<code>in</code>	<code>_vis</code>	<b>Visual</b> (p. 850) to track.
-----------------	-------------------	----------------------------------

10.179.3.15 `void gazebo::rendering::Visual::Fini ( )`

Helper for the destructor.

10.179.3.16 `unsigned int gazebo::rendering::Visual::GetAttachedObjectCount ( ) const`

Return the number of attached movable objects.

Returns

The number of attached movable objects.

10.179.3.17 `math::Box gazebo::rendering::Visual::GetBoundingBox ( ) const`

Get the bounding box for the visual.

Returns

The bounding box in world coordinates.

**10.179.3.18** `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. 859).

**Parameters**

<code>in</code>	<code>_index</code>	Index of the child to retrieve.
-----------------	---------------------	---------------------------------

**Returns**

Pointer to the child visual, NULL if index is invalid.

**10.179.3.19** `unsigned int gazebo::rendering::Visual::GetChildCount ( )`

Get the number of attached visuals.

**Returns**

The number of children.

**10.179.3.20** `std::string gazebo::rendering::Visual::GetMaterialName ( ) const`

Get the name of the material.

**Returns**

The name of the visual applied to this visual.

**10.179.3.21** `std::string gazebo::rendering::Visual::GetMeshName ( ) const`

The name of the mesh set in the visual's SDF.

**Returns**

Name of the mesh.

**10.179.3.22** `std::string gazebo::rendering::Visual::GetName ( ) const`

Get the name of the visual.

**Returns**

The name of the visual.

**10.179.3.23** `std::string gazebo::rendering::Visual::GetNormalMap ( ) const`

Get the normal map.

**Returns**

The name of the normal map material.

10.179.3.24 **VisualPtr** gazebo::rendering::Visual::GetParent ( ) const

Get the parent visual, if one exists.

**Returns**

Pointer to the parent visual, NULL if no parent.

10.179.3.25 **math::Pose** gazebo::rendering::Visual::GetPose ( ) const

Get the pose of the visual.

**Returns**

The **Visual** (p. 850)'s pose.

10.179.3.26 **math::Vector3** gazebo::rendering::Visual::GetPosition ( ) const

Get the position of the visual.

**Returns**

The visual's position.

10.179.3.27 **VisualPtr** gazebo::rendering::Visual::GetRootVisual ( )

Get the root visual.

**Returns**

The root visual, which is one level below the world visual.

10.179.3.28 **math::Quaternion** gazebo::rendering::Visual::GetRotation ( ) const

Get the rotation of the visual.

**Returns**

The visual's rotation.

10.179.3.29 **math::Vector3** gazebo::rendering::Visual::GetScale ( )

Get the scale.

**Returns**

The scaling factor.



10.179.3.30 `ScenePtr gazebo::rendering::Visual::GetScene ( ) const`

Get current.

Returns

Pointer to the scene.

10.179.3.31 `Ogre::SceneNode* gazebo::rendering::Visual::GetSceneNode ( ) const`

Return the scene Node of this visual entity.

Returns

The **Ogre** (p. 103) scene node.

10.179.3.32 `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".

10.179.3.33 `float gazebo::rendering::Visual::GetTransparency ( )`

Get the transparency.

Returns

The transparency.

10.179.3.34 `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. 1033)

**GZ\_VISIBILITY\_GUI** (p. 1033)

**GZ\_VISIBILITY\_NOT\_SELECTABLE** (p. 1033)

10.179.3.35 `bool gazebo::rendering::Visual::GetVisible ( ) const`

Get whether the visual is visible.

#### Returns

True if the visual is visible.

10.179.3.36 `math::Pose gazebo::rendering::Visual::GetWorldPose ( ) const`

Get the global pose of the node.

#### Returns

The pose in the world coordinate frame.

10.179.3.37 `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.
-----------------	--------------------	----------------------------

10.179.3.38 `void gazebo::rendering::Visual::Init ( )`

Helper for the constructor.

10.179.3.39 `void gazebo::rendering::Visual::InsertMesh ( const std::string & _meshName )`

Insert a mesh into **Ogre** (p. 103).

#### Parameters

<code>in</code>	<code>_meshName</code>	Name of the mesh to insert.
-----------------	------------------------	-----------------------------

10.179.3.40 `static void gazebo::rendering::Visual::InsertMesh ( const common::Mesh * _mesh ) [static]`

Insert a mesh into **Ogre** (p. 103).

#### Parameters

<code>in</code>	<code>_mesh</code>	Pointer to the mesh to insert.
-----------------	--------------------	--------------------------------

10.179.3.41 `bool gazebo::rendering::Visual::IsPlane ( ) const`

Return true if the visual is a plane.

**Returns**

True if a plane.

10.179.3.42 `bool gazebo::rendering::Visual::IsStatic ( ) const`

Return true if the visual is a static geometry.

**Returns**

True if the visual is static.

10.179.3.43 `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.
-----------------	-------------------	---------------------------

10.179.3.44 `virtual void gazebo::rendering::Visual::Load ( ) [virtual]`

Load the visual with default parameters.

Reimplemented in **`gazebo::rendering::ArrowVisual`** (p. 127), and **`gazebo::rendering::AxisVisual`** (p. 130).

10.179.3.45 `void gazebo::rendering::Visual::LoadFromMsg ( ConstVisualPtr & _msg )`

Load from a message.

**Parameters**

<code>in</code>	<code>_msg</code>	<b>A</b> (p. 107) visual message.
-----------------	-------------------	-----------------------------------

10.179.3.46 `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>	<b>A</b> (p. 107) unique name for the plugin
<code>_sdf</code>	The SDF to pass into the plugin.

10.179.3.47 `void gazebo::rendering::Visual::MakeStatic ( )`

Make the visual objects static renderables.

10.179.3.48 `void gazebo::rendering::Visual::MoveToPosition ( const math::Pose & _pose, double _time )`

Move to a pose and over a given time.

#### Parameters

in	<code>_pose</code>	Pose the visual will end at.
in	<code>_time</code>	Time it takes the visual to move to the pose.

10.179.3.49 `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>_poses</code>	Series of poses the visual will move to.
in	<code>_time</code>	Time it takes the visual to move to the pose.
in	<code>_onComplete</code>	Callback used when the move is complete.

10.179.3.50 `void gazebo::rendering::Visual::RemovePlugin ( const std::string & _name )`

Remove a running plugin.

#### Parameters

<code>_name</code>	The unique name of the plugin to remove
--------------------	---

10.179.3.51 `void gazebo::rendering::Visual::SetAmbient ( const common::Color & _color )`

Set the ambient color of the visual.

#### Parameters

in	<code>_color</code>	The ambient color.
----	---------------------	--------------------

10.179.3.52 `void gazebo::rendering::Visual::SetCastShadows ( bool _shadows )`

Set whether the visual should cast shadows.

#### Parameters

in	<code>_shadows</code>	True to enable shadows.
----	-----------------------	-------------------------

10.179.3.53 `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.
----	---------------	------------------------

10.179.3.54 `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. 397).

10.179.3.55 `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.
----	---------------------	----------------------------------

10.179.3.56 `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.

10.179.3.57 `void gazebo::rendering::Visual::SetName ( const std::string & _name )`

Set the name of the visual.

## Parameters

in	<i>_name</i>	Name of the visual
----	--------------	--------------------

10.179.3.58 `void gazebo::rendering::Visual::SetNormalMap ( const std::string & _nmap )`

Set the normal map.

## Parameters

in	<i>_nmap</i>	Name of the normal map material.
----	--------------	----------------------------------

10.179.3.59 `void gazebo::rendering::Visual::SetPose ( const math::Pose & _pose )`

Set the pose of the visual.

Parameters

<code>in</code>	<code><i>_pose</i></code>	The new pose of the visual.
-----------------	---------------------------	-----------------------------

10.179.3.60 `void gazebo::rendering::Visual::SetPosition ( const math::Vector3 & _pos )`

Set the position of the visual.

Parameters

<code>in</code>	<code><i>_pos</i></code>	The position to set the visual to.
-----------------	--------------------------	------------------------------------

10.179.3.61 `void gazebo::rendering::Visual::SetRibbonTrail ( bool _value, const common::Color & _initialColor, const common::Color & _changeColor )`

True on or off a ribbon trail.

Parameters

<code>in</code>	<code><i>_value</i></code>	True to enable ribbon trail.
<code>in</code>	<code><i>_initialColor</i></code>	The initial color of the ribbon trail.
<code>in</code>	<code><i>_changeColor</i></code>	Color to change too as the trail grows.

10.179.3.62 `void gazebo::rendering::Visual::SetRotation ( const math::Quaternion & _rot )`

Set the rotation of the visual.

Parameters

<code>in</code>	<code><i>_rot</i></code>	The rotation of the visual.
-----------------	--------------------------	-----------------------------

10.179.3.63 `void gazebo::rendering::Visual::SetScale ( const math::Vector3 & _scale )`

Set the scale.

Parameters

<code>in</code>	<code><i>_scale</i></code>	The scaling factor for the visual.
-----------------	----------------------------	------------------------------------

10.179.3.64 `void gazebo::rendering::Visual::SetScene ( ScenePtr _scene )`

Set current scene.

## Parameters

in	_scene	Pointer to the scene.
----	--------	-----------------------

10.179.3.65 void gazebo::rendering::Visual::SetShaderType ( const std::string & *.type* )

Set the shader type for the visual's material.

## Parameters

in	_type	Shader type string: "vertex", "pixel", "normal_map_object_space", "normal_map_tangent_space".
----	-------	---

10.179.3.66 void gazebo::rendering::Visual::SetSkeletonPose ( const msgs::PoseAnimation & *.pose* )

Set animation skeleton pose.

## Parameters

in	_pose	Skelton message
----	-------	-----------------

10.179.3.67 void gazebo::rendering::Visual::SetSpecular ( const common::Color & *\_color* )

Set the specular color of the visual.

## Parameters

in	_color	Specular color.
----	--------	-----------------

10.179.3.68 void gazebo::rendering::Visual::SetTransparency ( float *.trans* )

Set the transparency.

## Parameters

in	_trans	The transparency, between 0 and 1 where 0 is no transparency.
----	--------	---

10.179.3.69 void gazebo::rendering::Visual::SetVisibilityFlags ( uint32\_t *.flags* )

Set visibility flags for this visual and all children.

## Parameters

in	_flags	The visibility flags.
----	--------	-----------------------

## See Also

**GZ\_VISIBILITY\_ALL** (p. 1033)

**GZ\_VISIBILITY\_GUI** (p. 1033)

**GZ\_VISIBILITY\_NOT\_SELECTABLE** (p. 1033)

10.179.3.70 `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.

10.179.3.71 `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.
-----------------	---------------------------	---

10.179.3.72 `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.
-----------------	---------------------------	---

10.179.3.73 `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.
-----------------	--------------------------	---

10.179.3.74 `void gazebo::rendering::Visual::ShowBoundingBox ( )`

Display the bounding box visual.

10.179.3.75 `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.
-----------------	---------------------------	--



10.179.3.76 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.
-----------	--------------	---

10.179.3.77 void gazebo::rendering::Visual::ShowJoints ( bool *\_show* )

Display joint visuals.

#### Parameters

<i>in</i>	<i>_show</i>	True to show joint visualizations.
-----------	--------------	------------------------------------

10.179.3.78 void gazebo::rendering::Visual::ShowSkeleton ( bool *\_show* )

Display the skeleton visuals.

#### Parameters

<i>in</i>	<i>_show</i>	True to show skeleton visuals.
-----------	--------------	--------------------------------

10.179.3.79 void gazebo::rendering::Visual::ToggleVisible ( )

Toggle whether this visual is visible.

10.179.3.80 void gazebo::rendering::Visual::Update ( )

Update the visual.

10.179.3.81 void gazebo::rendering::Visual::UpdateFromMsg ( ConstVisualPtr & *\_msg* )

Update a visual based on a message.

#### Parameters

<i>in</i>	<i>_msg</i>	The visual message.
-----------	-------------	---------------------

## 10.179.4 Member Data Documentation

10.179.4.1 VisualPtr gazebo::rendering::Visual::parent [protected]

Parent visual.

#### 10.179.4.2 ScenePtr gazebo::rendering::Visual::scene [protected]

Pointer to the visual's scene.

#### 10.179.4.3 Ogre::SceneNode\* gazebo::rendering::Visual::sceneNode [protected]

Pointer to the visual's scene node in **Ogre** (p. 103).

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

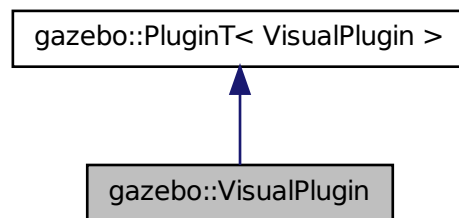
- **Visual.hh**

## 10.180 gazebo::VisualPlugin Class Reference

**A** (p. 107) 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

#### 10.180.1 Detailed Description

**A** (p. 107) plugin loaded within the gzserver on startup.

See [reference](#).

## 10.180.2 Constructor & Destructor Documentation

10.180.2.1 `gazebo::VisualPlugin::VisualPlugin ( )` `[inline]`

References `gazebo::PluginT< VisualPlugin >::type`, and `gazebo::VISUAL_PLUGIN`.

## 10.180.3 Member Function Documentation

10.180.3.1 `virtual void gazebo::VisualPlugin::Init ( )` `[inline]`, `[virtual]`

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.180.3.2 `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.

10.180.3.3 `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:

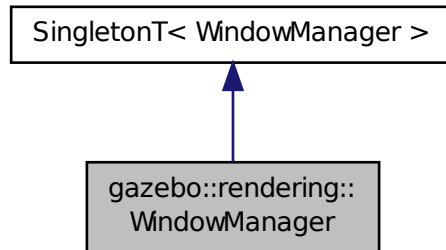
- `common/Plugin.hh`

## 10.181 gazebo::rendering::WindowManager Class Reference

Class to mangage 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. 103) 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

### 10.181.1 Detailed Description

Class to manage render windows.

### 10.181.2 Member Function Documentation

10.181.2.1 int gazebo::rendering::WindowManager::CreateWindow ( const std::string & \_ogreHandle, uint32\_t \_width, uint32\_t \_height )

Create a window.

## Parameters

in	<code>_ogreHandle</code>	String representing the ogre window handle.
in	<code>_width</code>	Width of the window in pixels.
in	<code>_height</code>	Height of the window in pixels.

10.181.2.2 void gazebo::rendering::WindowManager::Fini ( )

Shutdown all the windows.

10.181.2.3 float gazebo::rendering::WindowManager::GetAvgFPS ( uint32\_t *\_id* )

Get the average FPS.

## Parameters

in	<code>_id</code>	ID of the window.
----	------------------	-------------------

## Returns

The frames per second.

10.181.2.4 uint32\_t gazebo::rendering::WindowManager::GetTriangleCount ( uint32\_t *\_id* )

Get the triangle count.

## Parameters

in	<code>_id</code>	ID of the window.
----	------------------	-------------------

## Returns

The triangle count.

10.181.2.5 Ogre::RenderWindow\* gazebo::rendering::WindowManager::GetWindow ( uint32\_t *\_id* )

Get the render window associated with the given id.

## Parameters

in	<code>_id</code>	ID of the window.
----	------------------	-------------------

## Returns

Pointer to the render window, NULL if the id is invalid.

10.181.2.6 void gazebo::rendering::WindowManager::Moved ( uint32\_t *\_id* )

Tells **Ogre** (p. 103) the window has moved, and needs updating.

## Parameters

in	<code>_id</code>	ID of the window.
----	------------------	-------------------

10.181.2.7 `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.

10.181.2.8 `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**

## 10.182 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.*

### 10.182.1 Detailed Description

Draws a wireframe box.

## 10.182.2 Constructor & Destructor Documentation

10.182.2.1 gazebo::rendering::WireBox::WireBox ( VisualPtr *\_parent*, const math::Box & *\_box* ) [explicit]

Constructor.

### Parameters

in	<i>_box</i>	Dimension of the box to draw.
----	-------------	-------------------------------

10.182.2.2 gazebo::rendering::WireBox::~~WireBox ( )

Destructor.

## 10.182.3 Member Function Documentation

10.182.3.1 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.
----	-------------	--------------------------------

10.182.3.2 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:

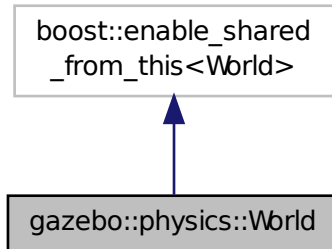
- **WireBox.hh**

## 10.183 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 **EnqueueMsg** (msgs::Pose \*\_msg)  
*Enqueue a pose message for publication.*
- 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. 274) 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. 274).*
- 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. 765).*
- **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. 274) tree.*
- 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. 283) on the **physics::Link** (p. 404) in **World::Update**.*

### 10.183.1 Detailed Description

The world provides access to all other object within a simulated environment.

The **World** (p. 875) is the container for all models and their components (links, joints, sensors, plugins, etc), and **World-Plugin** (p. 887) instances. Many core function are also handled in the **World** (p. 875), including physics update, model updates, and message processing.

### 10.183.2 Constructor & Destructor Documentation

10.183.2.1 `gazebo::physics::World::World ( const std::string & _name = " " ) [explicit]`

Constructor.

Constructor for the **World** (p. 875). Must specify a unique name.

#### Parameters

in	_name	Name of the world.
----	-------	--------------------

10.183.2.2 gazebo::physics::World::~~World ( )

Destructor.

### 10.183.3 Member Function Documentation

10.183.3.1 void gazebo::physics::World::Clear ( )

Remove all entities from the world.

10.183.3.2 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.

10.183.3.3 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.

10.183.3.4 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.
----	----------------	------------------------------------

10.183.3.5 void gazebo::physics::World::EnqueueMsg ( msgs::Pose \* *\_msg* )

Enqueue a pose message for publication.

These messages will be transmitted at the end of every iteration.

#### Parameters

in	<i>_msg</i>	The message to enqueue.
----	-------------	-------------------------

10.183.3.6 void gazebo::physics::World::Fini ( )

Finalize the world.

Call this function to tear-down the world.

10.183.3.7 BasePtr gazebo::physics::World::GetByName ( const std::string & *\_name* )

Get an element by name.

Searches the list of entities, and return a pointer to the model with a matching `_name`.

#### Parameters

<code>in</code>	<code>_name</code>	The name of the <b>Model</b> (p. 469) to find.
-----------------	--------------------	--

#### Returns

**A** (p. 107) pointer to the entity, or NULL if no entity was found.

#### 10.183.3.8 `bool gazebo::physics::World::GetEnablePhysicsEngine ( ) [inline]`

check if physics engine is enabled/disabled.

#### Parameters

<code>True</code>	if the physics engine is enabled.
-------------------	-----------------------------------

#### 10.183.3.9 `EntityPtr gazebo::physics::World::GetEntity ( const std::string & _name )`

Get a pointer to an **Entity** (p. 274) based on a name.

This function is the same as `GetByName`, but limits the search to only Entities.

#### Parameters

<code>in</code>	<code>_name</code>	The name of the <b>Entity</b> (p. 274) to find.
-----------------	--------------------	---

#### Returns

**A** (p. 107) pointer to the **Entity** (p. 274), or NULL if no **Entity** (p. 274) was found.

#### 10.183.3.10 `EntityPtr gazebo::physics::World::GetEntityBelowPoint ( const math::Vector3 & _pt )`

Get the nearest entity below a point.

Projects a Ray down (-Z axis) starting at the given point. The first entity hit by the Ray is returned.

#### Parameters

<code>in</code>	<code>_pt</code>	The 3D point to search below
-----------------	------------------	------------------------------

#### Returns

**A** (p. 107) pointer to nearest **Entity** (p. 274), NULL if none is found.

#### 10.183.3.11 `ModelPtr gazebo::physics::World::GetModel ( unsigned int _index ) const`

Get a model based on an index.

Get a **Model** (p. 469) using an index, where index must be greater than zero and less than **World::GetModelCount()** (p. 881)

## Parameters

in	_index	The index of the model [0..GetModelCount)
----	--------	---

## Returns

**A** (p. 107) pointer to the **Model** (p. 469). NULL if \_index is invalid.

10.183.3.12 **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

in	_name	The name of the <b>Model</b> (p. 469) to find.
----	-------	--

## Returns

**A** (p. 107) pointer to the **Model** (p. 469), or NULL if no model was found.

10.183.3.13 **ModelPtr** gazebo::physics::World::GetModelBelowPoint ( const math::Vector3 & \_pt )

Get the nearest model below a point.

This function makes use of **World::GetEntityBelowPoint** (p. 880).

## Parameters

in	_pt	The 3D point to search below.
----	-----	-------------------------------

## Returns

**A** (p. 107) pointer to nearest **Model** (p. 469), NULL if none is found.

## 10.183.3.14 unsigned int gazebo::physics::World::GetModelCount ( ) const

Get the number of models.

## Returns

The number of models in the **World** (p. 875).

10.183.3.15 **Model\_V** gazebo::physics::World::GetModels ( ) const

Get a list of all the models.

**Returns**

**A** (p. 107) list of all the Models in the world.

10.183.3.16 `std::string gazebo::physics::World::GetName ( ) const`

Get the name of the world.

**Returns**

The name of the world.

10.183.3.17 `common::Time gazebo::physics::World::GetPauseTime ( ) const`

Get the amount of time simulation has been paused.

**Returns**

The pause time.

10.183.3.18 `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.

10.183.3.19 `common::Time gazebo::physics::World::GetRealTime ( ) const`

Get the real time (elapsed time).

**Returns**

The real time.

10.183.3.20 `EntityPtr gazebo::physics::World::GetSelectedEntity ( ) const`

Get the selected **Entity** (p. 274).

The selected entity is set via the GUI.

**Returns**

**A** (p. 107) point to the **Entity** (p. 274), NULL if nothing is selected.

10.183.3.21 `boost::mutex*` gazebo::physics::World::GetSetWorldPoseMutex ( ) const [inline]

Get the set world pose mutex.

#### Returns

Pointer to the mutex.

10.183.3.22 `common::Time` gazebo::physics::World::GetSimTime ( ) const

Get the world simulation time, note if you want the PC wall clock call `common::Time::GetWallTime` (p. 765).

#### Returns

The current simulation time

10.183.3.23 `common::Time` gazebo::physics::World::GetStartTime ( ) const

Get the wall time simulation was started.

#### Returns

The start time.

10.183.3.24 `void` gazebo::physics::World::Init ( )

Initialize the world.

This is called after Load.

10.183.3.25 `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).
-----------------	---------------------------	--

10.183.3.26 `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>	<b>A</b> (p. 107) reference to an SDF object.
-----------------	-------------------	---

10.183.3.27 `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

<i>in</i>	<i>_sdfString</i>	<b>A</b> (p. 107) string containing valid SDF markup.
-----------	-------------------	---

10.183.3.28 `bool gazebo::physics::World::IsLoaded ( ) const`

Return true if the world has been loaded.

#### Returns

True if **World::Load** (p. 884) has completed.

10.183.3.29 `bool gazebo::physics::World::IsPaused ( ) const`

Returns the state of the simulation true if paused.

#### Returns

True if paused.

10.183.3.30 `void gazebo::physics::World::Load ( sdf::ElementPtr _sdf )`

Load the world using SDF parameters.

Load a world from and SDF pointer.

#### Parameters

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

10.183.3.31 `void gazebo::physics::World::LoadPlugin ( const std::string & _filename, const std::string & _name, sdf::ElementPtr _sdf )`

Load a plugin.

#### Parameters

<i>in</i>	<i>_filename</i>	The filename of the plugin.
<i>in</i>	<i>_name</i>	<b>A</b> (p. 107) unique name for the plugin.
<i>in</i>	<i>_sdf</i>	The SDF to pass into the plugin.



10.183.3.32 `void gazebo::physics::World::PrintEntityTree ( )`

Print **Entity** (p. 274) tree.

Prints all the entities to stdout.

10.183.3.33 `void gazebo::physics::World::RemovePlugin ( const std::string & _name )`

Remove a running plugin.

#### Parameters

<code>in</code>	<code><i>_name</i></code>	The unique name of the plugin to remove.
-----------------	---------------------------	--

10.183.3.34 `void gazebo::physics::World::Reset ( )`

Reset time and model poses, configurations in simulation.

10.183.3.35 `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. 136).

#### Parameters

<code>in</code>	<code><i>_type</i></code>	The type of reset.
-----------------	---------------------------	--------------------

10.183.3.36 `void gazebo::physics::World::ResetTime ( )`

Reset simulation time back to zero.

10.183.3.37 `void gazebo::physics::World::Run ( )`

Run the world in a thread.

Run the update loop.

10.183.3.38 `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

<code>in</code>	<code><i>_filename</i></code>	Name of the file to save into.
-----------------	-------------------------------	--------------------------------

10.183.3.39 `void gazebo::physics::World::SetPaused ( bool _p )`

Set whether the simulation is paused.

#### Parameters

<code>in</code>	<code>_p</code>	True pauses the simulation. False runs the simulation.
-----------------	-----------------	--

10.183.3.40 `void gazebo::physics::World::SetSimTime ( const common::Time & _t )`

Set the sim time.

#### Parameters

<code>in</code>	<code>_t</code>	The new simulation time
-----------------	-----------------	-------------------------

10.183.3.41 `void gazebo::physics::World::SetState ( const WorldState & _state )`

Set the current world state.

#### Parameters

<code>_state</code>	The state to set the <b>World</b> (p. 875) to.	
---------------------	--	--

10.183.3.42 `void gazebo::physics::World::StepWorld ( int _steps )`

Step callback.

#### Parameters

<code>in</code>	<code>_steps</code>	The number of steps the <b>World</b> (p. 875) should take.
-----------------	---------------------	--

10.183.3.43 `void gazebo::physics::World::Stop ( )`

Stop the world.

Stop the update loop.

10.183.3.44 `std::string gazebo::physics::World::StripWorldName ( const std::string & _name ) const`

Return a version of the name with "<world\_name>::" removed.

#### Parameters

<code>in</code>	<code>_name</code>	Usually the name of an entity.
-----------------	--------------------	--------------------------------

#### Returns

The stripped world name.

10.183.3.45 void gazebo::physics::World::UpdateStateSDF ( )

Update the state SDF value from the current state.

### 10.183.4 Member Data Documentation

10.183.4.1 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. 283) on the **physics::Link** (p. 404) in World::Update.

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

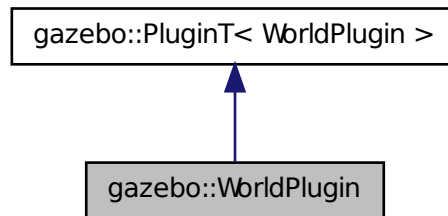
- **World.hh**

## 10.184 gazebo::WorldPlugin Class Reference

**A** (p. 107) plugin with access to **physics::World** (p. 875).

```
#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

### 10.184.1 Detailed Description

**A** (p. 107) plugin with access to **physics::World** (p. 875).

See [reference](#).

### 10.184.2 Constructor & Destructor Documentation

10.184.2.1 `gazebo::WorldPlugin::WorldPlugin ( )` `[inline]`

Constructor.

References `gazebo::PluginT< WorldPlugin >::type`, and `gazebo::WORLD_PLUGIN`.

10.184.2.2 `virtual gazebo::WorldPlugin::~~WorldPlugin ( )` `[inline],[virtual]`

Destructor.

### 10.184.3 Member Function Documentation

10.184.3.1 `virtual void gazebo::WorldPlugin::Init ( )` `[inline],[virtual]`

10.184.3.2 `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

<code>in</code>	<code>_world</code>	Pointer the World
<code>in</code>	<code>_sdf</code>	Pointer the the SDF element of the plugin.

10.184.3.3 `virtual void gazebo::WorldPlugin::Reset ( )` `[inline],[virtual]`

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

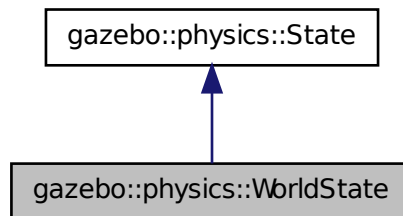
- `common/Plugin.hh`

## 10.185 gazebo::physics::WorldState Class Reference

Store state information of a **physics::World** (p. 875) 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. 888) has a **ModelState** (p. 485) 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.*

## Friends

- `std::ostream & operator<< (std::ostream &_out, const gazebo::physics::WorldState &_state)`  
*Stream insertion operator.*

## Additional Inherited Members

### 10.185.1 Detailed Description

Store state information of a **physics::World** (p. 875) object.

Instances of this class contain the state of a **World** (p. 875) at a specific time. **World** (p. 875) state includes the state of all models, and their children.

### 10.185.2 Constructor & Destructor Documentation

#### 10.185.2.1 gazebo::physics::WorldState::WorldState ( )

Default constructor.

#### 10.185.2.2 gazebo::physics::WorldState::WorldState ( const WorldPtr \_world ) [explicit]

Constructor.

Generate a **WorldState** (p. 888) from an instance of a **World** (p. 875).

#### Parameters

in	_world	Pointer to a world
----	--------	--------------------

#### 10.185.2.3 gazebo::physics::WorldState::WorldState ( const sdf::ElementPtr \_sdf ) [explicit]

Constructor.

Build a **WorldState** (p. 888) from SDF data

#### Parameters

in	_sdf	SDF data to load a world state from.
----	------	--------------------------------------

#### 10.185.2.4 virtual gazebo::physics::WorldState::~~WorldState ( ) [virtual]

Destructor.

### 10.185.3 Member Function Documentation

#### 10.185.3.1 void gazebo::physics::WorldState::FillSDF ( sdf::ElementPtr \_sdf )

Populate a state SDF element with data from the object.

## Parameters

out	_sdf	SDF element to populate.
-----	------	--------------------------

## 10.185.3.2 ModelState gazebo::physics::WorldState::GetModelState ( unsigned int \_index ) const

Get a model state.

Get the state of a **Model** (p. 469) based on an index. The min index is and the max is **WorldState::GetModelStateCount()** (p. 891).

## Parameters

in	_index	Index of the model.
----	--------	---------------------

## Returns

**State** (p. 729) of the requested **Model** (p. 469).

## 10.185.3.3 ModelState gazebo::physics::WorldState::GetModelState ( const std::string &amp; \_modelName ) const

Get a model state by model name.

## Parameters

in	_modelName	Name of the model state to get.
----	------------	---------------------------------

## Returns

The model state.

## Exceptions

<b>common::Exception</b> (p. 310)	When the _modelName doesn't exist.
--------------------------------------	------------------------------------

## 10.185.3.4 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.

## 10.185.3.5 const std::vector&lt;ModelState&gt;&amp; gazebo::physics::WorldState::GetModelStates ( ) const

Get the model states.

**Returns**

**A** (p. 107) vector of model states.

10.185.3.6 `bool gazebo::physics::WorldState::HasModelState ( const std::string & _modelName ) const`

Return true if **WorldState** (p. 888) has a **ModelState** (p. 485) with the given name.

**Parameters**

<code>in</code>	<code><i>_modelName</i></code>	Name of the model to search for.
-----------------	--------------------------------	----------------------------------

**Returns**

True if the **ModelState** (p. 485) exists.

10.185.3.7 `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.

10.185.3.8 `virtual void gazebo::physics::WorldState::Load ( const sdf::ElementPtr _elem ) [virtual]`

Load state from SDF element.

Set a **WorldState** (p. 888) from an SDF element containing **WorldState** (p. 888) info.

**Parameters**

<code>in</code>	<code><i>_elem</i></code>	Pointer to the <b>WorldState</b> (p. 888) SDF element.
-----------------	---------------------------	--

Reimplemented from **gazebo::physics::State** (p. 732).

10.185.3.9 `WorldState gazebo::physics::WorldState::operator+ ( const WorldState & _state ) const`

Addition operator.

**Parameters**

<code>in</code>	<code><i>_pt</i></code>	<b>A</b> (p. 107) state to add.
-----------------	-------------------------	---------------------------------

**Returns**

The resulting state.



10.185.3.10 WorldState gazebo::physics::WorldState::operator- ( const WorldState & *\_state* ) const

Subtraction operator.

## Parameters

<i>in</i>	<i>_pt</i>	<b>A</b> (p. 107) state to subtract.
-----------	------------	--------------------------------------

## Returns

The resulting state.

10.185.3.11 WorldState& gazebo::physics::WorldState::operator= ( const WorldState & *\_state* )

Assignment operator.

## Parameters

<i>in</i>	<i>_state</i>	<b>State</b> (p. 729) value
-----------	---------------	-----------------------------

## Returns

Reference to this

## 10.185.4 Friends And Related Function Documentation

10.185.4.1 std::ostream& operator<< ( std::ostream & *\_out*, const gazebo::physics::WorldState & *\_state* ) [friend]

Stream insertion operator.

## Parameters

<i>in</i>	<i>_out</i>	output stream
<i>in</i>	<i>_state</i>	<b>World</b> (p. 875) state to output

## Returns

the stream

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

- **WorldState.hh**



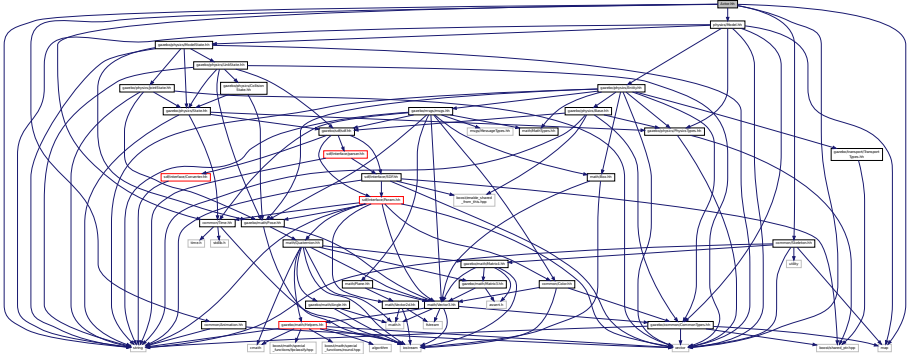
# Chapter 11

## File Documentation

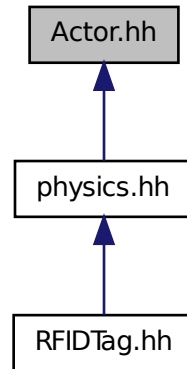
### 11.1 Actor.hh File Reference

```
#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. 107) class enables GPU based mesh model / skeleton scriptable animation.
- struct **gazebo::physics::TrajectoryInfo**

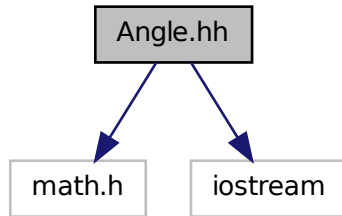
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::common**  
*Common namespace.*
- namespace **gazebo::physics**  
*namespace for physics*

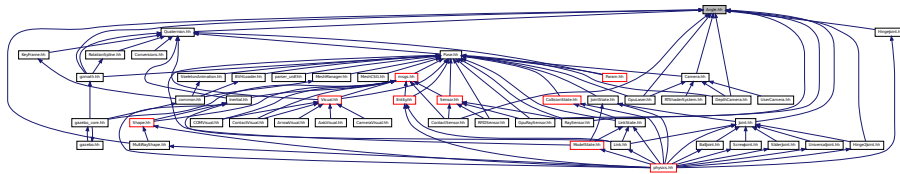
## 11.2 Angle.hh File Reference

```
#include <math.h>  
#include <iostream>
```

Include dependency graph for Angle.hh:



This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::math::Angle**  
*An angle and related functions.*

## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::math**  
*Math namespace.*

## Macros

- #define **GZ\_DTOR**(d) ((d) \* M\_PI / 180)  
*Converts degrees to radians.*
- #define **GZ\_NORMALIZE**(a) (atan2(sin(a), cos(a)))  
*Macro tha normalizes an angle in the range -Pi to Pi.*
- #define **GZ\_RTOD**(r) ((r) \* 180 / M\_PI)  
*Macro that converts radians to degrees.*

## 11.2.1 Macro Definition Documentation

### 11.2.1.1 `#define GZ_DTOR( d ) ((d) * M_PI / 180)`

Converts degrees to radians.

#### Parameters

<i>in</i>	<i>degrees</i>	
-----------	----------------	--

#### Returns

radians

### 11.2.1.2 `#define GZ_NORMALIZE( a ) (atan2(sin(a), cos(a)))`

Macro tha normalizes an angle in the range -Pi to Pi.

#### Parameters

<i>in</i>	<i>angle</i>	
-----------	--------------	--

#### Returns

the angle, in range

### 11.2.1.3 `#define GZ_RTOD( r ) ((r) * 180 / M_PI)`

Macro that converts radians to degrees.

#### Parameters

<i>in</i>	<i>radians</i>	
-----------	----------------	--

## Returns

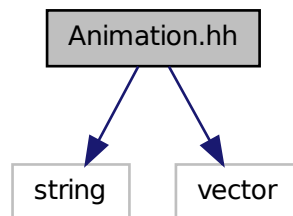
degrees

## 11.3 Animation.hh File Reference

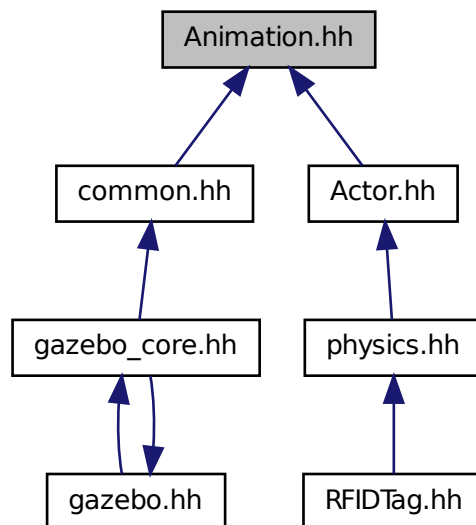
```
#include <string>
```

```
#include <vector>
```

Include dependency graph for Animation.hh:



This graph shows which files directly or indirectly include this file:



## Classes

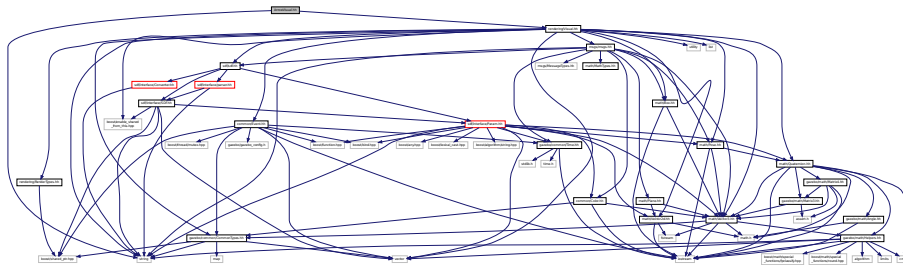
- class **gazebo::common::Animation**  
*Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.*
- class **gazebo::common::NumericAnimation**  
*A (p. 107) numeric animation.*
- class **gazebo::common::PoseAnimation**  
*A (p. 107) pose animation.*

## 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

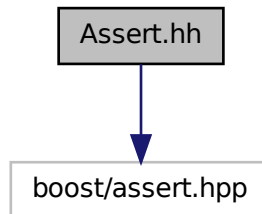
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::rendering**  
*Rendering namespace.*
- namespace **ogre**



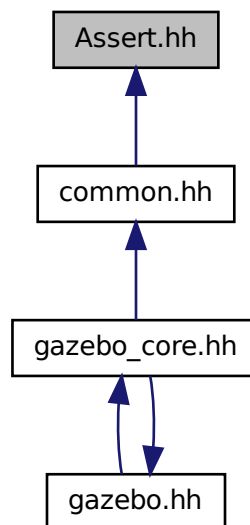
## 11.5 Assert.hh File Reference

```
#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.*

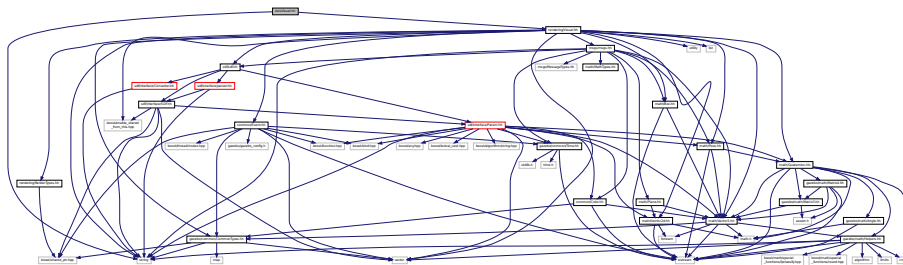
## 11.5.1 Macro Definition Documentation

### 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.*

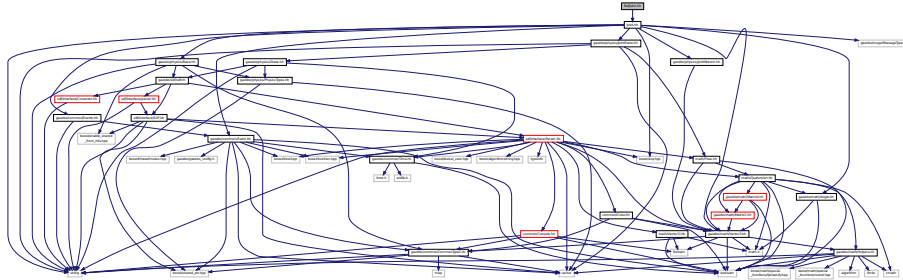
- namespace **gazebo::rendering**

*Rendering namespace.*

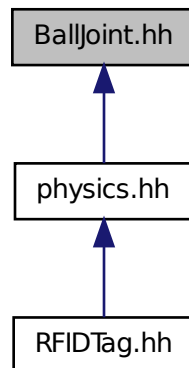
## 11.7 BallJoint.hh File Reference

```
#include "Joint.hh"
```

Include dependency graph for BallJoint.hh:



This graph shows which files directly or indirectly include this file:



### Classes

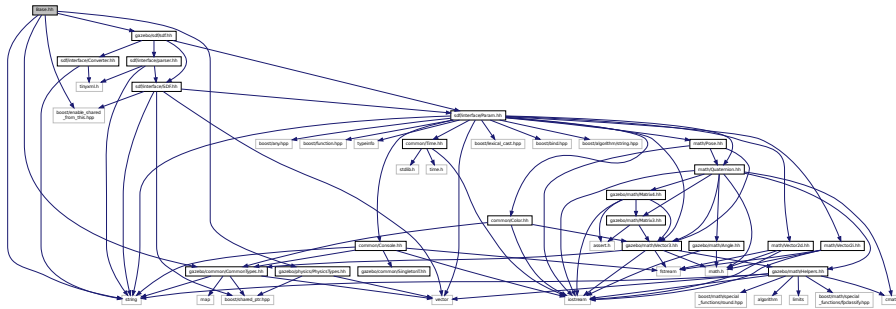
- class **gazebo::physics::BallJoint**< T >  
*Base (p. 133) class for a ball joint.*

### 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. 133) class for most physics classes.

### Namespaces

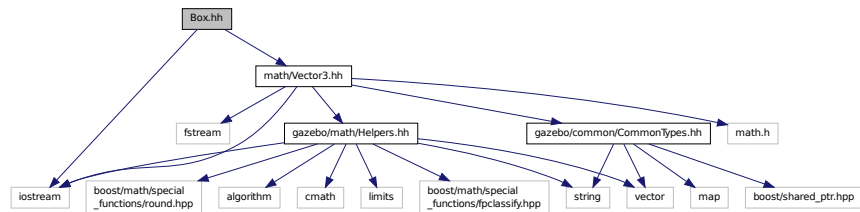
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*

### Variables

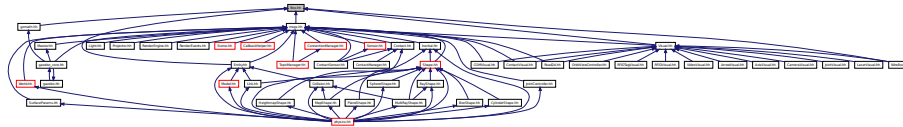
- static std::string **gazebo::physics::EntityTypeName []**  
*String names for the different entity types.*

## 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.*

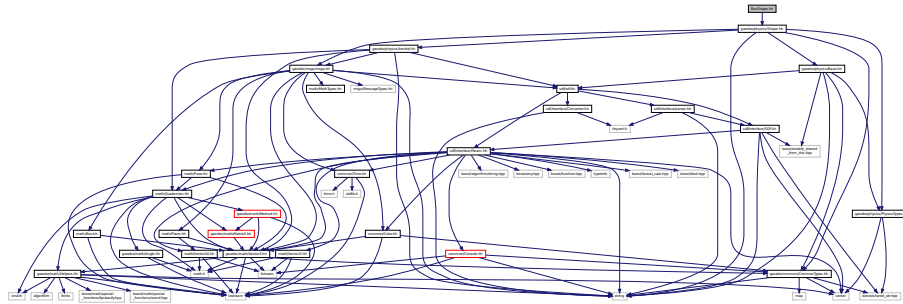
- namespace **gazebo::math**

*Math namespace.*

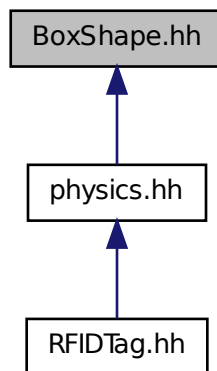
## 11.10 BoxShape.hh File Reference

```
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for BoxShape.hh:



This graph shows which files directly or indirectly include this file:



## Classes

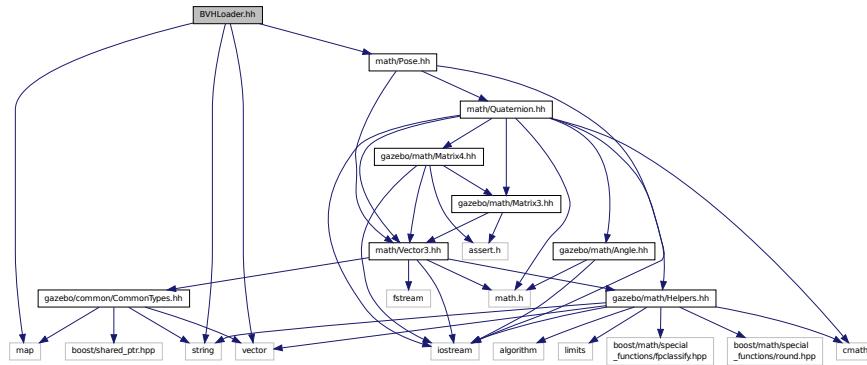
- class **gazebo::physics::BoxShape**  
*Box geometry primitive.*

## Namespaces

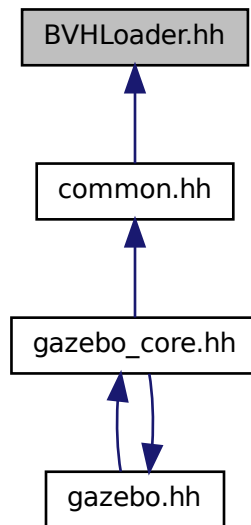
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*

## 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`

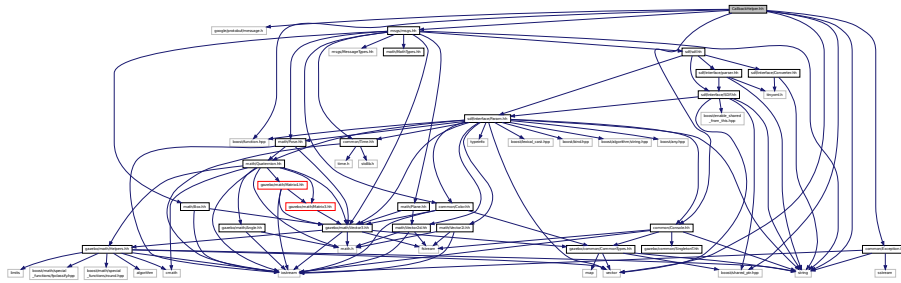
11.11.1.6 `#define Z_ROTATION 5`

## 11.12 CallbackHelper.hh File Reference

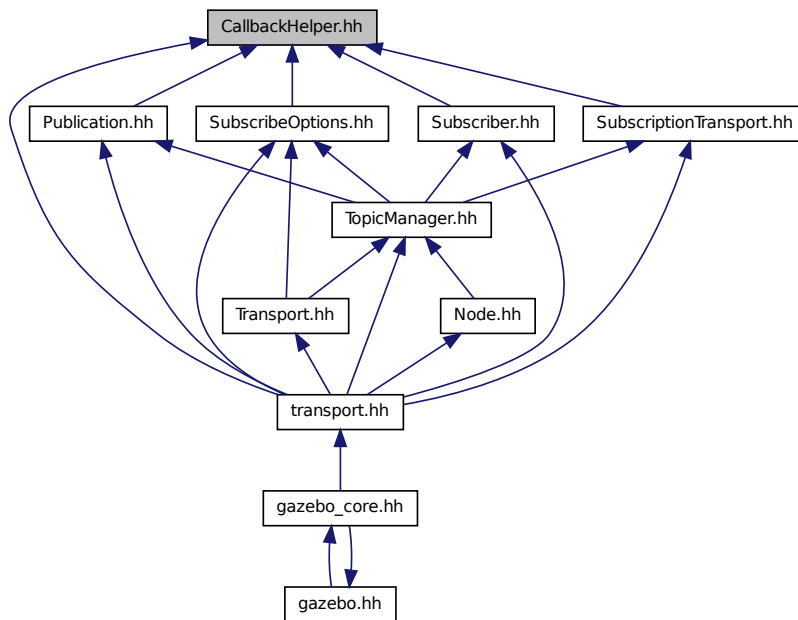
```
#include <google/protobuf/message.h>
#include <boost/function.hpp>
#include <boost/shared_ptr.hpp>
#include <vector>
#include <string>
#include "common/Console.hh"
#include "msgs/msgs.hh"
#include "common/Exception.hh"
```



Include dependency graph for CallbackHelper.hh:



This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::transport::CallbackHelper**  
*A (p. 107) 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

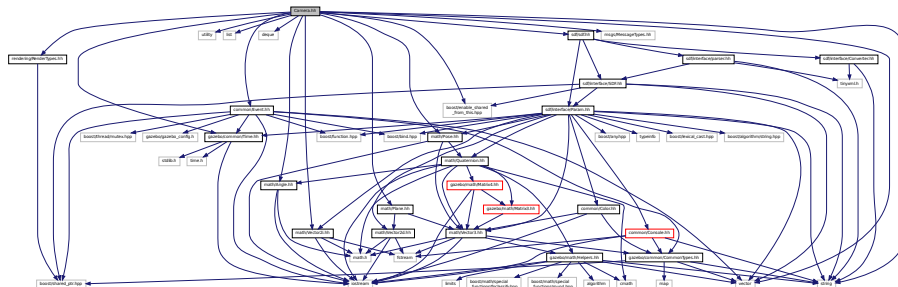
- typedef CallbackHelper \* **gazebo::transport::CallbackHelperPtr**

*boost shared pointer to **transport::CallbackHelper** (p. 153)*

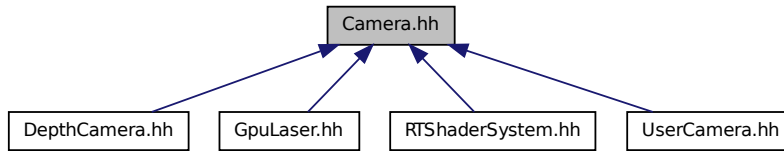
## 11.13 Camera.hh File Reference

```
#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**

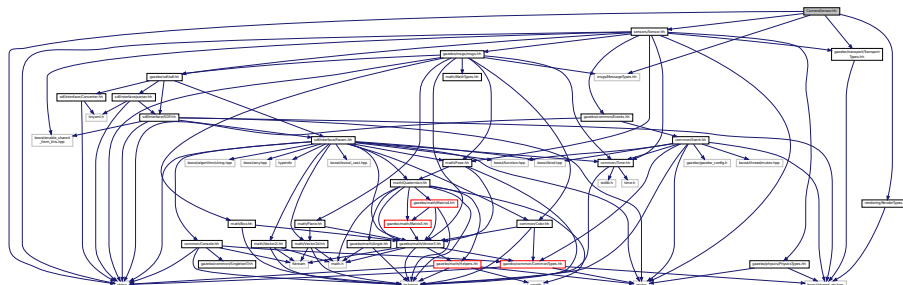
## 11.14 CameraSensor.hh File Reference

```

#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**

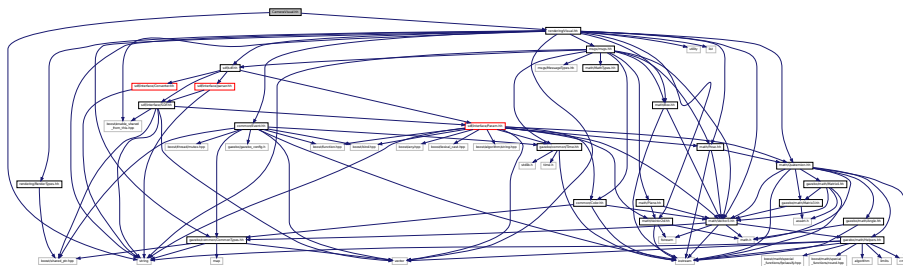
*Basic camera sensor.*

## 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.*

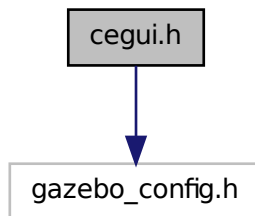
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::rendering**  
*Rendering namespace.*

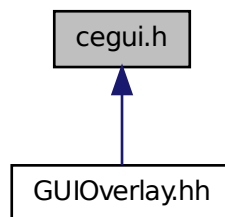
## 11.16 cegui.h File Reference

```
#include "gazebo_config.h"
```

Include dependency graph for cegui.h:



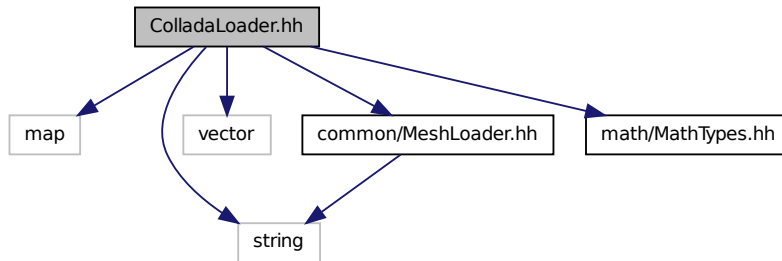
This graph shows which files directly or indirectly include this file:



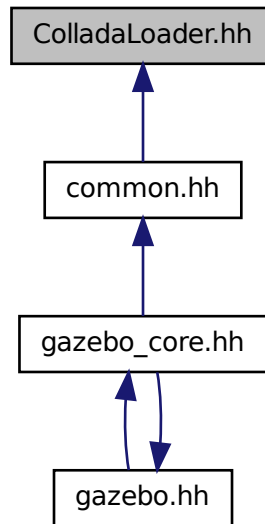
## 11.17 ColladaLoader.hh File Reference

```
#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**

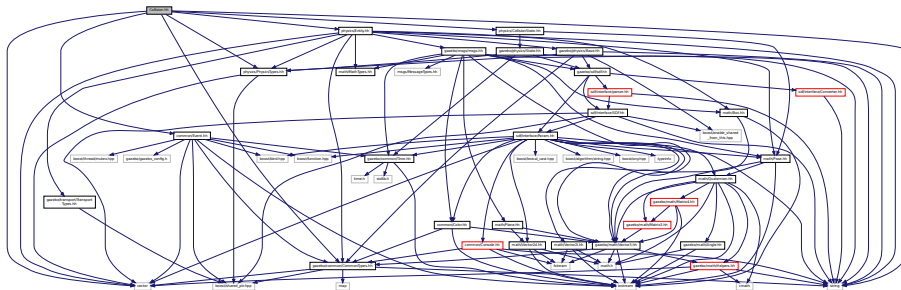
*Forward declarations for the common classes.*

- namespace **gazebo::common**

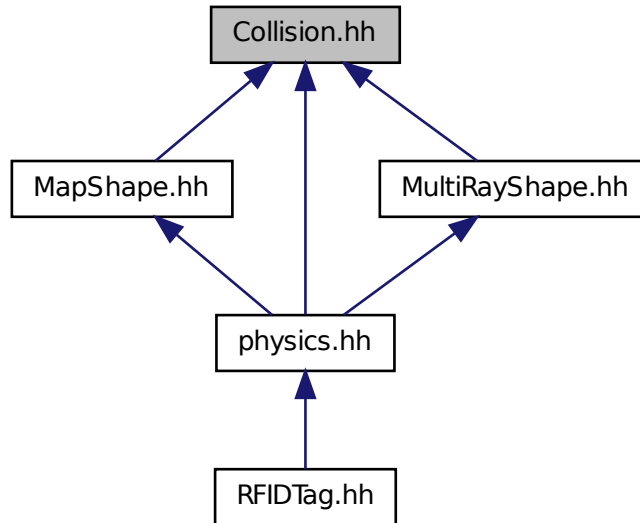
*Common namespace.*

## 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. 133) class for all collision entities.*

## Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

- namespace **gazebo::physics**

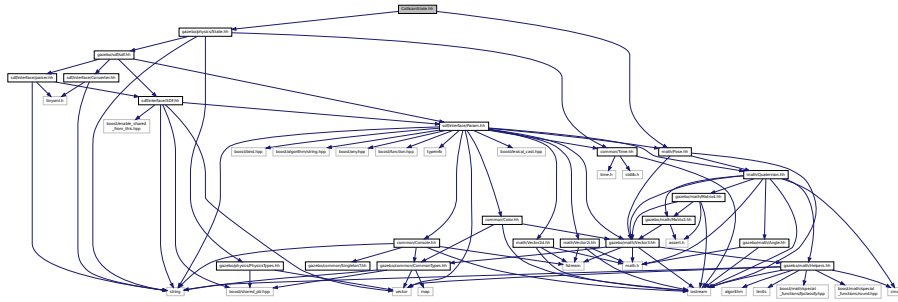
*namespace for physics*

## 11.19 CollisionState.hh File Reference

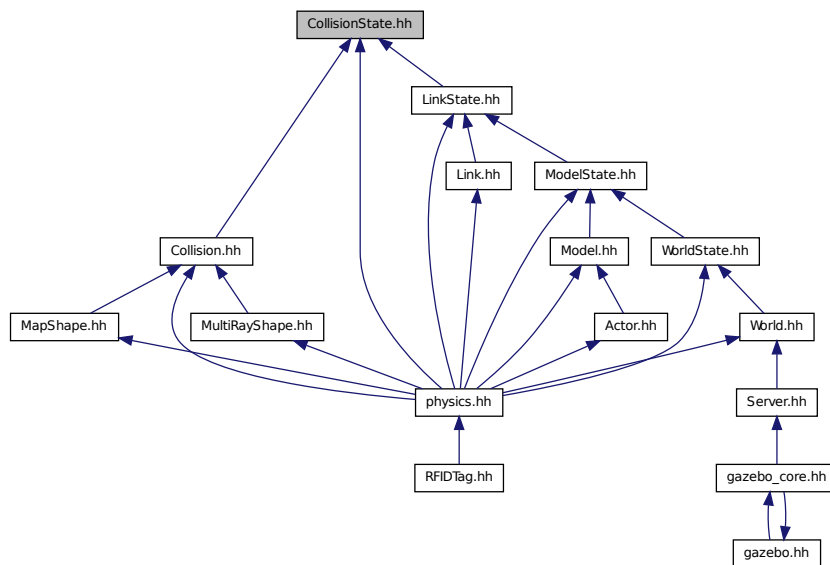
```
#include "gazebo/physics/State.hh"
#include "gazebo/math/Pose.hh"
```



Include dependency graph for CollisionState.hh:



This graph shows which files directly or indirectly include this file:



## Classes

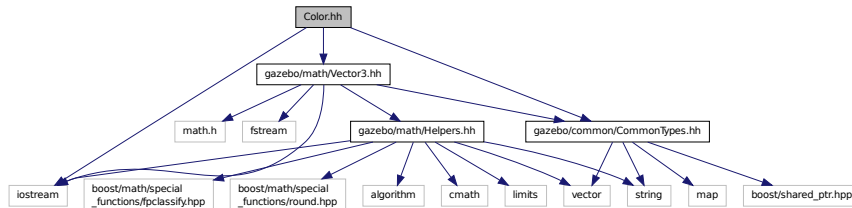
- class **gazebo::physics::CollisionState**  
Store state information of a *physics::Collision* (p. 190) object.

## Namespaces

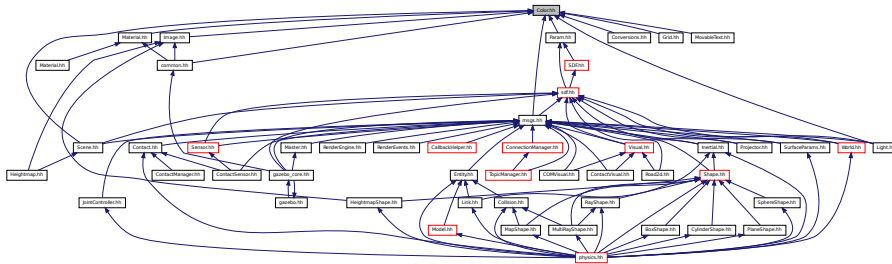
- namespace **gazebo**  
Forward declarations for the common classes.
- namespace **gazebo::physics**  
namespace for physics

## 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.*

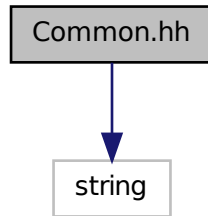
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::common**  
*Common namespace.*

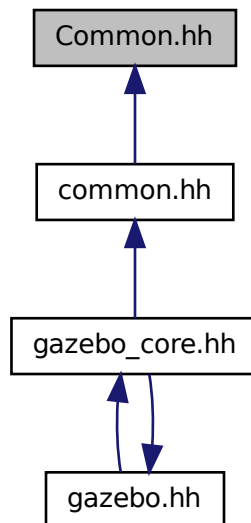
## 11.21 Common.hh File Reference

```
#include <string>
```

Include dependency graph for Common.hh:



This graph shows which files directly or indirectly include this file:



## Namespaces

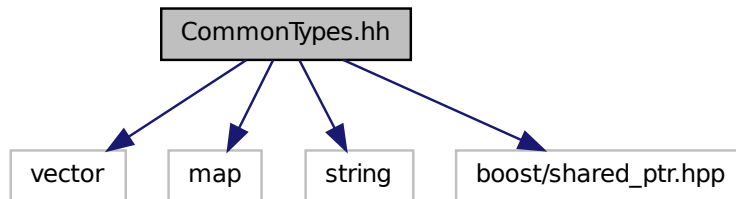
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::common**  
*Common namespace.*

## Functions

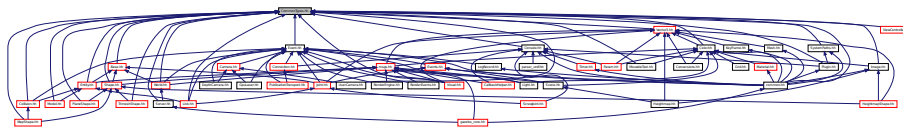
- void **gazebo::common::add\_search\_path\_suffix** (const std::string &\_suffix)  
*add path prefix to **common::SystemPaths** (p. 754)*
- std::string **gazebo::common::find\_file** (const std::string &\_file, bool \_searchLocalPath=true)  
*search for file in **common::SystemPaths** (p. 754)*
- std::string **gazebo::common::find\_file\_path** (const std::string &\_file)  
*search for a file in **common::SystemPaths** (p. 754)*

## 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. 285) 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 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**

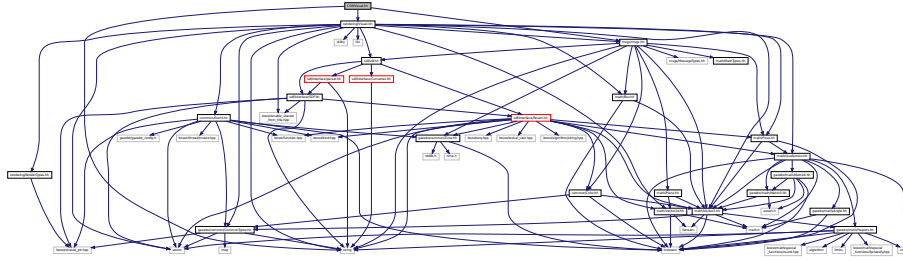
11.22.1.3 #define **NULL** 0

Referenced by gazebo::transport::TopicManager::Advertise(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::event::EventT< T >::Disconnect(), gazebo::transport::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.*

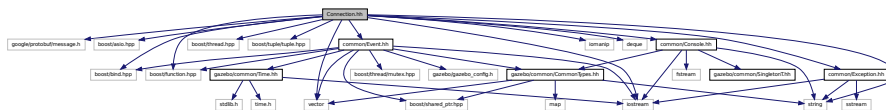
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::rendering**  
*Rendering namespace.*
- namespace **ogre**

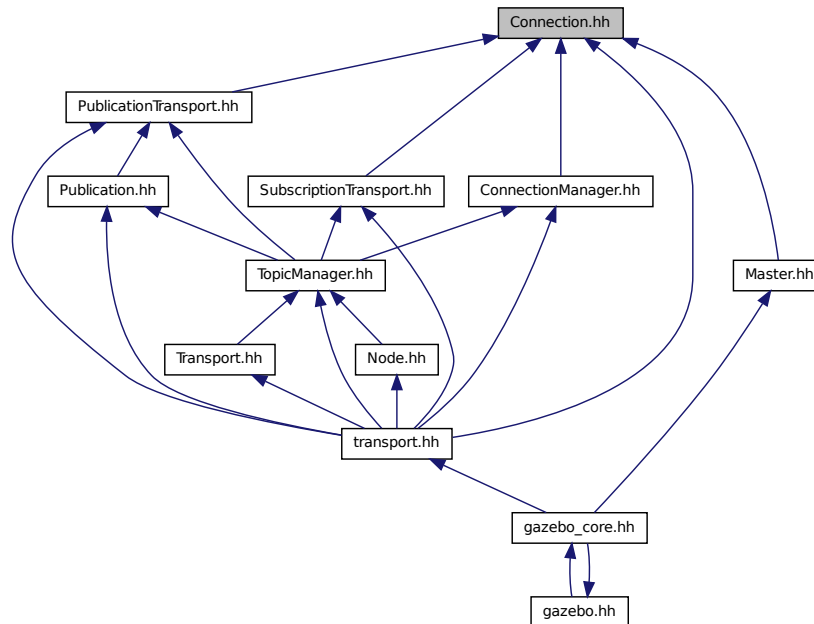
## 11.24 Connection.hh File Reference

```
#include <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.*

## 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

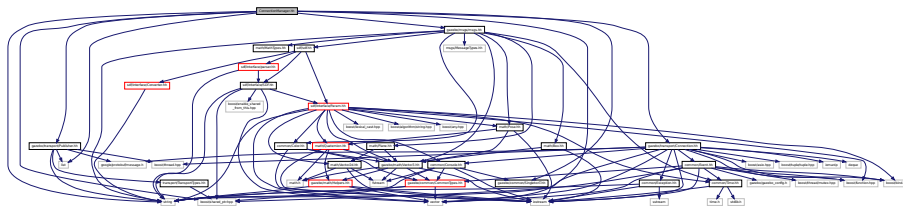
### 11.24.1.1 #define HEADER\_LENGTH 8

Referenced by gazebo::transport::Connection::AsyncRead().

## 11.25 ConnectionManager.hh File Reference

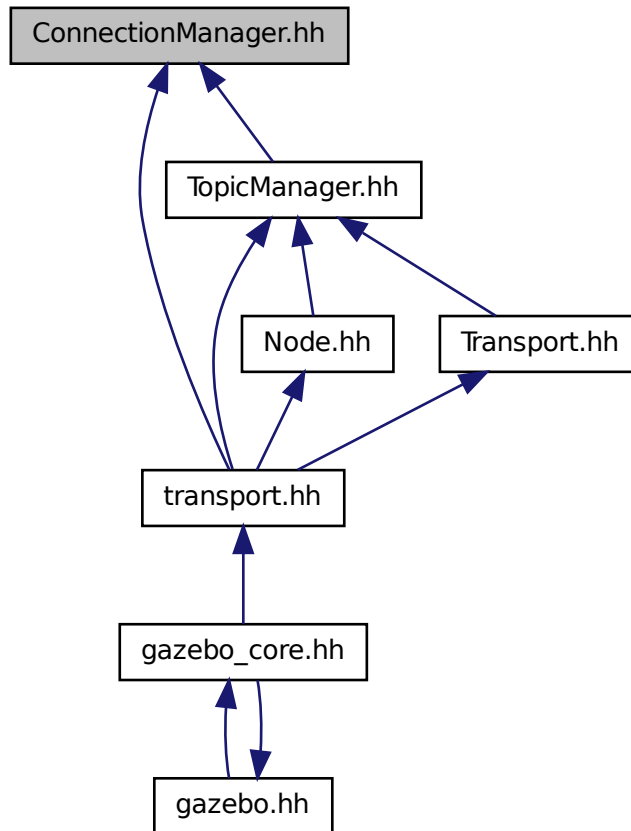
```
#include <boost/shared_ptr.hpp>
#include <string>
#include <list>
#include <vector>
#include "gazebo_msgs/msgs.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/transport/Publisher.hh"
#include "gazebo/transport/Connection.hh"
```

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.*

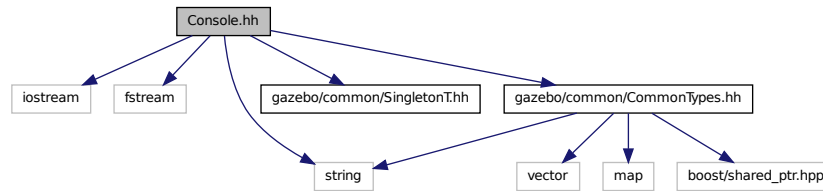
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::transport**

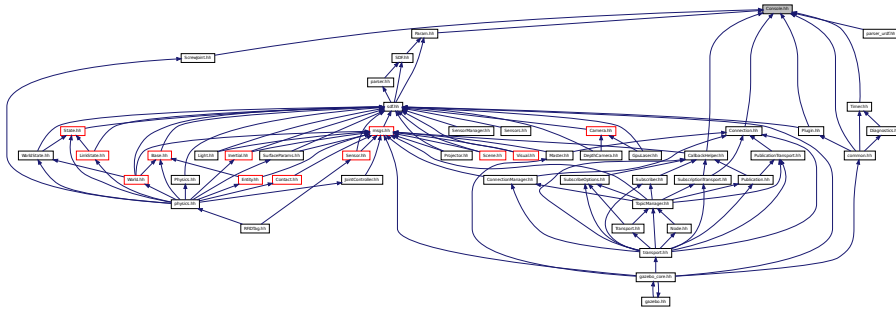
## 11.26 Console.hh File Reference

```
#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.*

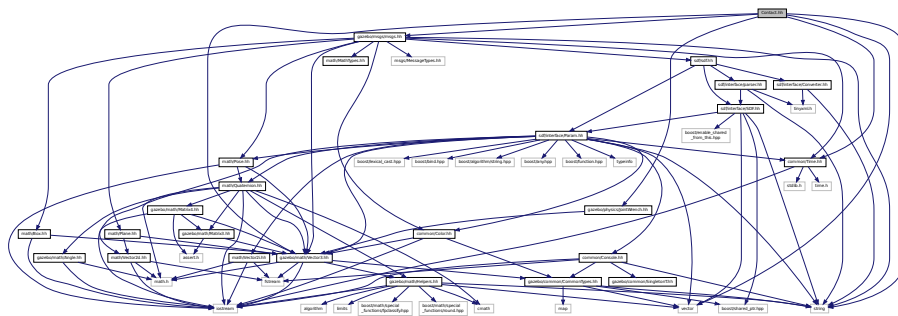
- #define **gzwarn**

*Output a warning message.*

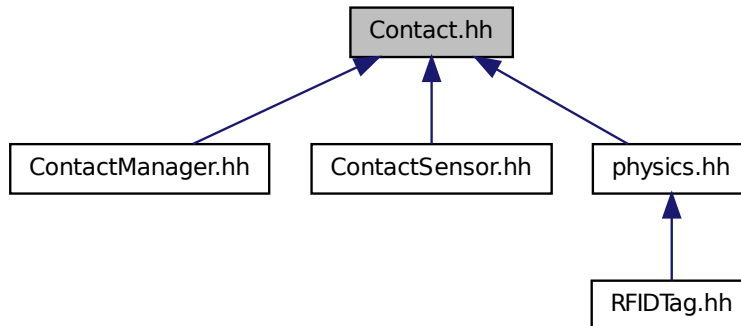
## 11.27 Contact.hh File Reference

```
#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. 107) 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

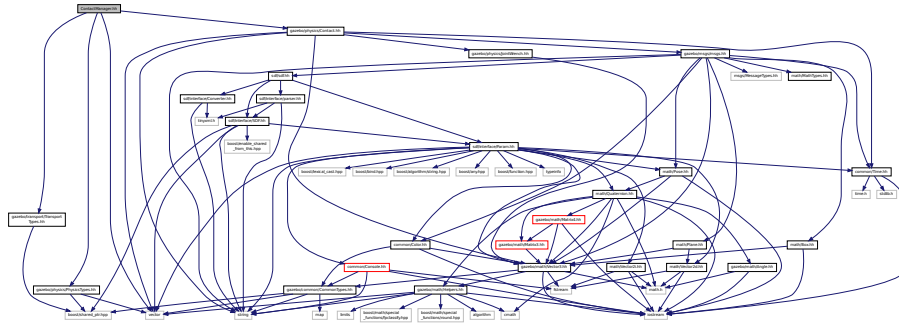
11.27.1.2 #define **MAX\_CONTACT\_JOINTS** 32

## 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

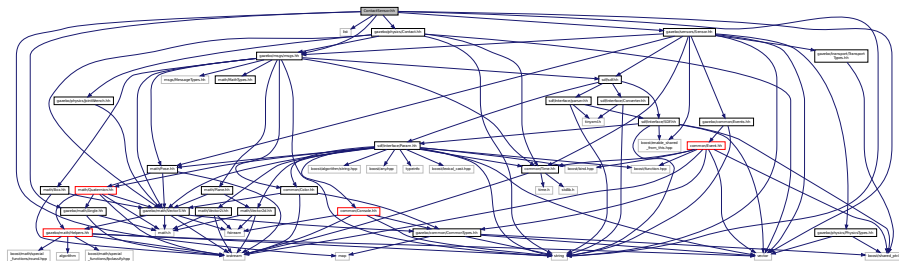
- class **gazebo::physics::ContactManager**  
*Aggregates all the contact information generated by the collision detection engine.*

## 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.*

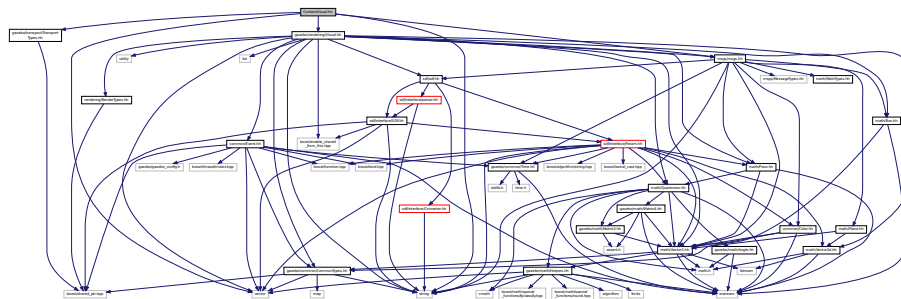
## Namespaces

- 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. 240) *Conversions.hh* (p. 931) *rendering/Conversions.hh* (p. 931).

### Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

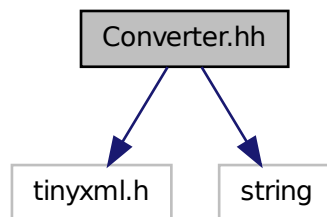
- namespace **gazebo::rendering**

*Rendering namespace.*

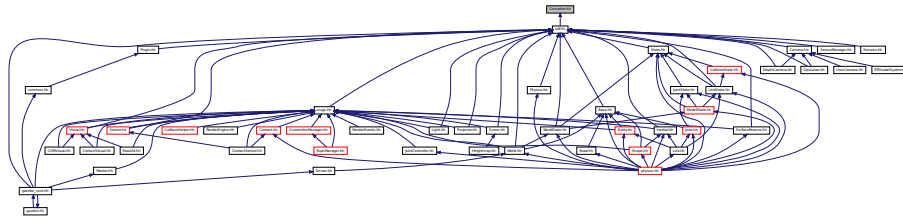
## 11.32 Converter.hh File Reference

```
#include <tinycl.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. 669) to another.*

## Namespaces

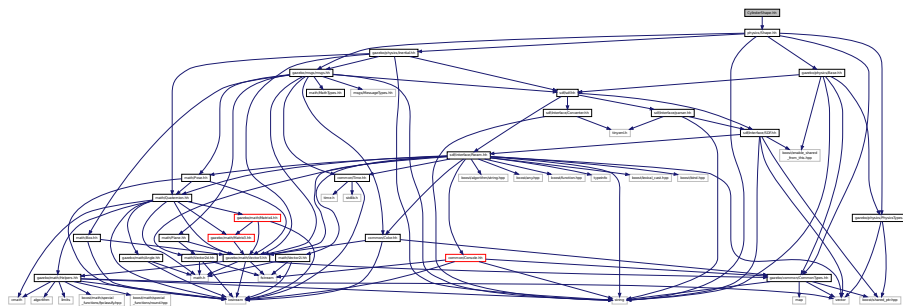
- namespace **sdf**

*namespace for Simulation Description Format parser*

## 11.33 CylinderShape.hh File Reference

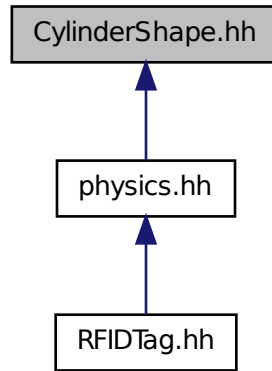
```
#include "physics/Shape.hh"
```

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.*

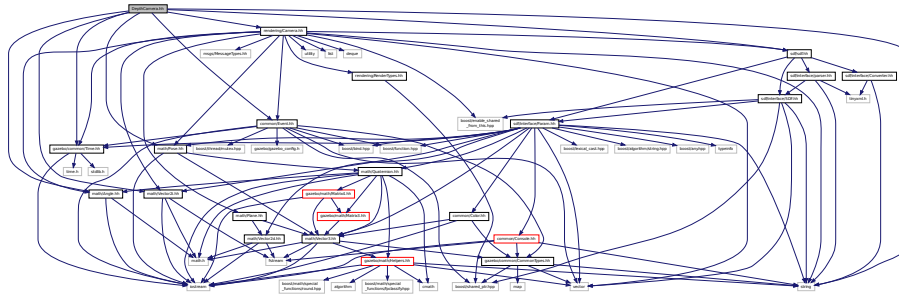
- namespace **gazebo::physics**

*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

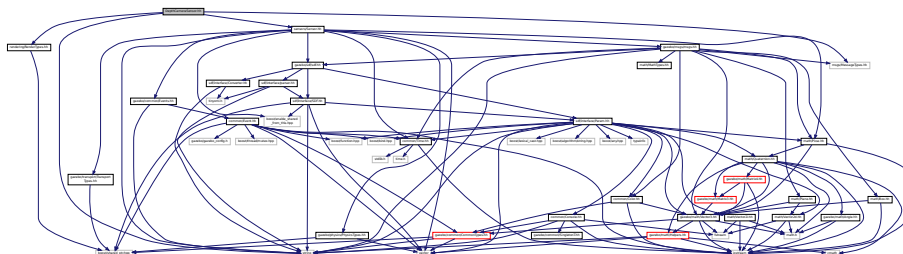
- class **gazebo::rendering::DepthCamera**  
*Depth camera used to render depth data into an image buffer.*

## 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**

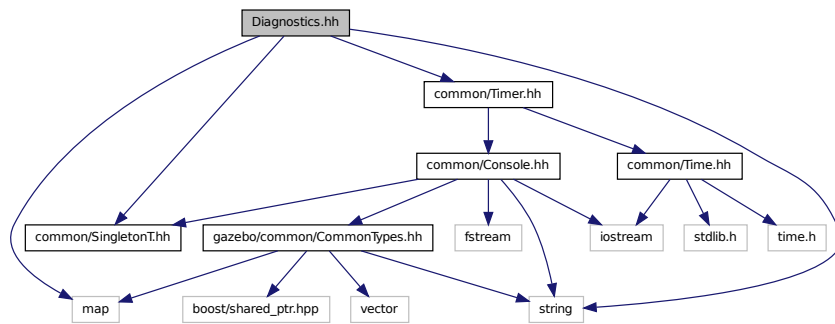
*Forward declarations for the common classes.*

- namespace **gazebo::sensors**

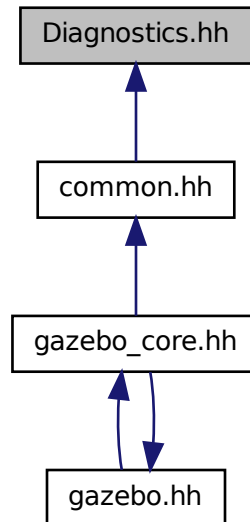
*Sensors namespace.*

## 11.36 Diagnostics.hh File Reference

```
#include <map>
#include <string>
#include "common/SingletonT.hh"
#include "common/Timer.hh"
Include dependency graph for Diagnostics.hh:
```



This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::common::DiagnosticManager**  
*A (p. 107) diagnostic manager class.*
- class **gazebo::common::DiagnosticTimer**  
*A (p. 107) timer designed for diagnostics.*

## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::common**  
*Common namespace.*

## Macros

- **#define DIAG\_TIMER(name) DiagnosticManager::Instance()->CreateTimer(name);**  
*Create an instance of common::DiagnosticManager.*

## Typedefs

- typedef DiagnosticTimer \* **gazebo::common::DiagnosticTimerPtr**

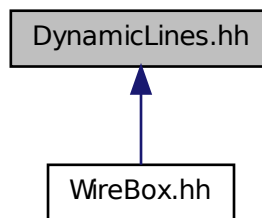
## 11.37 DynamicLines.hh File Reference

```
#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.*

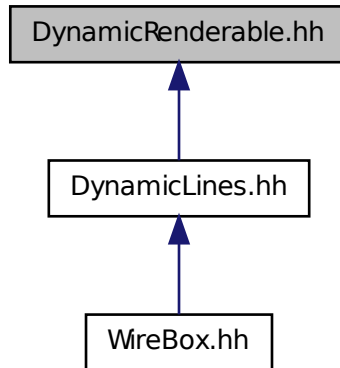
## 11.38 DynamicRenderable.hh File Reference

```
#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**

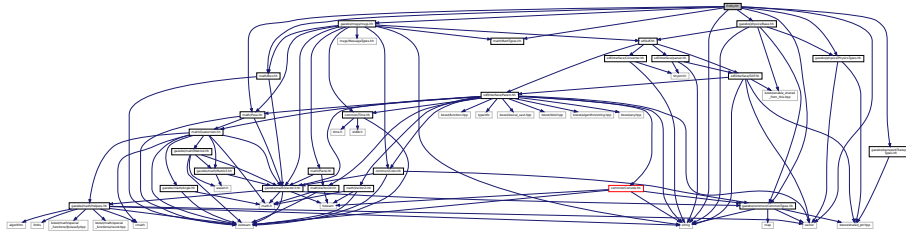
*Rendering namespace.*

## 11.39 Entity.hh File Reference

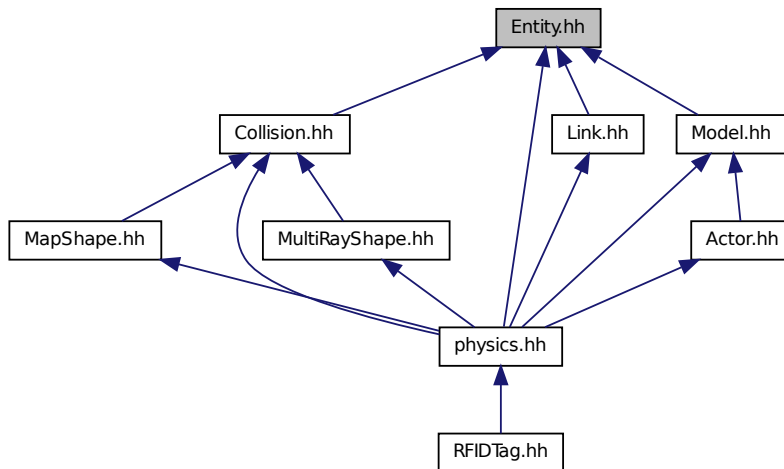
```

#include <string>
#include <vector>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/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. 133) class for all physics objects in Gazebo.*

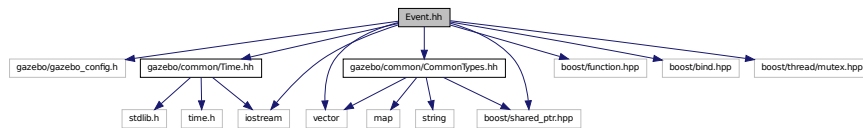
## Namespaces

- 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. 107) class that encapsulates a connection.
- class **gazebo::event::Event**  
Base class for all events.
- class **gazebo::event::EventT < T >**  
A (p. 107) class for event processing.

## Namespaces

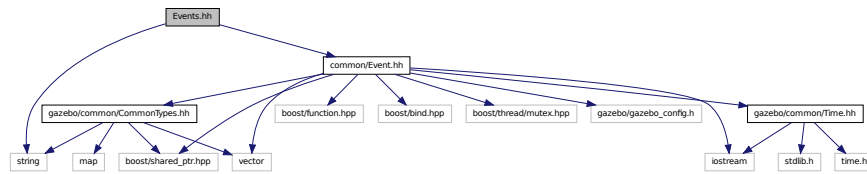
- namespace **gazebo**  
Forward declarations for the common classes.
- namespace **gazebo::event**  
*Event* (p. 285) namespace.

## 11.41 Events.hh File Reference

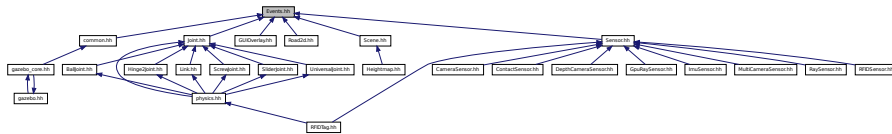
```
#include <string>
#include "common/Event.hh"
```



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. 285) class to get notifications for simulator events.

## Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

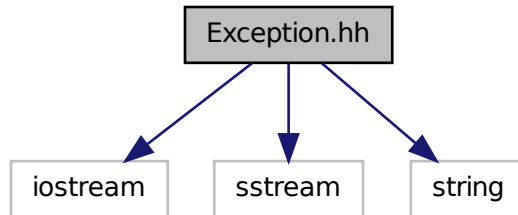
- namespace **gazebo::event**

*Event* (p. 285) namespace.

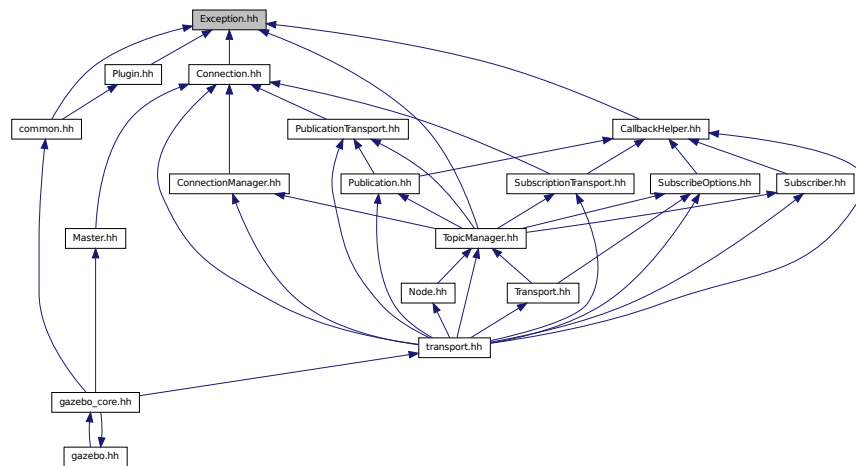
## 11.42 Exception.hh File Reference

```
#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.*

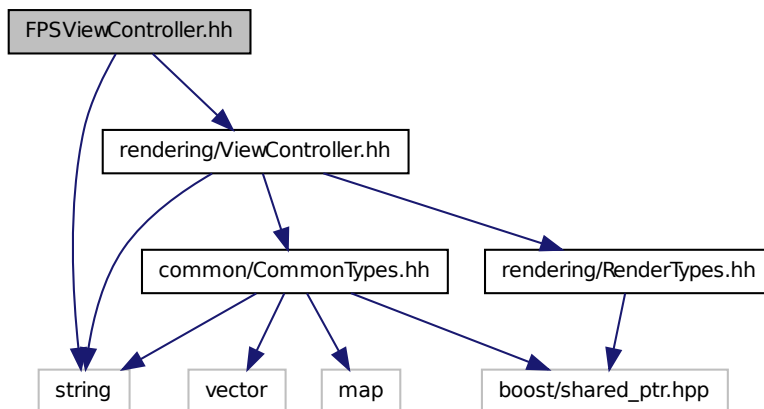
## Macros

- #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**

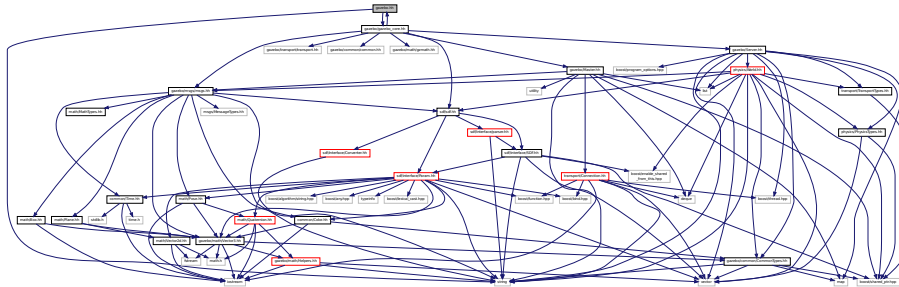
*Rendering namespace.*

## 11.44 gazebo.hh File Reference

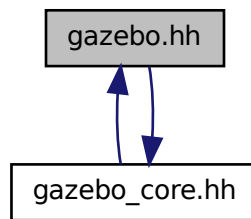
```
#include <gazebo/gazebo_core.hh>
```

```
#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.*

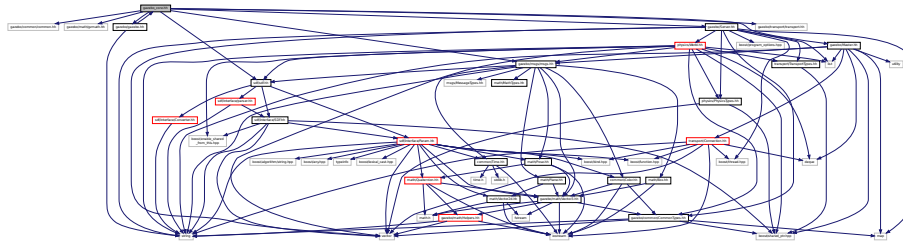
### Functions

- 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** ()

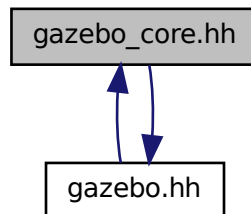
## 11.45 gazebo\_core.hh File Reference

```
#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:



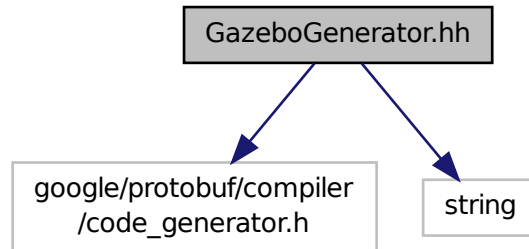
This graph shows which files directly or indirectly include this file:



## 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. 87).*

## 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

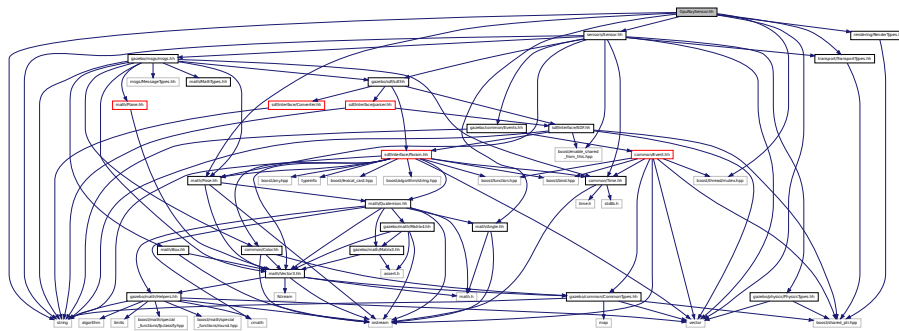
- class **gazebo::rendering::GpuLaser**  
*GPU based laser distance sensor.*

## 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.*

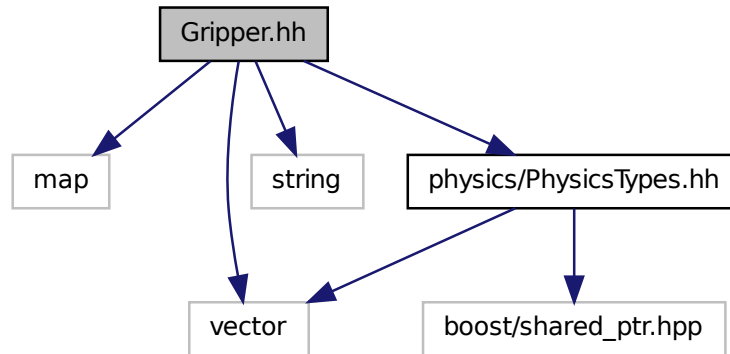
- namespace **Ogre**

## 11.50 Gripper.hh File Reference

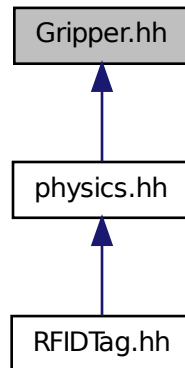
```
#include <map>
#include <vector>
#include <string>
#include "physics/PhysicsTypes.hh"
```



Include dependency graph for Gripper.hh:



This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::physics::Gripper**  
A (p. 107) gripper abstraction.

## Namespaces

- namespace **gazebo**

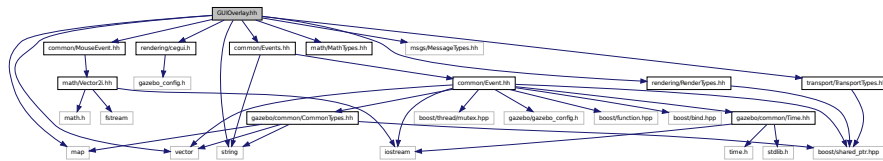
Forward declarations for the common classes.

- 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. 107) 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.*

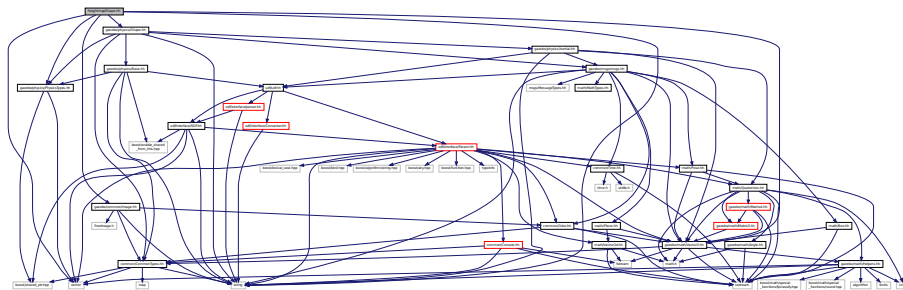
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::rendering**  
*Rendering namespace.*
- namespace **Ogre**

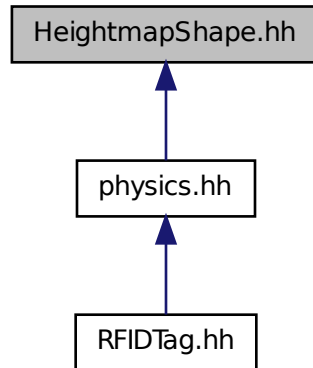
## 11.53 HeightmapShape.hh File Reference

```
#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. 344) collision shape builds a heightmap from an image.

## Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

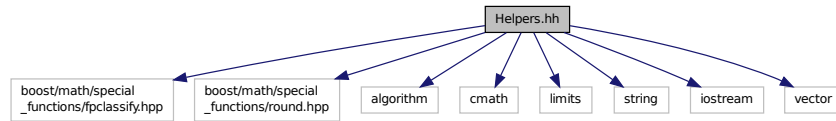
- namespace **gazebo::physics**

*namespace for physics*

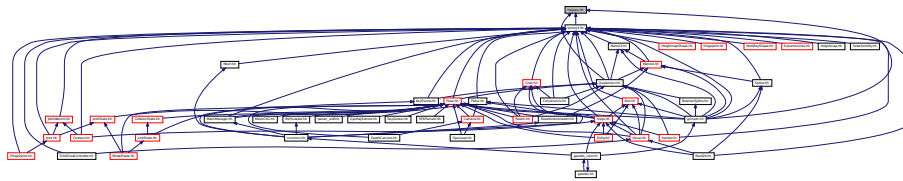
## 11.54 Helpers.hh File Reference

```
#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()`

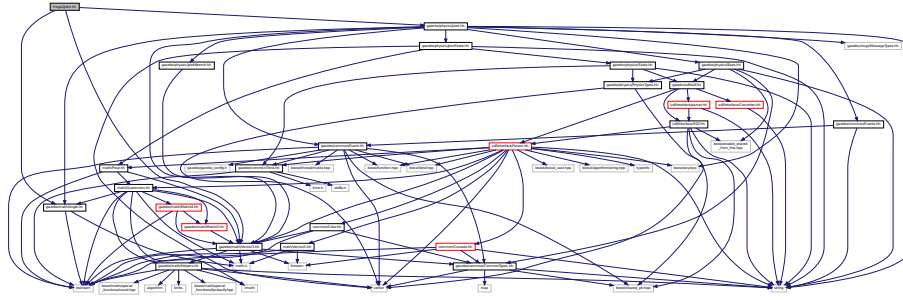
11.54.1.3 `#define GZ_FLT_MAX std::numeric_limits<float>::max()`

11.54.1.4 `#define GZ_FLT_MIN std::numeric_limits<float>::min()`

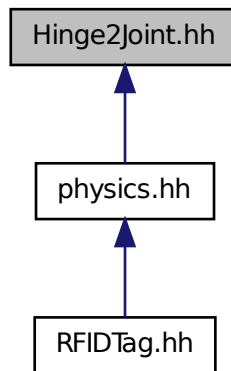
## 11.55 Hinge2Joint.hh File Reference

```
#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

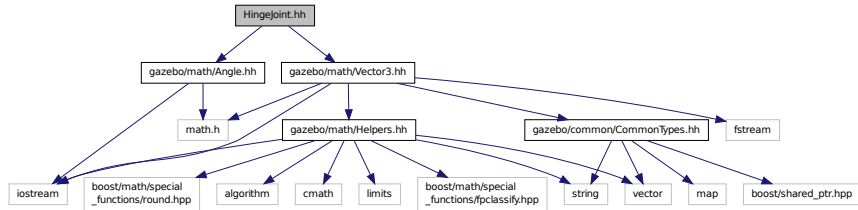
- class **gazebo::physics::Hinge2Joint**< T >  
*A (p. 107) two axis hinge joint.*

## Namespaces

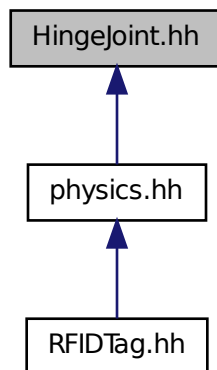
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*

## 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. 107) single axis hinge joint.*

### Namespaces

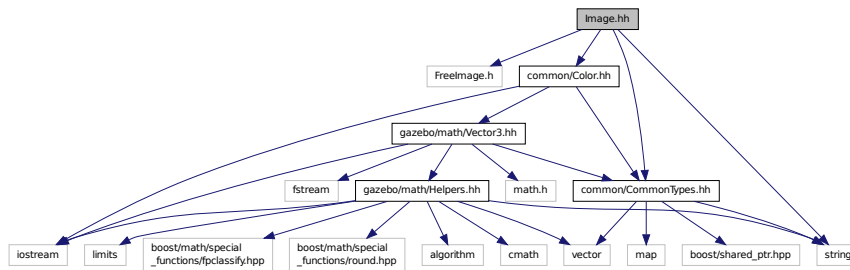
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*



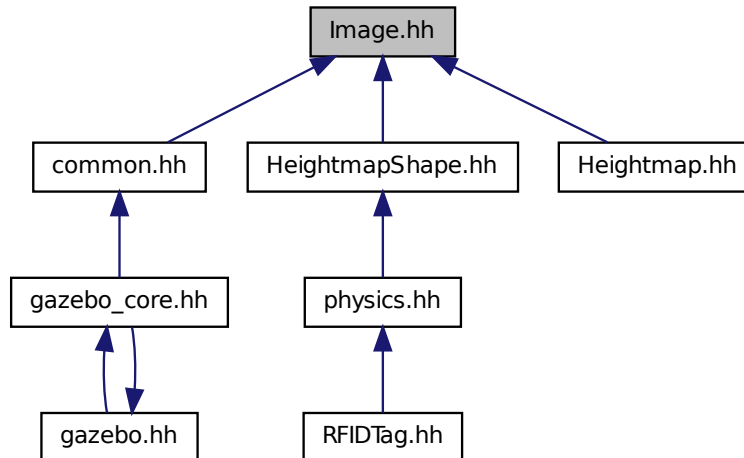
## 11.57 Image.hh File Reference

```
#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.*

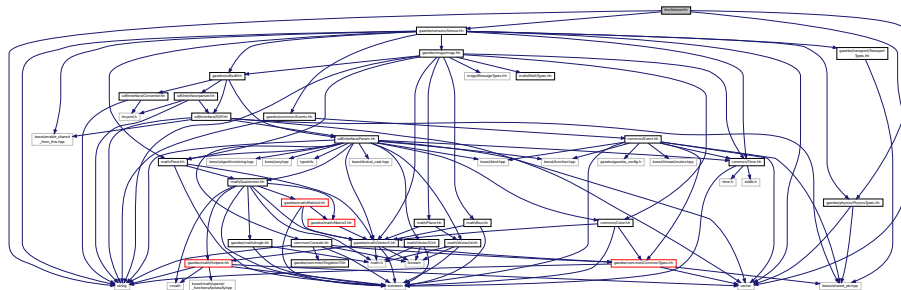
## Variables

- 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.*

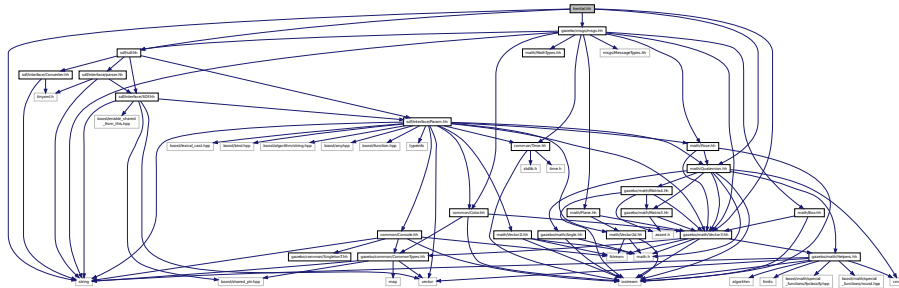
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::sensors**  
*Sensors namespace.*

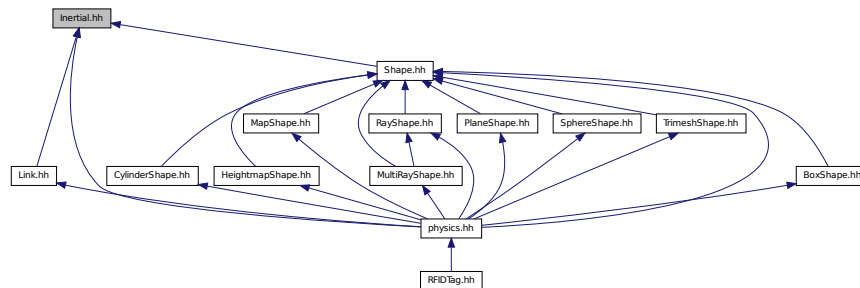
## 11.59 Inertial.hh File Reference

```
#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. 107) class for inertial information about a link.*

### Namespaces

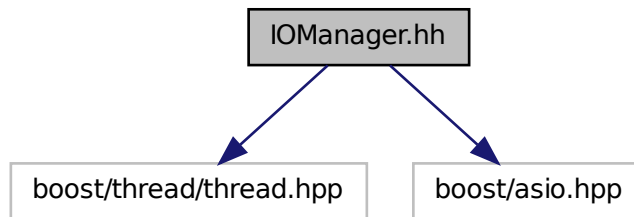
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*

## 11.60 IOManager.hh File Reference

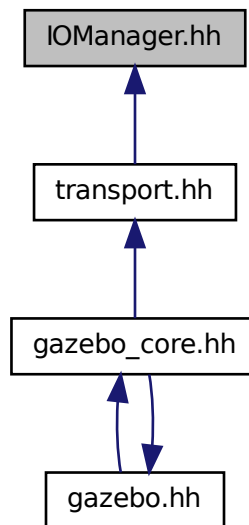
```
#include <boost/thread/thread.hpp>
```

```
#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.*

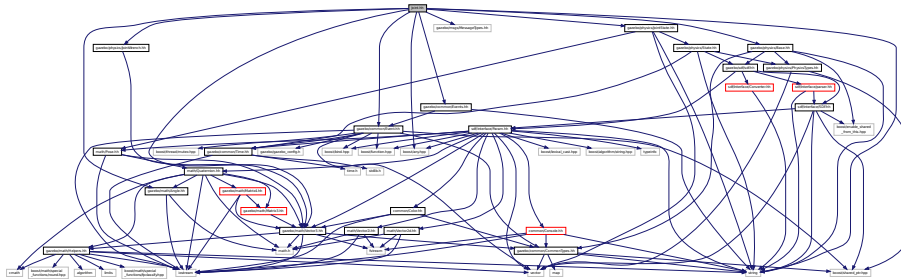
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::transport**

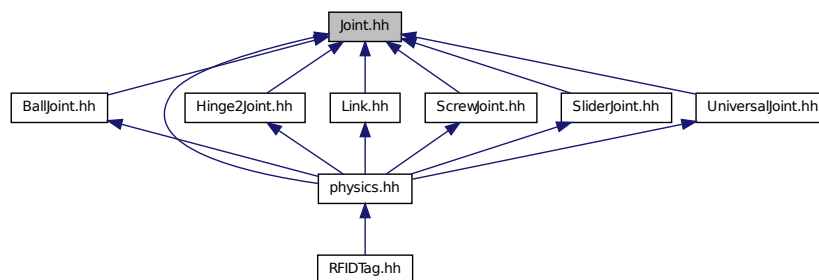
## 11.61 Joint.hh File Reference

```
#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. 133) class for all joints.*

## Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

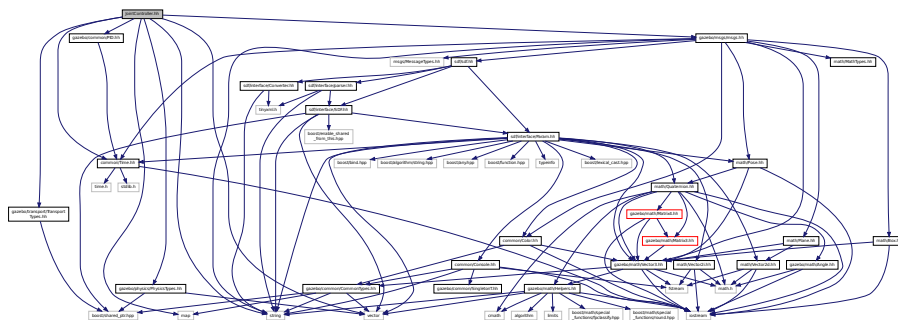
- namespace **gazebo::physics**

*namespace for physics*

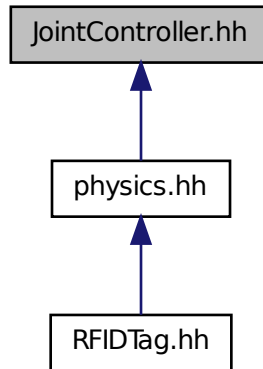
## 11.62 JointController.hh File Reference

```
#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 msgs/msgs.hh"
```

Include dependency graph for JointController.hh:



This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::physics::JointController**

*A* (p. 107) class for manipulating **physics::Joint** (p. 371).

## Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

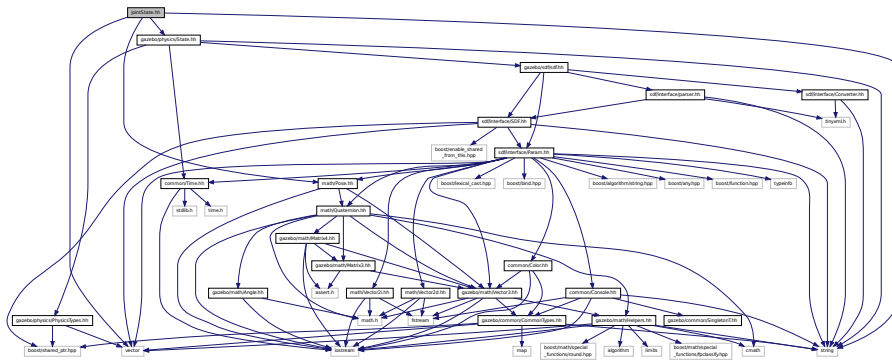
- namespace **gazebo::physics**

*namespace for physics*

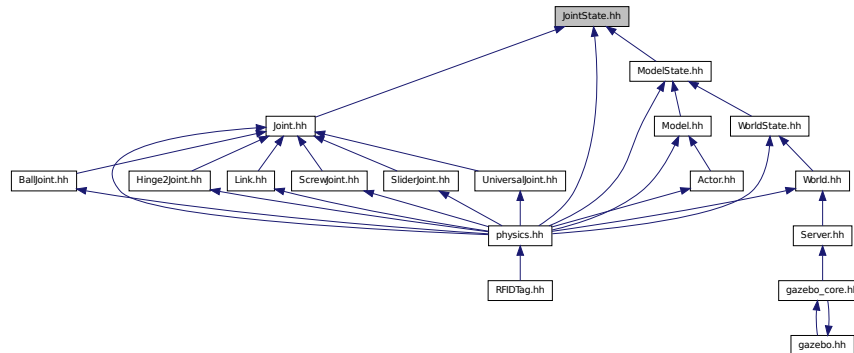
## 11.63 JointState.hh File Reference

```
#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. 371)

## Namespaces

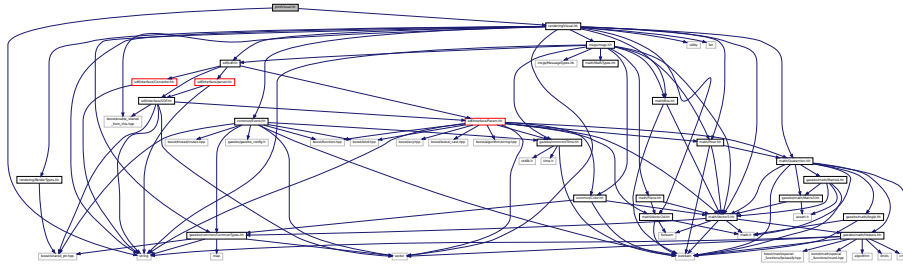
- namespace **gazebo**  
Forward declarations for the common classes.
- namespace **gazebo::physics**  
namespace for physics

## 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.*

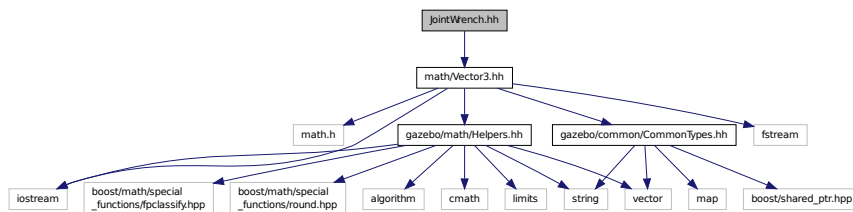
- namespace **gazebo::rendering**

*Rendering namespace.*

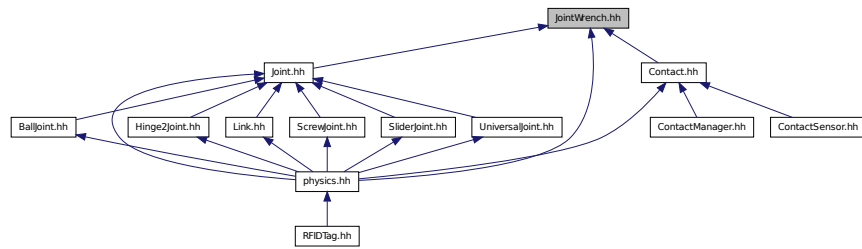
## 11.65 JointWrench.hh File Reference

```
#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

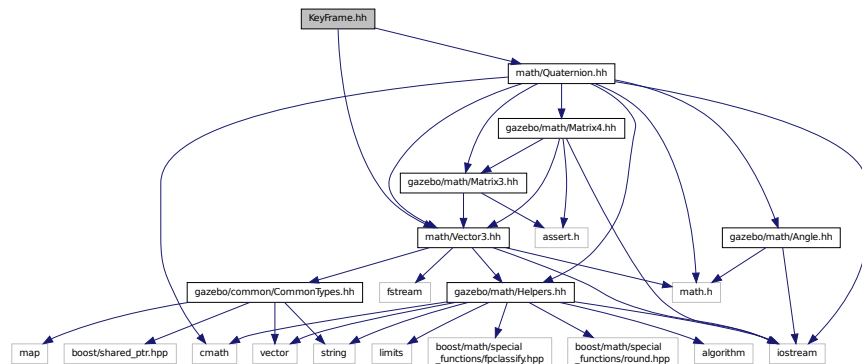
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*

## 11.66 KeyFrame.hh File Reference

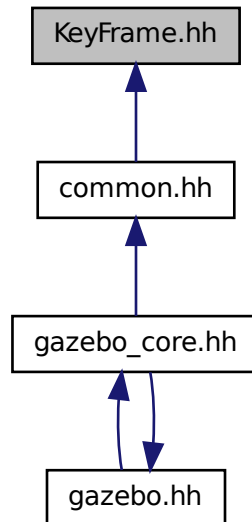
```
#include "math/Vector3.hh"
```

```
#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. 107) key frame in an animation.*
- class **gazebo::common::NumericKeyFrame**  
*A (p. 107) keyframe for a **NumericAnimation** (p. 531).*
- class **gazebo::common::PoseKeyFrame**  
*A (p. 107) keyframe for a **PoseAnimation** (p. 581).*

## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::common**  
*Common namespace.*

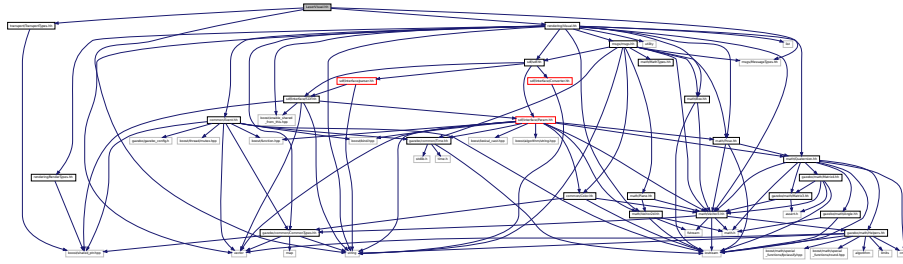
## 11.67 LaserVisual.hh File Reference

```

#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.*

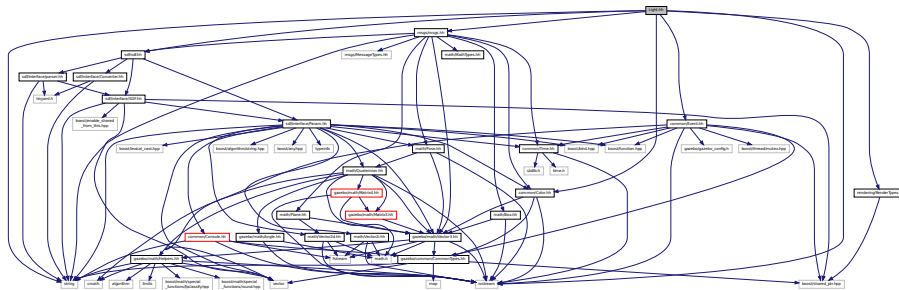
## Namespaces

- 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. 107) light source.*

## Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

- namespace **gazebo::rendering**

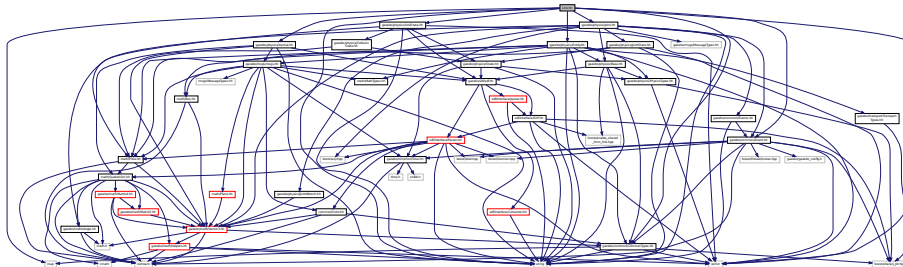
*Rendering namespace.*

- namespace **Ogre**

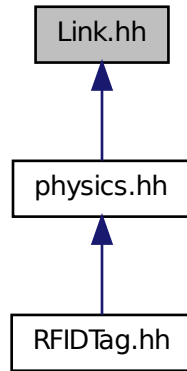
## 11.69 Link.hh File Reference

```
#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. 404) 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.*

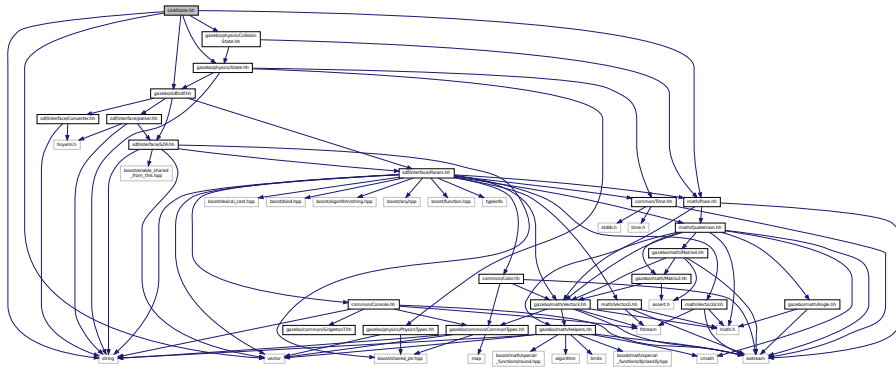
- namespace **gazebo::physics**

*namespace for physics*

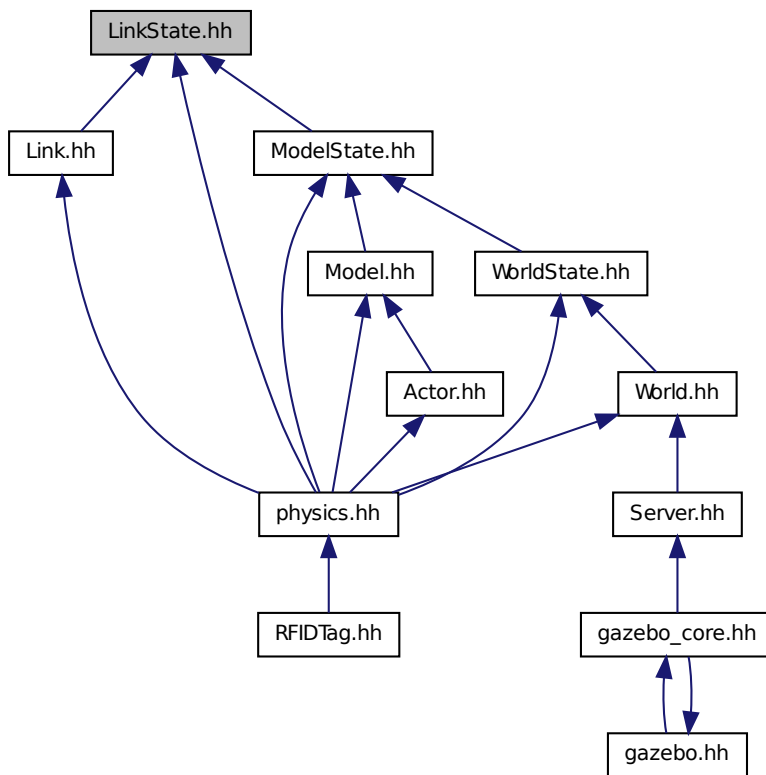
## 11.70 LinkState.hh File Reference

```
#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. 404) object.

## Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

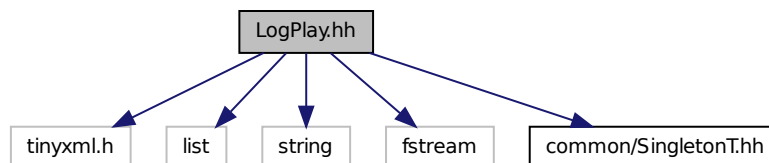
- namespace **gazebo::physics**

*namespace for physics*

## 11.71 LogPlay.hh File Reference

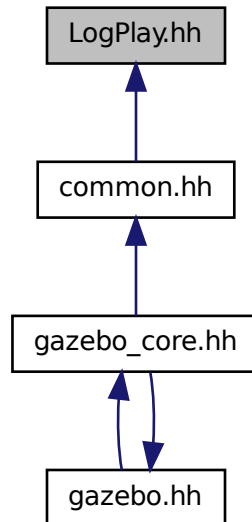
```
#include <tinyxml.h>
#include <list>
#include <string>
#include <fstream>
#include "common/SingletonT.hh"
```

Include dependency graph for LogPlay.hh:





This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::common::LogPlay**

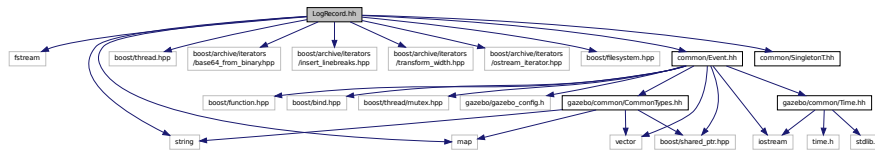
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::common**  
*Common namespace.*

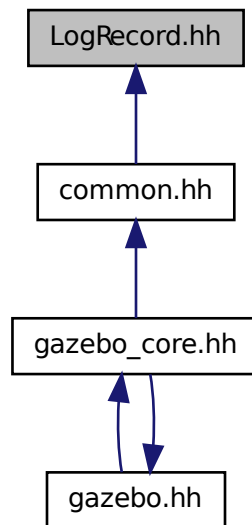
## 11.72 LogRecord.hh File Reference

```
#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 "common/Event.hh"
#include "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

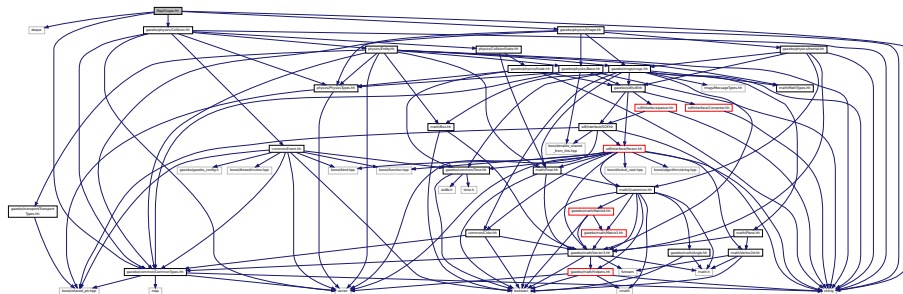
#### 11.72.1.1 `#define GZ_LOG_VERSION "1.0"`

## 11.73 mainpage.html File Reference

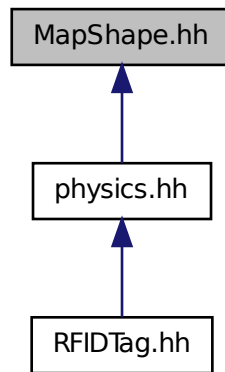
## 11.74 MapShape.hh File Reference

```
#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:



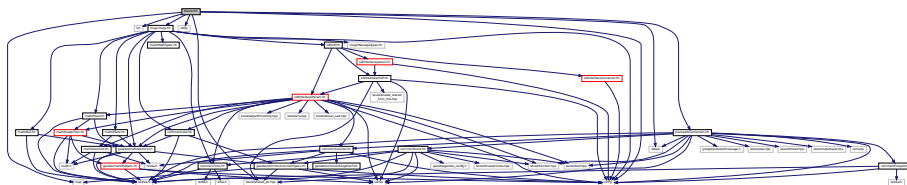
This graph shows which files directly or indirectly include this file:



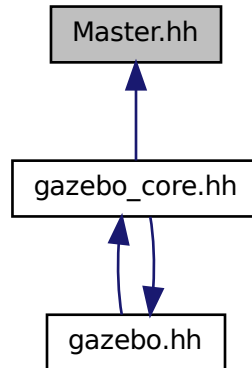
## 11.75 Master.hh File Reference

```
#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. 107) ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

## Namespaces

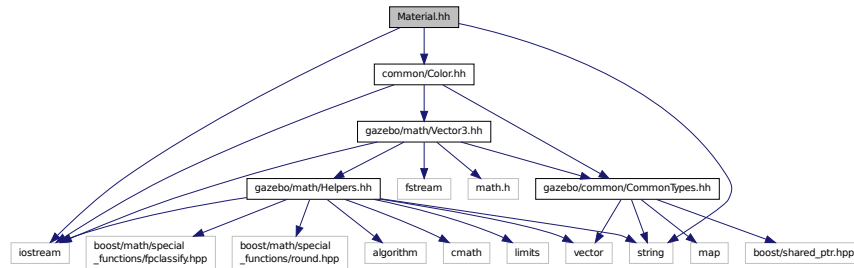
- namespace **gazebo**

*Forward declarations for the common classes.*

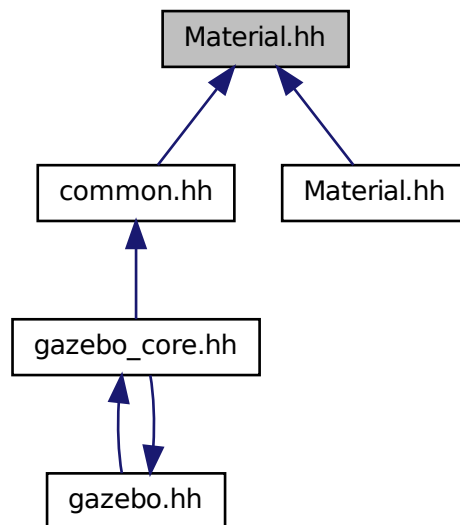
## 11.76 Material.hh File Reference

```
#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

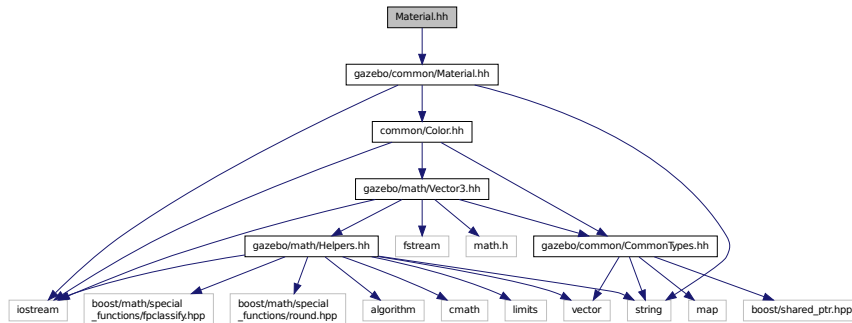
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::common**

*Common namespace.*

## 11.77 Material.hh File Reference

```
#include "gazebo/common/Material.hh"
```

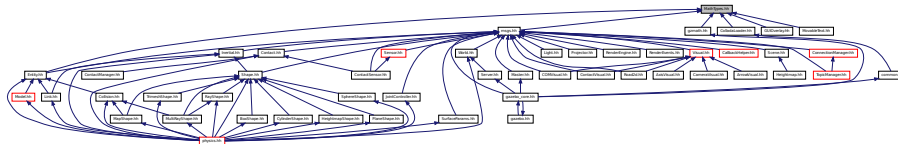
Include dependency graph for rendering/Material.hh:



## 11.78 MathTypes.hh File Reference

Forward declarations for the math classes.

This graph shows which files directly or indirectly include this file:



## Namespaces

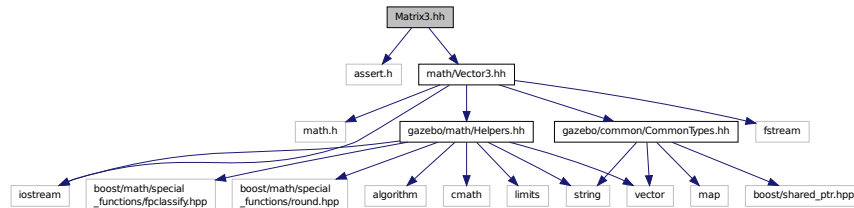
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::math**  
*Math namespace.*

### 11.78.1 Detailed Description

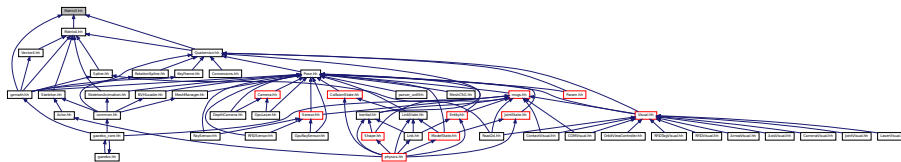
Forward declarations for the math classes.

## 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. 107) *3x3 matrix class.*

### Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

- namespace **gazebo::math**

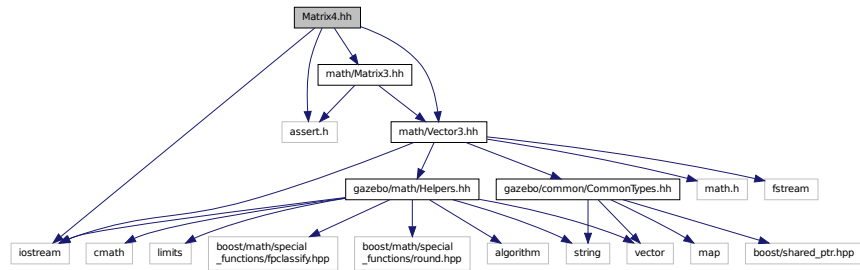
*Math namespace.*

## 11.80 Matrix4.hh File Reference

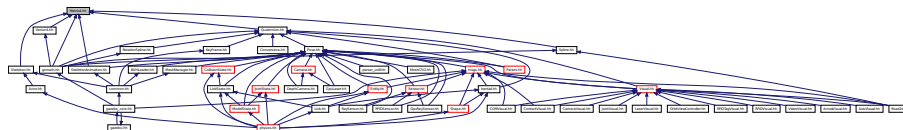
```
#include <assert.h>
#include <iostream>
#include "math/Vector3.hh"
#include "math/Matrix3.hh"
```



Include dependency graph for Matrix4.hh:



This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::math::Matrix4**

*A (p. 107) 3x3 matrix class.*

## Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

- namespace **gazebo::math**

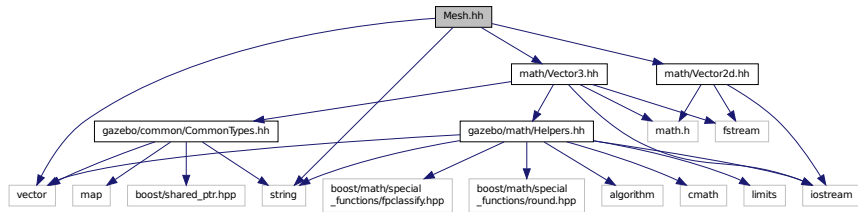
*Math namespace.*

## 11.81 Mesh.hh File Reference

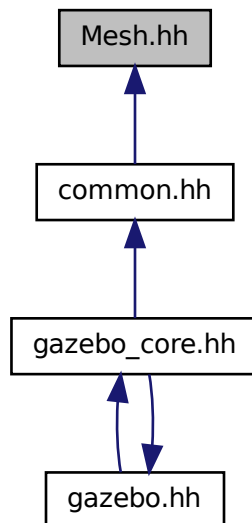
```

#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. 107) 3D mesh.
- struct **gazebo::common::NodeAssignment**  
Vertex to node weighted assignment for skeleton animation visualization.
- class **gazebo::common::SubMesh**  
A (p. 107) child mesh.

## Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

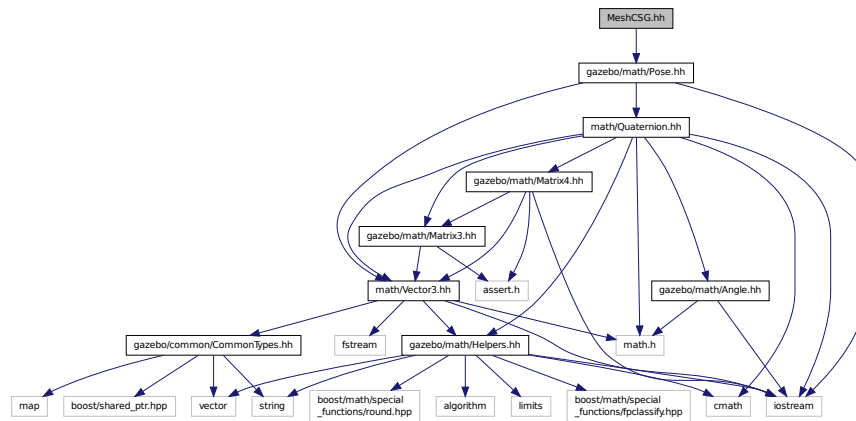
- namespace **gazebo::common**

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` **GPtrArray**
- typedef `_GtsSurface` **GtsSurface**

#### 11.82.1 Typedef Documentation

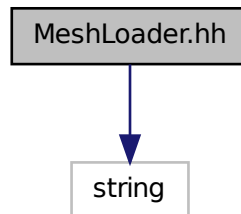
##### 11.82.1.1 typedef `_GPtrArray` **GPtrArray**

11.82.1.2 `typedef _GtsSurface GtsSurface`

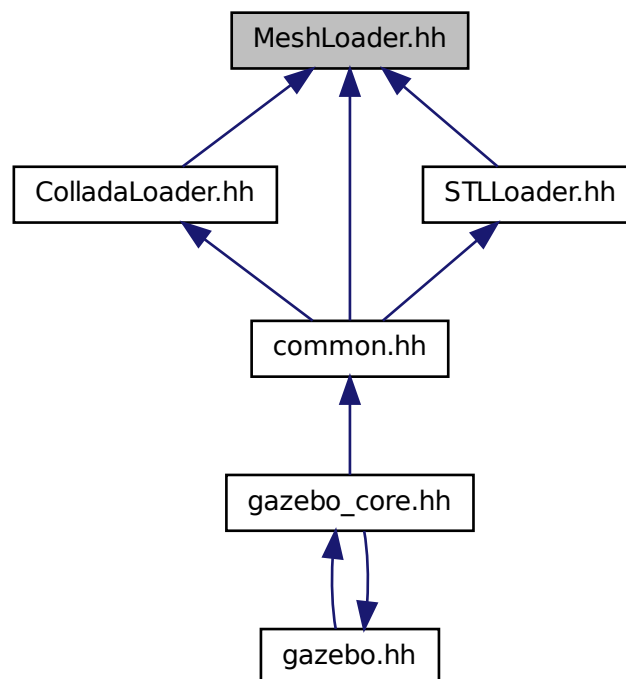
## 11.83 MeshLoader.hh File Reference

```
#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.*

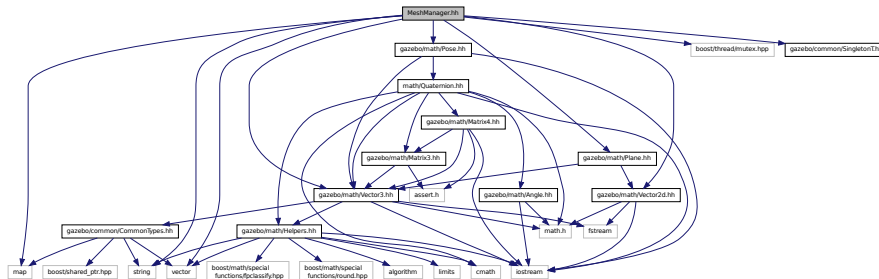
- namespace `gazebo::common`

*Common namespace.*

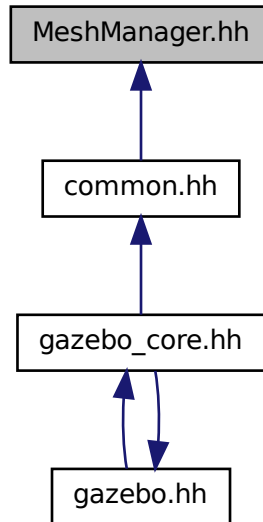
## 11.84 MeshManager.hh File Reference

```
#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.*

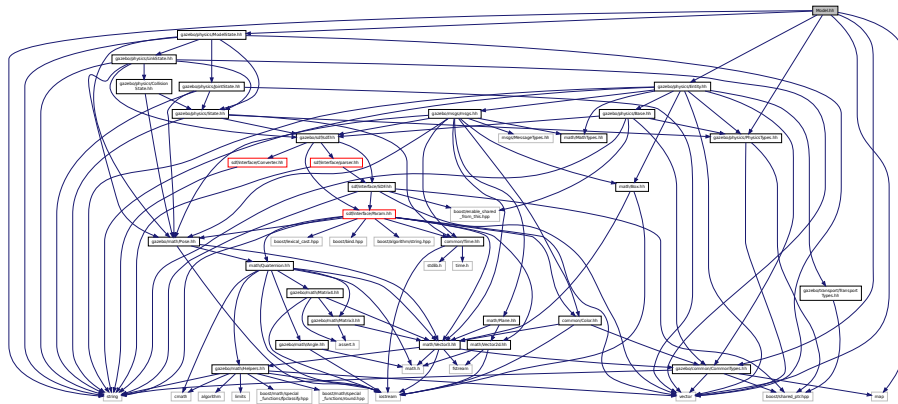
## 11.85 Model.hh File Reference

```

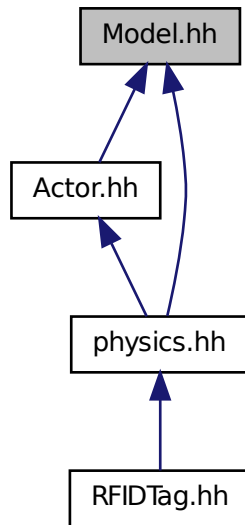
#include <string>
#include <map>
#include <vector>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/ModelState.hh"
#include "gazebo/physics/Entity.hh"

```

Include dependency graph for Model.hh:



This graph shows which files directly or indirectly include this file:



**Classes**

- class **gazebo::physics::Model**  
*A (p. 107) model is a collection of links, joints, and plugins.*

**Namespaces**

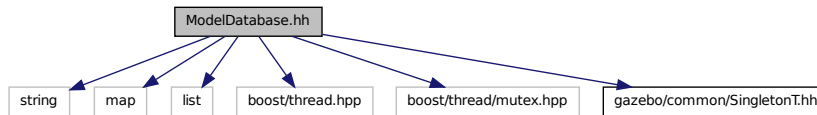
- namespace **boost**

- 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.



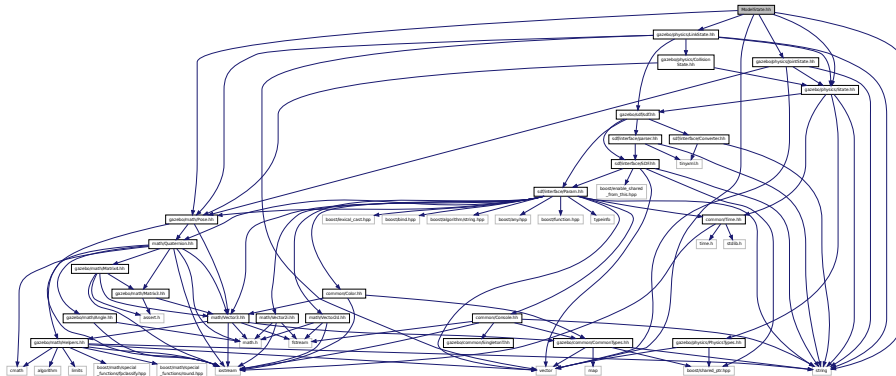
11.86.1.2 `#define GZ_MODEL_MANIFEST_FILENAME "model.config"`

The file name of model XML configuration.

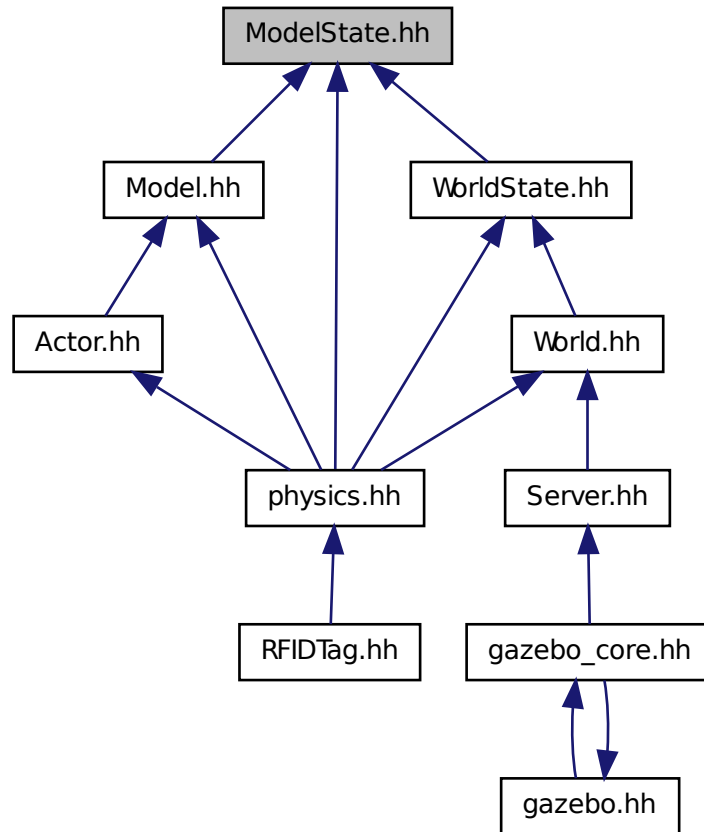
## 11.87 ModelState.hh File Reference

```
#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. 469) object.

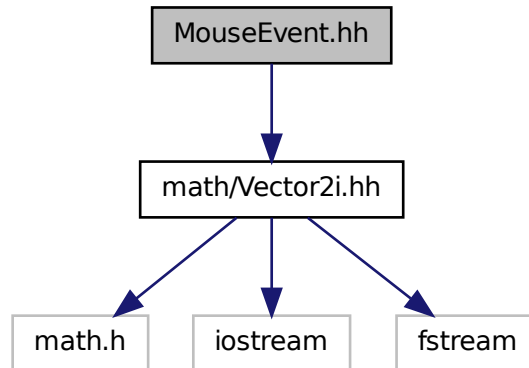
## Namespaces

- namespace **gazebo**  
Forward declarations for the common classes.
- namespace **gazebo::physics**  
namespace for physics

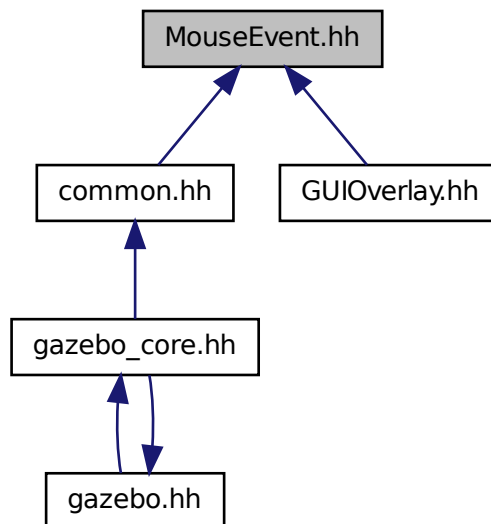
## 11.88 MouseEvent.hh File Reference

```
#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.*

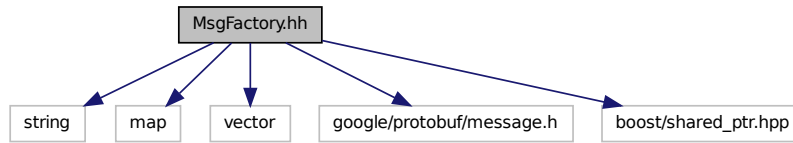
## Namespaces

- 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. 107) 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

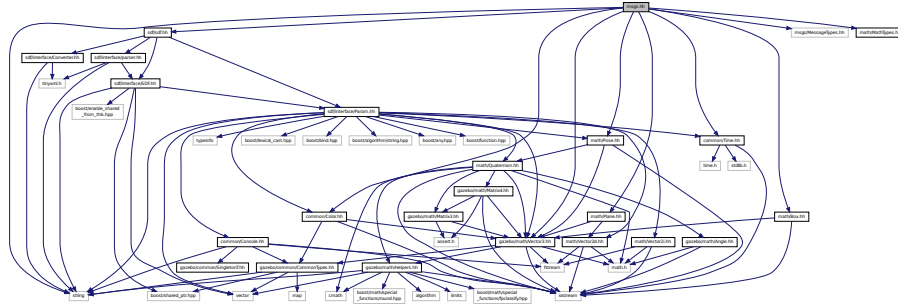
```
#include <string>
```

```

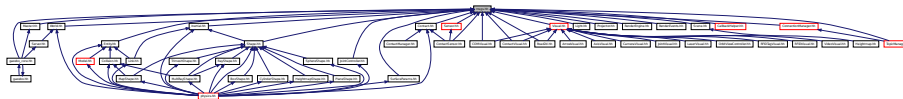
#include "msgs/MessageTypes.hh"
#include "sdf/sdf.hh"
#include "math/MathTypes.hh"
#include "math/Vector3.hh"
#include "math/Pose.hh"
#include "math/Plane.hh"
#include "math/Box.hh"
#include "common/Color.hh"
#include "common/Time.hh"

```

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. 821) to a msgs::Vector3d.*
- msgs::Quaternion **gazebo::msgs::Convert** (const math::Quaternion &\_q)  
*Convert a **math::Quaternion** (p. 598) to a msgs::Quaternion.*
- msgs::Pose **gazebo::msgs::Convert** (const math::Pose &\_p)  
*Convert a **math::Pose** (p. 573) to a msgs::Pose.*
- msgs::Color **gazebo::msgs::Convert** (const common::Color &\_c)  
*Convert a **common::Color** (p. 203) to a msgs::Color.*
- msgs::Time **gazebo::msgs::Convert** (const common::Time &\_t)  
*Convert a **common::Time** (p. 760) to a msgs::Time.*

- `msgs::PlaneGeom gazebo::msgs::Convert` (const `math::Plane &_p`)  
*Convert a **math::Plane** (p. 563) 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. 598).*
- `math::Pose gazebo::msgs::Convert` (const `msgs::Pose &_p`)  
*Convert a `msgs::Pose` to a **math::Pose** (p. 573).*
- `common::Color gazebo::msgs::Convert` (const `msgs::Color &_c`)  
*Convert a `msgs::Color` to a **common::Color** (p. 203).*
- `common::Time gazebo::msgs::Convert` (const `msgs::Time &_t`)  
*Convert a `msgs::Time` to a **common::Time** (p. 760).*
- `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. 352).*
- `void gazebo::msgs::Set` (`msgs::Image * _msg`, const `common::Image &_i`)  
*Set a `msgs::Image` from a **common::Image** (p. 352).*
- `void gazebo::msgs::Set` (`msgs::Vector3d * _pt`, const `math::Vector3 &_v`)  
*Set a `msgs::Vector3d` from a **math::Vector3** (p. 821).*
- `void gazebo::msgs::Set` (`msgs::Vector2d * _pt`, const `math::Vector2d &_v`)  
*Set a `msgs::Vector2d` from a **math::Vector3** (p. 821).*
- `void gazebo::msgs::Set` (`msgs::Quaternion * _q`, const `math::Quaternion &_v`)  
*Set a `msgs::Quaternion` from a **math::Quaternion** (p. 598).*
- `void gazebo::msgs::Set` (`msgs::Pose * _p`, const `math::Pose &_v`)  
*Set a `msgs::Pose` from a **math::Pose** (p. 573).*
- `void gazebo::msgs::Set` (`msgs::Color * _c`, const `common::Color &_v`)  
*Set a `msgs::Color` from a **common::Color** (p. 203).*
- `void gazebo::msgs::Set` (`msgs::Time * _t`, const `common::Time &_v`)  
*Set a `msgs::Time` from a **common::Time** (p. 760).*
- `void gazebo::msgs::Set` (`msgs::PlaneGeom * _p`, const `math::Plane &_v`)  
*Set a `msgs::PlaneGeom` from a **math::Plane** (p. 563).*
- `void gazebo::msgs::Stamp` (`msgs::Header * _header`)

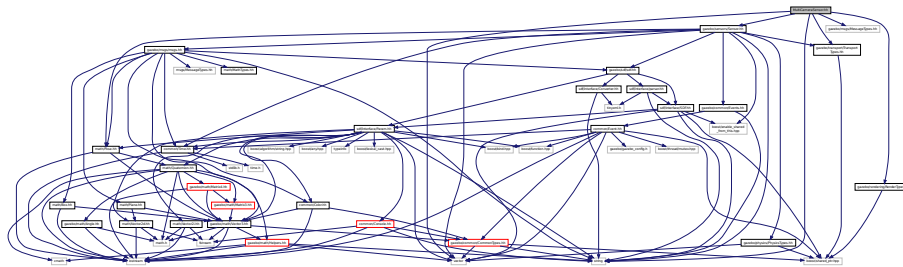
*Time stamp a header.*

- 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

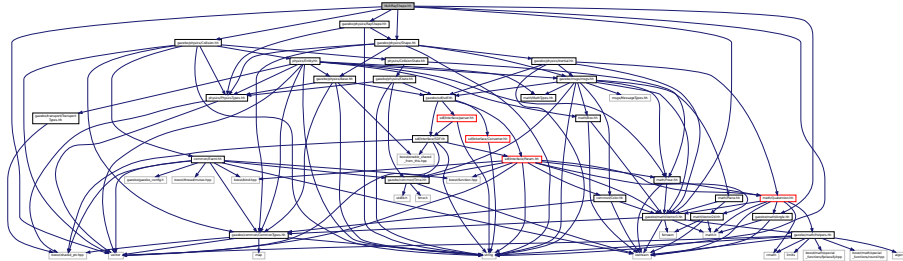
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::sensors**  
*Sensors namespace.*

## 11.93 MultiRayShape.hh File Reference

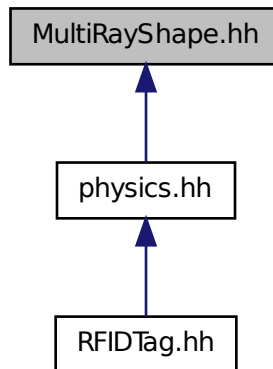
```
#include <vector>
```



```
#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

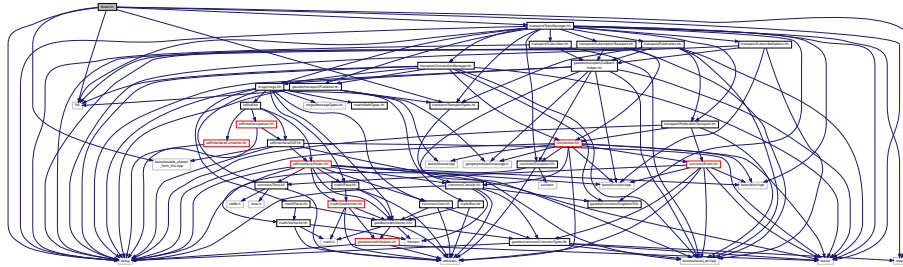
- class **gazebo::physics::MultiRayShape**  
*Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.*

## Namespaces

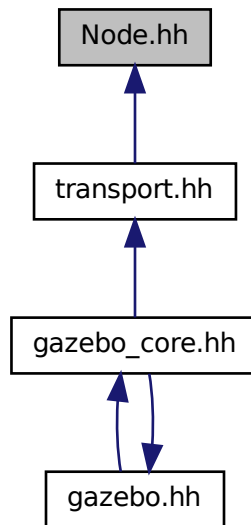
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*

## 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. 107) *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**

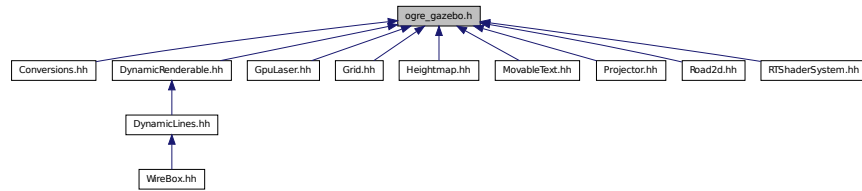
## 11.95 ogre\_gazebo.h File Reference

```
#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:



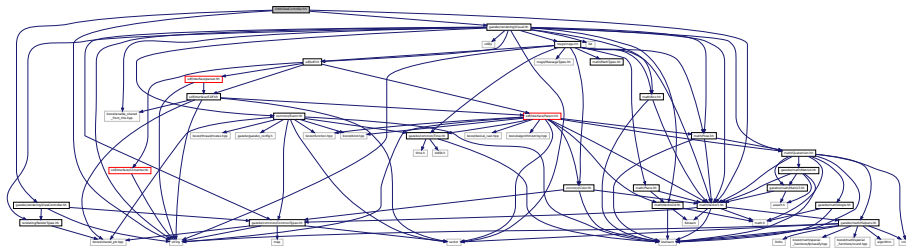
## 11.96 OrbitViewController.hh File Reference

```

#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>

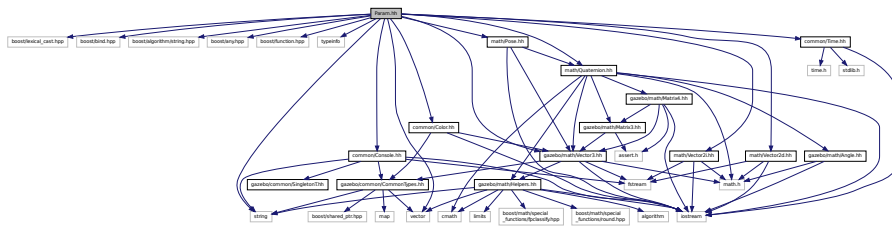
```

```

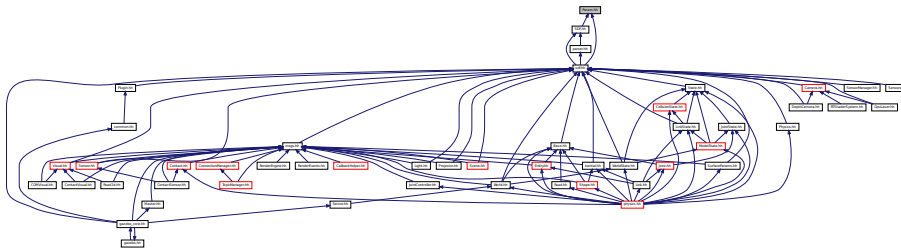
#include <boost/bind.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/any.hpp>
#include <boost/function.hpp>
#include <typeinfo>
#include <string>
#include <vector>
#include "common/Console.hh"
#include "common/Color.hh"
#include "common/Time.hh"
#include "math/Vector3.hh"
#include "math/Vector2i.hh"
#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. 107) parameter class.*
- class **sdf::ParamT< T >**  
*Templatized parameter class.*

## Namespaces

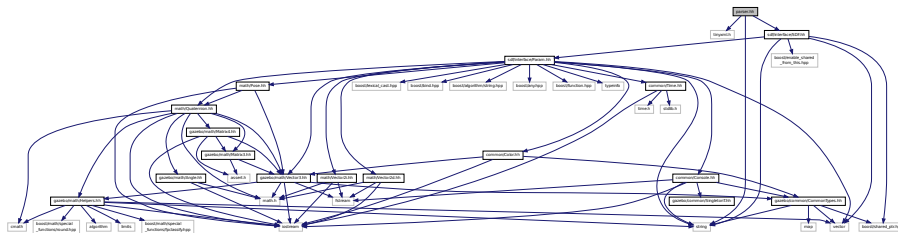
- namespace **sdf**  
*namespace for Simulation Description Format parser*

## Typedefs

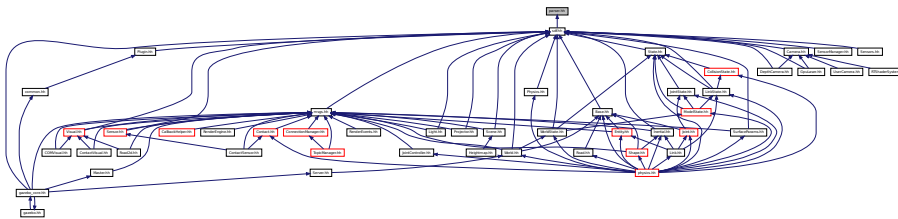
- typedef std::vector< ParamPtr > **sdf::Param\_V**
- typedef Param \* **sdf::ParamPtr**

## 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)



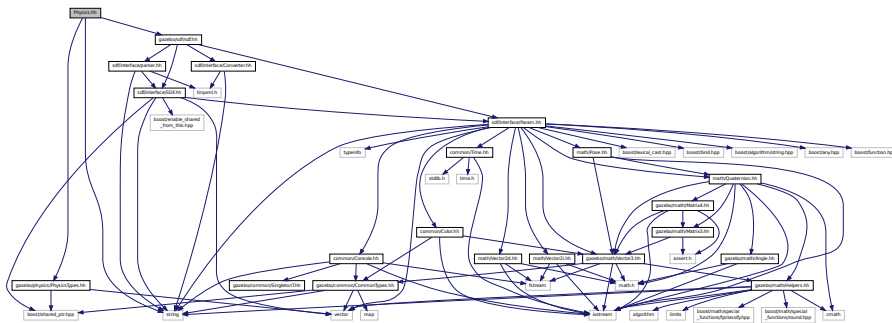
## Typedefs

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

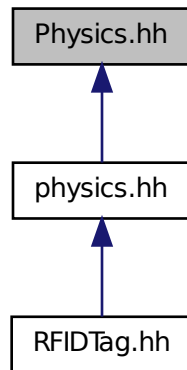
## 11.100 Physics.hh File Reference

```
#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*

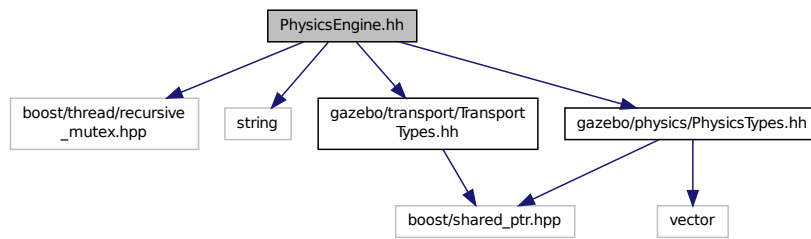
## Functions

- 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. 75).*
- 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. 759)'s and call **gazebo::transport::init** (p. 76).*
- void **gazebo::physics::load\_world** (WorldPtr \_world, sdf::ElementPtr \_sdf)  
*Load world from **sdf::Element** (p. 266) pointer.*
- void **gazebo::physics::load\_worlds** (sdf::ElementPtr \_sdf)  
*load multiple worlds from single **sdf::Element** (p. 266) pointer*
- void **gazebo::physics::pause\_world** (WorldPtr \_world, bool \_pause)  
*Pause world by calling **World::SetPaused** (p. 886).*
- 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. 885) 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. 886) given a pointer to it.*
- void **gazebo::physics::stop\_worlds** ()  
*stop multiple worlds stored in static variable gazebo::g\_worlds*

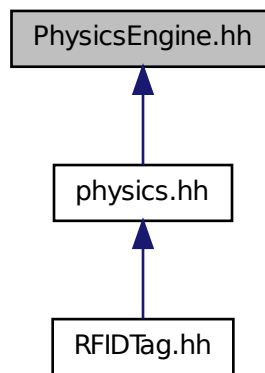
## 11.101 PhysicsEngine.hh File Reference

```
#include <boost/thread/recursive_mutex.hpp>
#include <string>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for PhysicsEngine.hh:



This graph shows which files directly or indirectly include this file:



## Classes

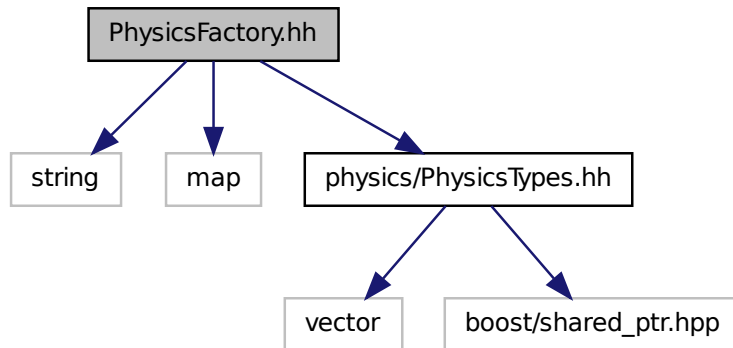
- class **gazebo::physics::PhysicsEngine**  
*Base (p. 133) class for a physics engine.*

## Namespaces

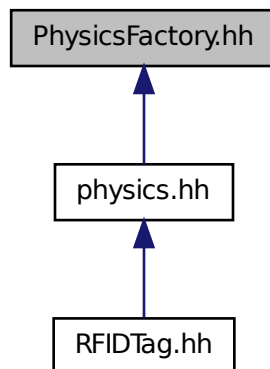
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*

## 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.*

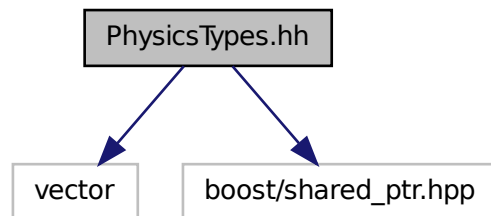
## Typedefs

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

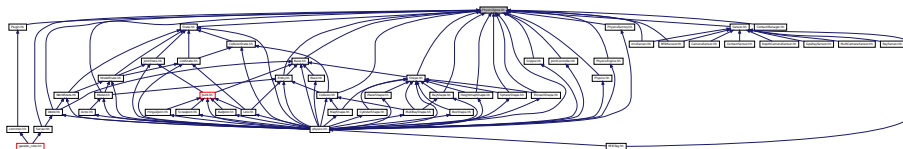
## 11.103 PhysicsTypes.hh File Reference

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** 0x00000003  
*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 0xFFFFFFFF`

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`

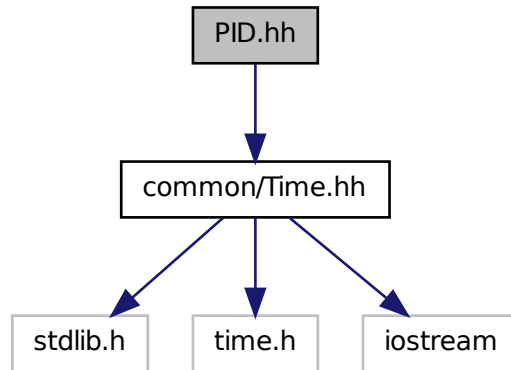
Collision object will collide with nothing.

#### 11.103.2.5 `#define GZ_SENSOR_COLLIDE 0x00000003`

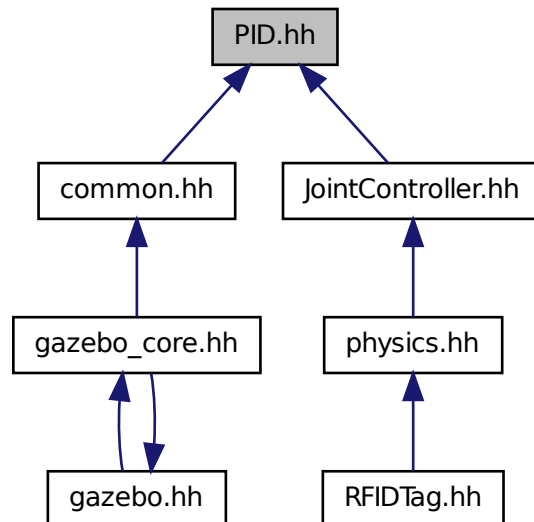
Collision object will collide only with sensors.

## 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

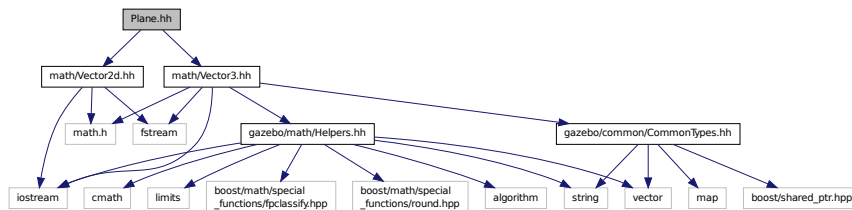
- class **gazebo::common::PID**  
*Generic PID (p. 559) controller class.*

## Namespaces

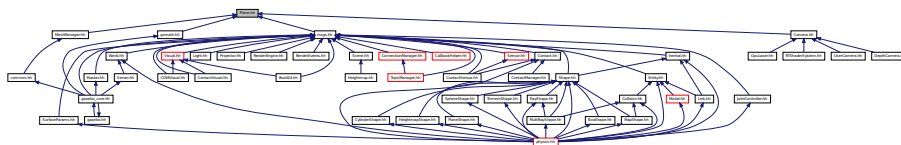
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::common**  
*Common namespace.*

### 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. 107) plane and related functions.*

## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::math**  
*Math namespace.*

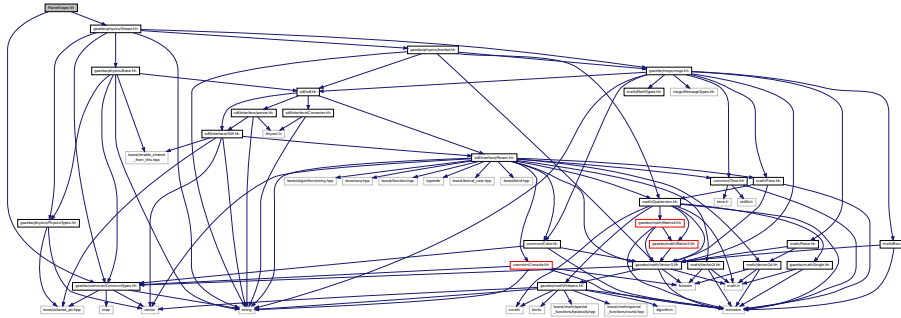


## 11.106 PlaneShape.hh File Reference

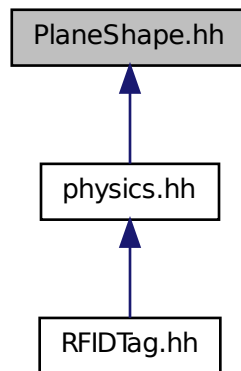
```
#include "gazebo/common/CommonTypes.hh"
```

```
#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. 190) for an infinite plane.

### Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*

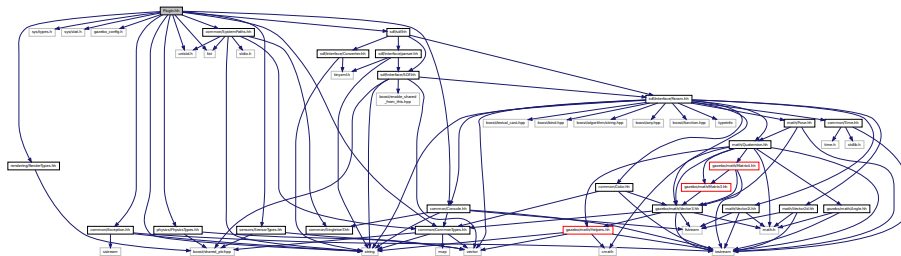
## 11.107 Plugin.hh File Reference

```

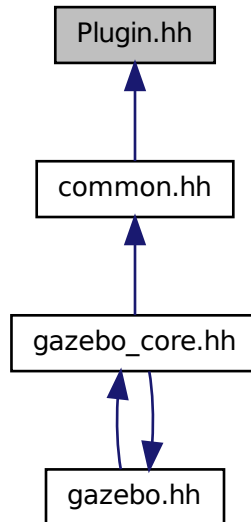
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <gazebo_config.h>
#include <list>
#include <string>
#include "common/CommonTypes.hh"
#include "common/SystemPaths.hh"
#include "common/Console.hh"
#include "common/Exception.hh"
#include "physics/PhysicsTypes.hh"
#include "sensors/SensorTypes.hh"
#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. 107) plugin with access to **physics::Model** (p. 469).*
- class **gazebo::PluginT< T >**  
*A (p. 107) class which all plugins must inherit from.*
- class **gazebo::SensorPlugin**  
*A (p. 107) plugin with access to **physics::Sensor**.*
- class **gazebo::SystemPlugin**  
*A (p. 107) plugin loaded within the gzserver on startup.*
- class **gazebo::VisualPlugin**  
*A (p. 107) plugin loaded within the gzserver on startup.*
- class **gazebo::WorldPlugin**  
*A (p. 107) plugin with access to **physics::World** (p. 875).*

## 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.

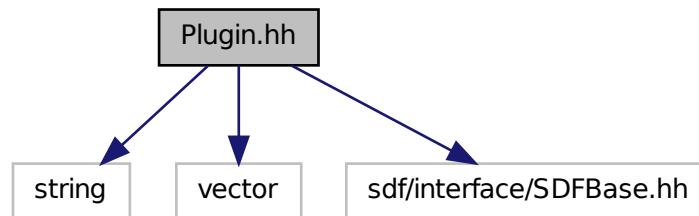
Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

**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

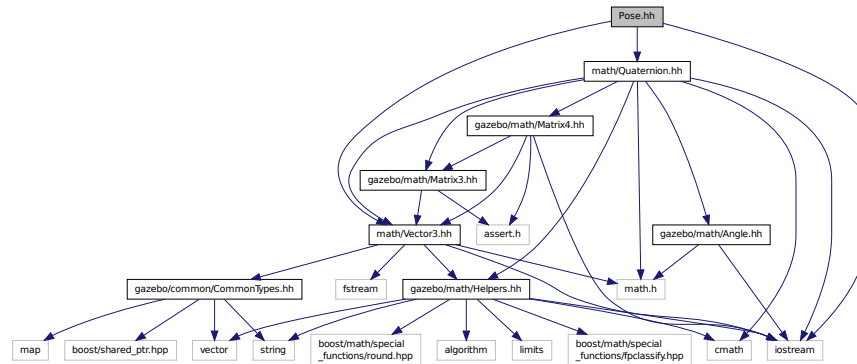
- namespace `sdf`

*namespace for Simulation Description Format parser*

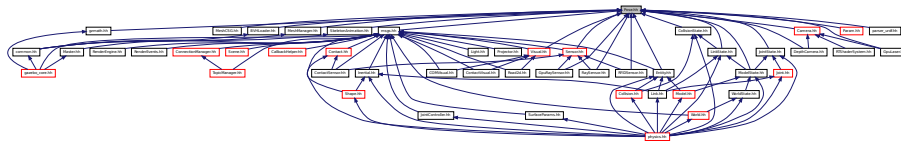
## 11.109 Pose.hh File Reference

```
#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.*

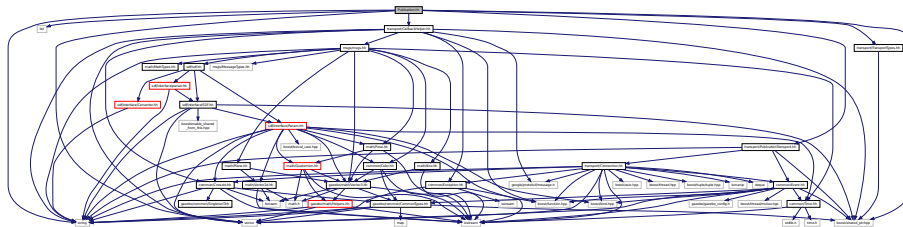
- namespace **gazebo::rendering**

*Rendering namespace.*

## 11.111 Publication.hh File Reference

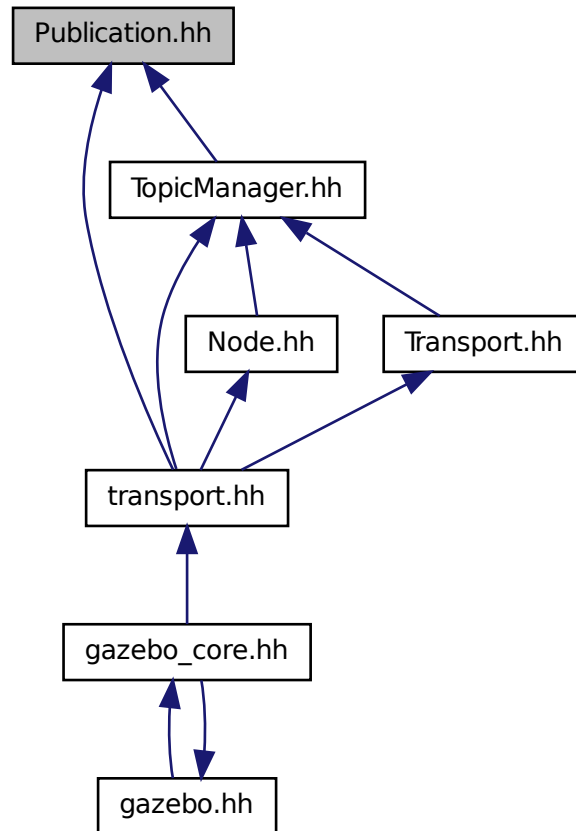
```
#include <boost/shared_ptr.hpp>
#include <list>
#include <string>
#include <vector>
#include "transport/CallbackHelper.hh"
#include "transport/TransportTypes.hh"
#include "transport/PublicationTransport.hh"
```

Include dependency graph for Publication.hh:





This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::transport::Publication**  
*A (p. 107) 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>
```



## Classes

- class **gazebo::transport::PublicationTransport**

*transport/transport.hh*

## Namespaces

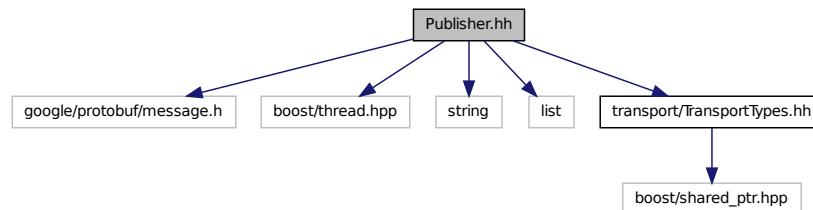
- namespace **gazebo**

*Forward declarations for the common classes.*

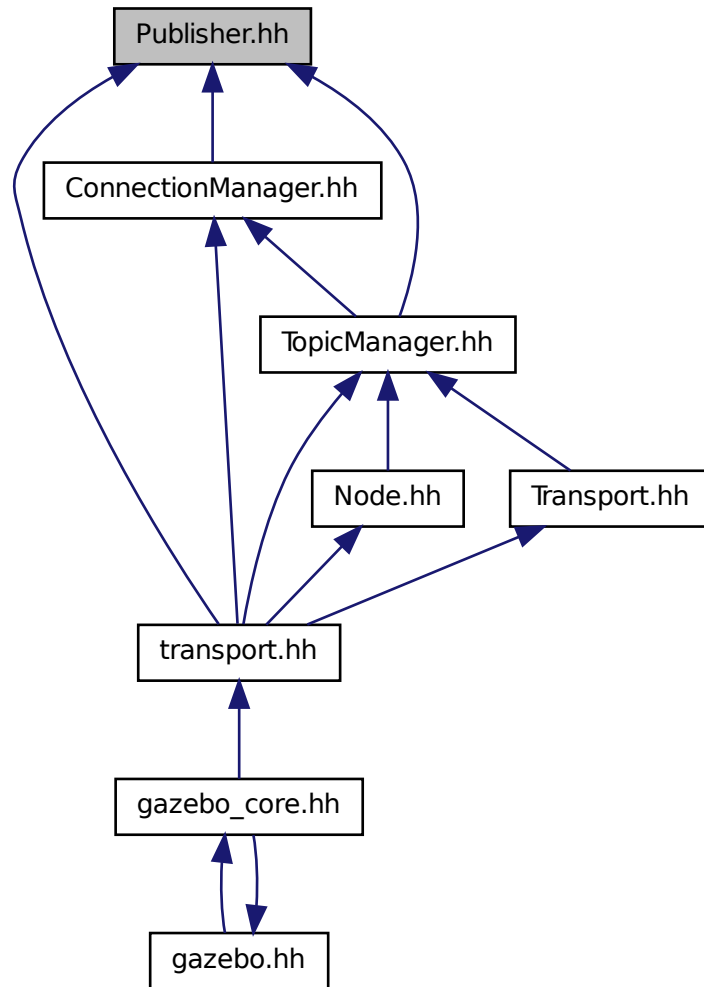
- namespace **gazebo::transport**

## 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. 107) publisher of messages on a topic.*

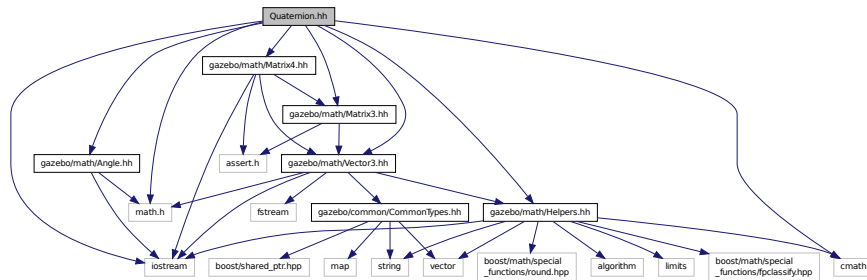
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::transport**

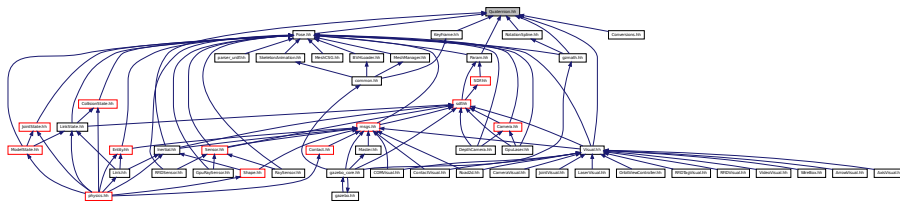
## 11.114 Quaternion.hh File Reference

```
#include <math.h>
#include <iostream>
#include <cmath>
#include "gazebo/math/Helpers.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Matrix3.hh"
#include "gazebo/math/Matrix4.hh"
```

Include dependency graph for Quaternion.hh:



This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::math::Quaternion**  
A (p. 107) quaternion class.

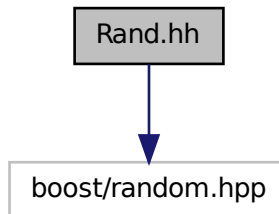
## Namespaces

- namespace **gazebo**  
Forward declarations for the common classes.
- namespace **gazebo::math**  
Math namespace.

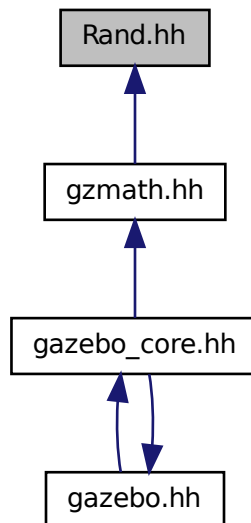
## 11.115 Rand.hh File Reference

```
#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.*

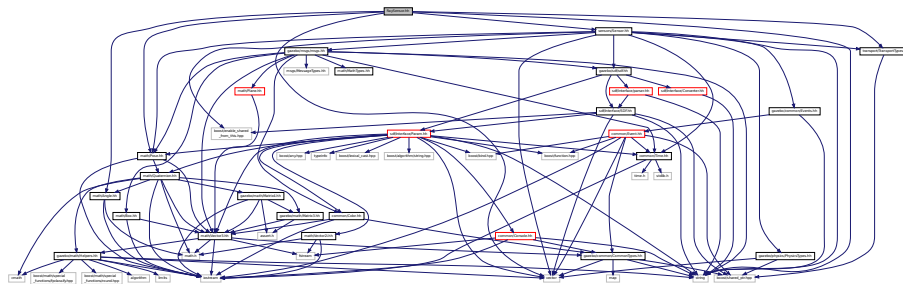
## Typedefs

- 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. 672) with one or more rays.*

## Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

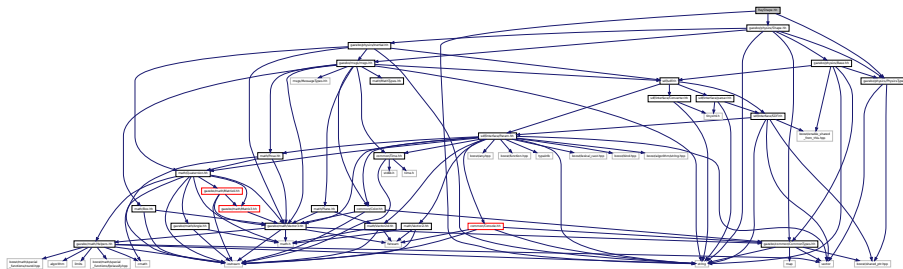
- namespace **gazebo::sensors**

*Sensors namespace.*

## 11.117 RayShape.hh File Reference

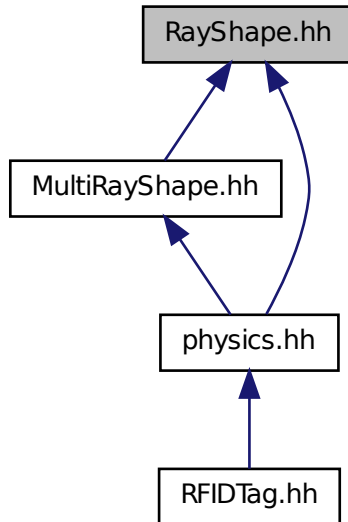
```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for RayShape.hh:





This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::physics::RayShape**  
*Base (p. 133) class for Ray collision geometry.*

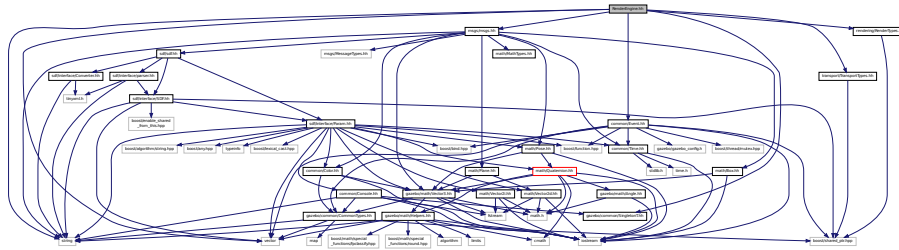
## Namespaces

- 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.*

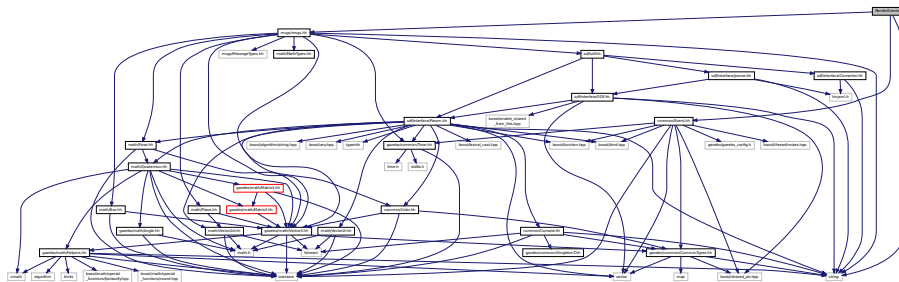
## Namespaces

- 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

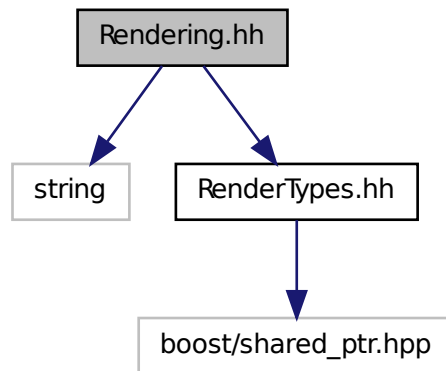
- class **gazebo::rendering::Events**  
*Base class for rendering events.*

## 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

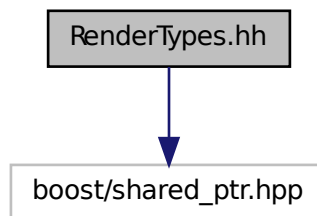
- rendering::ScenePtr **gazebo::rendering::create\_scene** (const std::string &\_name, bool \_enableVisualizations)  
*create **rendering::Scene** (p. 651) by name.*
- bool **gazebo::rendering::fini** ()  
*teardown rendering engine.*
- rendering::ScenePtr **gazebo::rendering::get\_scene** (const std::string &\_name)  
*get pointer to **rendering::Scene** (p. 651) 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. 651) 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.

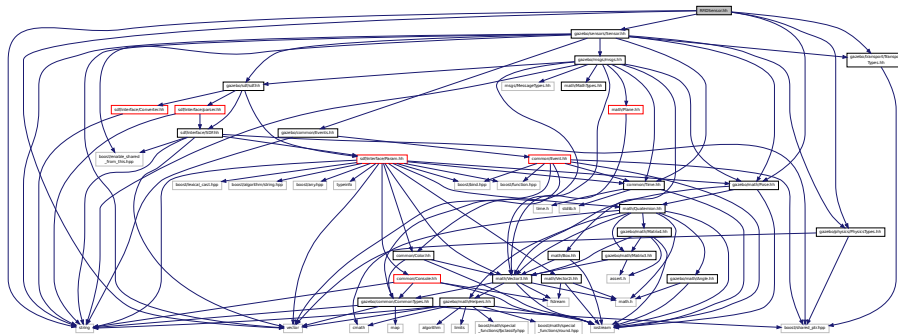
### 11.121.1.4 #define GZ\_VISIBILITY\_SELECTION 0x10000000

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. 672) class for RFID type of sensor.

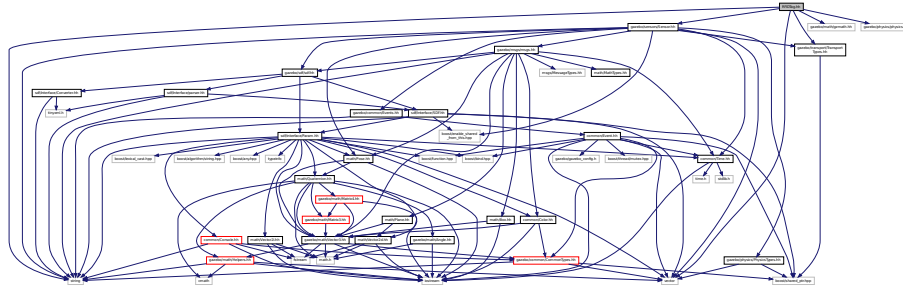
### Namespaces

- 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. 635) to interact with *RFIDTagSensors*.

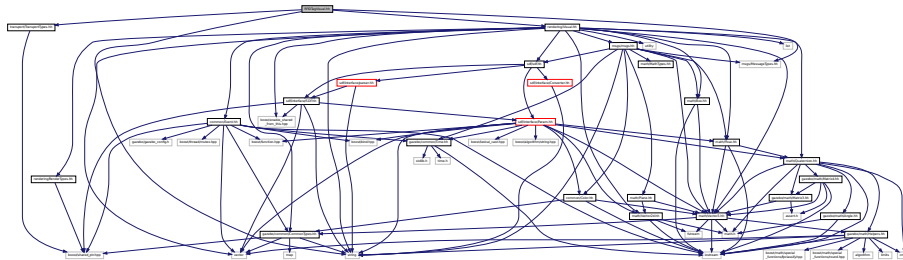
## Namespaces

- 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

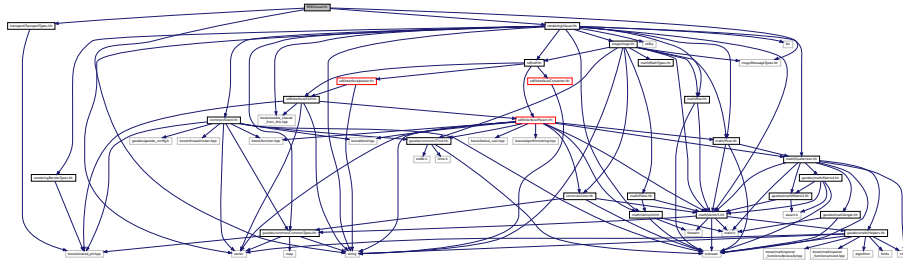
- class **gazebo::rendering::RFIDTagVisual**  
*Visualization for RFID tags sensor.*

## 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"
```





## 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**

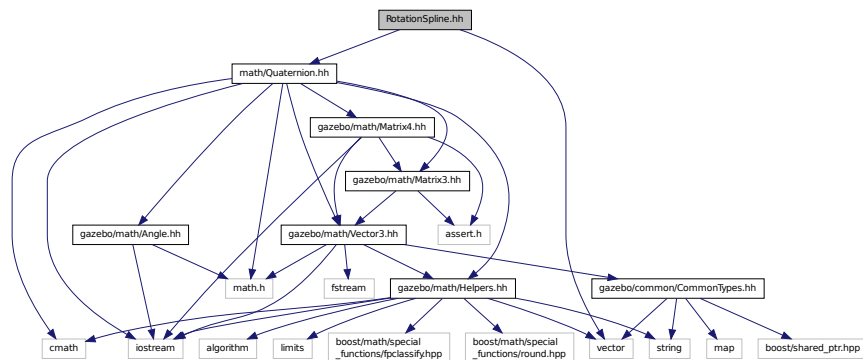
### Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::rendering**  
*Rendering namespace.*

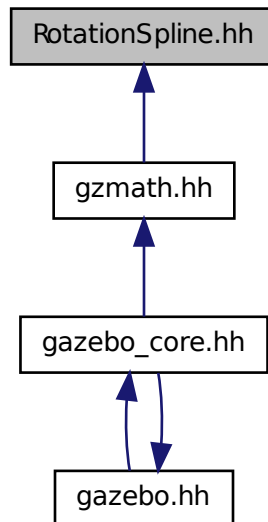
## 11.128 RotationSpline.hh File Reference

```
#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. 725) 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. 103)'s *Run-Time Shader* system.

## Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

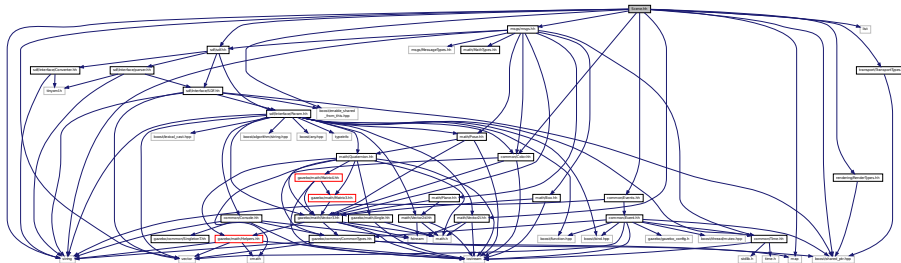
- namespace **gazebo::rendering**

Rendering namespace.

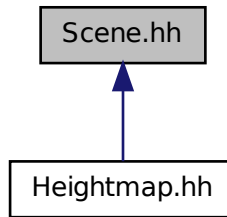
## 11.130 Scene.hh File Reference

```
#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.*

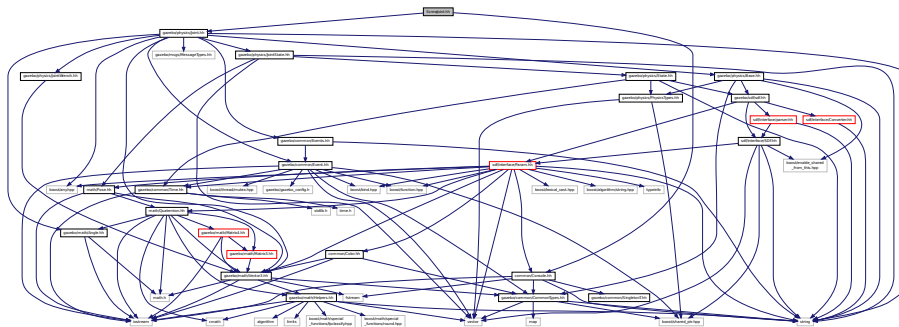
## Namespaces

- namespace **boost**
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::rendering**  
*Rendering namespace.*
- namespace **Ogre**
- namespace **SkyX**

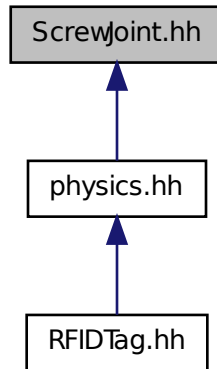
## 11.131 ScrewJoint.hh File Reference

```
#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. 107) screw joint, which has both prismatic and rotational DOFs.*

## Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

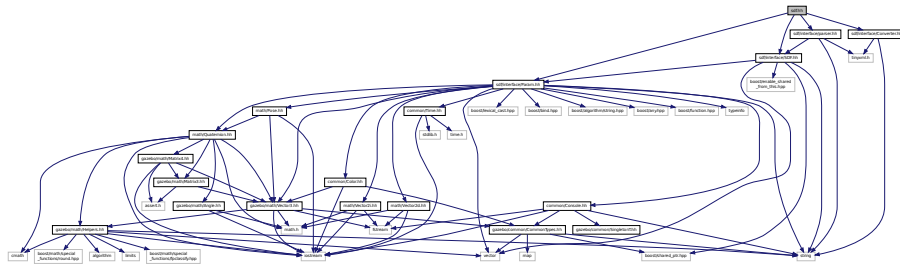
- namespace **gazebo::physics**

*namespace for physics*

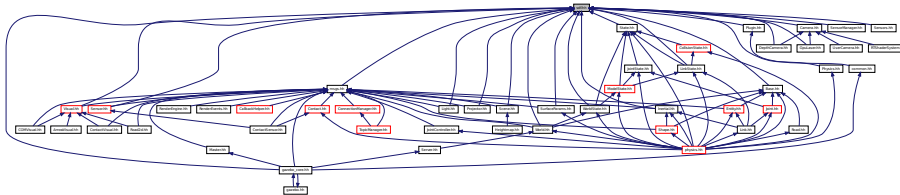
## 11.132 sdf.hh File Reference

```
#include "sdf/interface/SDF.hh"  
#include "sdf/interface/Param.hh"  
#include "sdf/interface/parser.hh"  
#include "sdf/interface/Converter.hh"
```

Include dependency graph for sdf.hh:



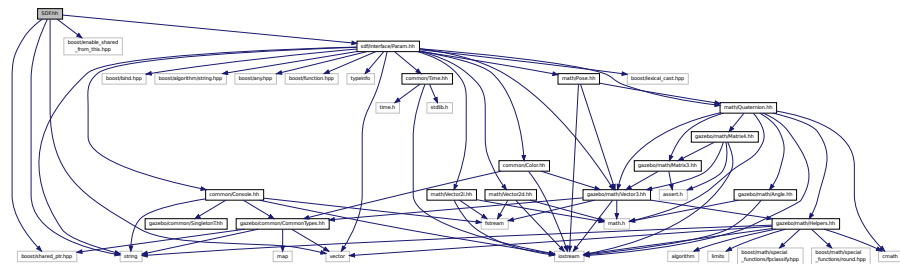
This graph shows which files directly or indirectly include this file:



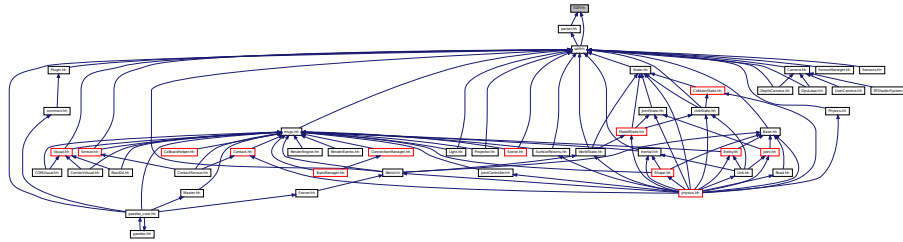
### 11.133 SDF.hh File Reference

```
#include <vector>
#include <string>
#include <boost/shared_ptr.hpp>
#include <boost/enable_shared_from_this.hpp>
#include "sdf/interface/Param.hh"
```

Include dependency graph for SDF.hh:



This graph shows which files directly or indirectly include this file:



## Classes

- class **sdf::Element**  
*SDF* (p. 669) *Element* (p. 266) class.
- class **sdf::SDF**  
*Base SDF* (p. 669) 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

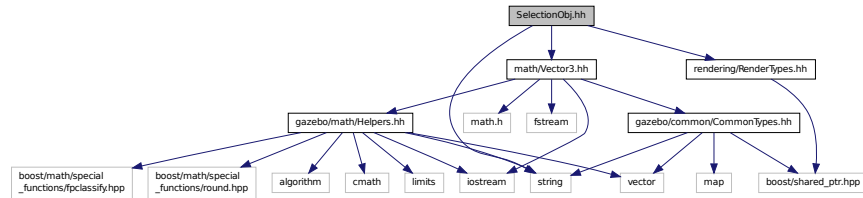
11.133.1.1 `#define SDF_VERSION "1.3"`

## 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. 107) graphical selection object.

## Namespaces

- namespace **gazebo**  
Forward declarations for the common classes.
- namespace **gazebo::rendering**  
Rendering namespace.

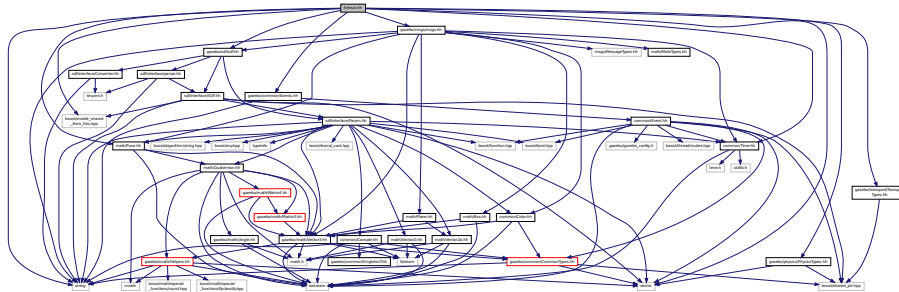
## 11.135 Sensor.hh File Reference

```

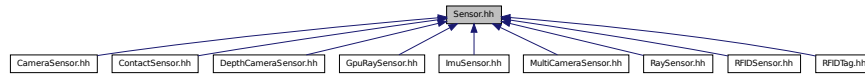
#include <boost/enable_shared_from_this.hpp>
#include <vector>
#include <string>
#include "gazebo/sdf/sdf.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/transport/TransportTypes.hh"

```

Include dependency graph for Sensor.hh:



This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::sensors::Sensor**

*Base class for sensors.*

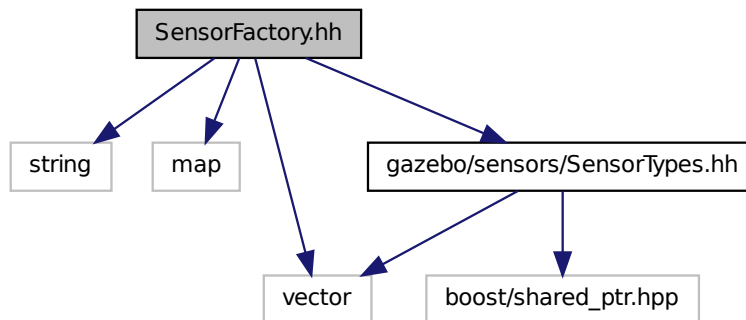
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::sensors**  
*Sensors namespace.*

## 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**



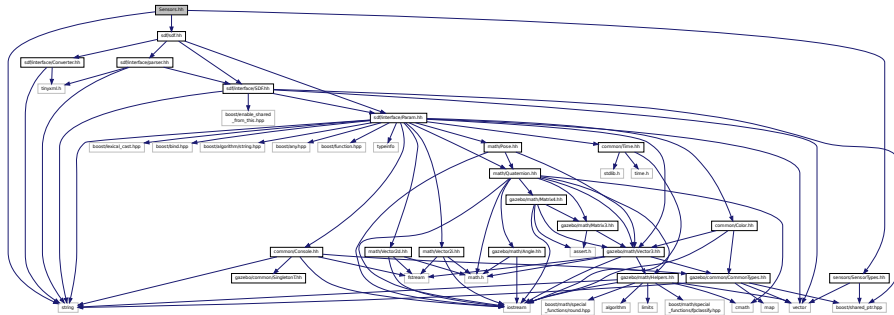
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::sensors**  
*Sensors namespace.*

## 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** ()

*Run sensor generation continuously. This is a blocking call.*

- void **gazebo::sensors::run\_once** (bool \_force=false)

*Run the sensor generation one step.*

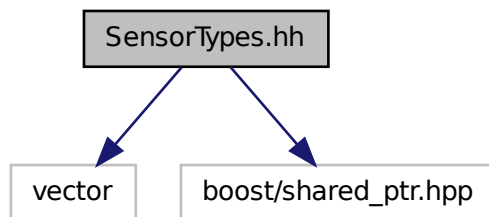
- void **gazebo::sensors::stop** ()

*Stop the sensor generation loop.*

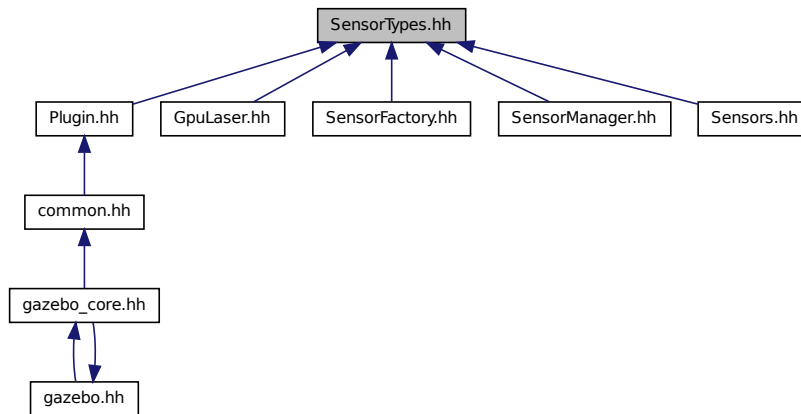
## 11.139 SensorTypes.hh File Reference

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< 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**

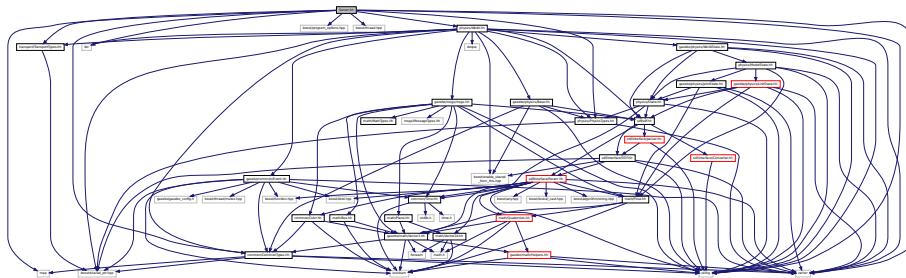
### 11.139.1 Detailed Description

Forward declarations and typedefs for sensors.

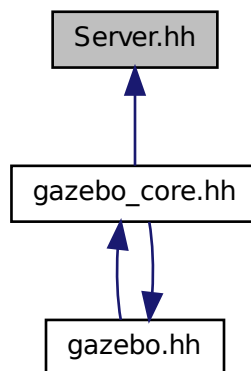
## 11.140 Server.hh File Reference

```
#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

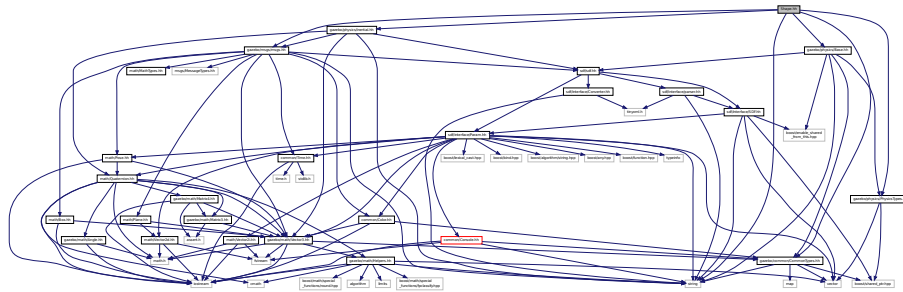
- namespace `boost`
- namespace `gazebo`

*Forward declarations for the common classes.*

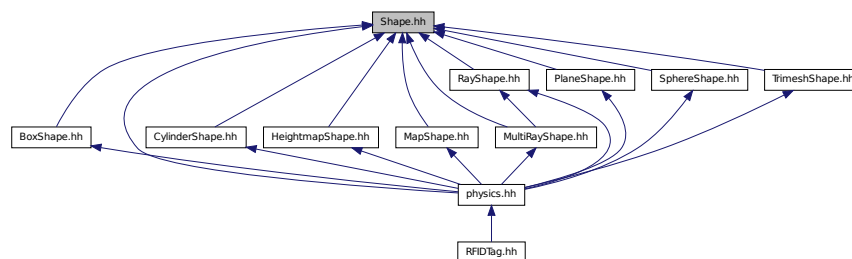
## 11.141 Shape.hh File Reference

```
#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. 133) class for all shapes.*



## Namespaces

- namespace **gazebo**

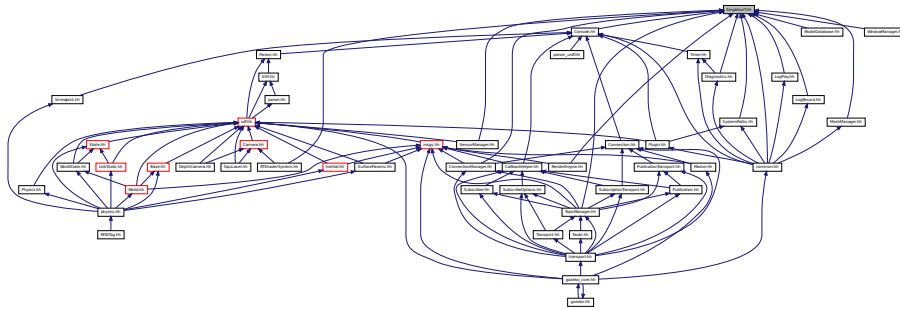
*Forward declarations for the common classes.*

- namespace **gazebo::physics**

*namespace for physics*

## 11.142 SingletonT.hh File Reference

This graph shows which files directly or indirectly include this file:



## Classes

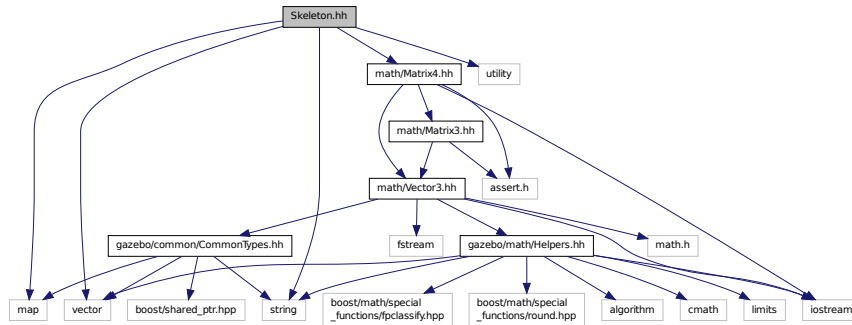
- class **SingletonT**< T >

*Singleton template class.*

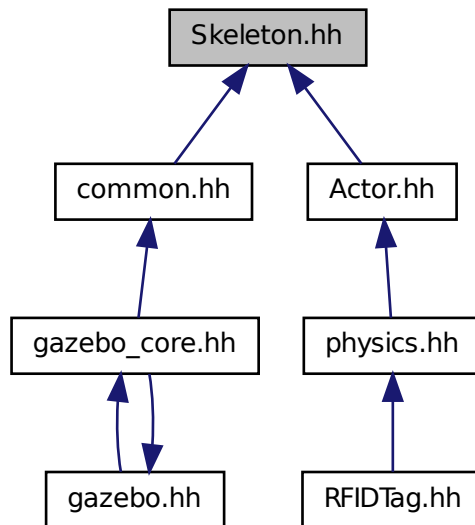
## 11.143 Skeleton.hh File Reference

```
#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. 526) *Skeleton.hh* (p. 1053) *common/common.hh*
- class **gazebo::common::Skeleton**  
*A* (p. 107) *skeleton*.
- class **gazebo::common::SkeletonNode**  
*A* (p. 107) *skeleton node*.

## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::common**  
*Common namespace.*

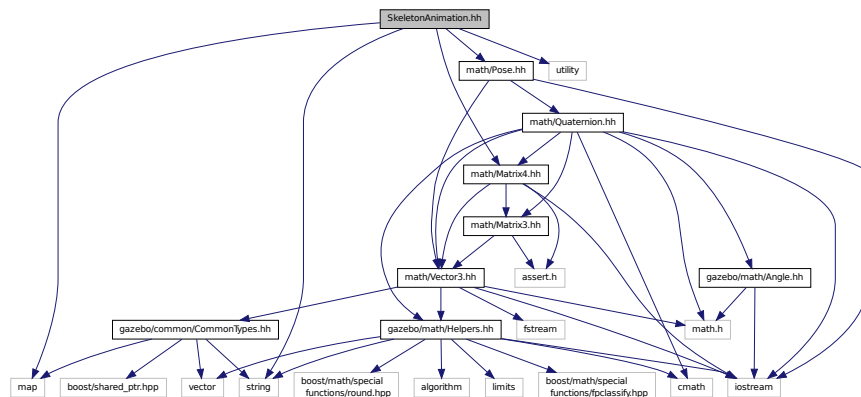
## Typedefs

- 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**

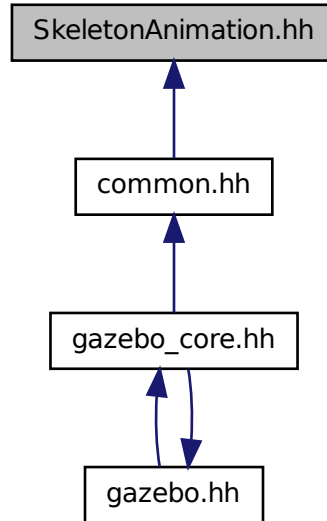
## 11.144 SkeletonAnimation.hh File Reference

```
#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. 698) animation.*

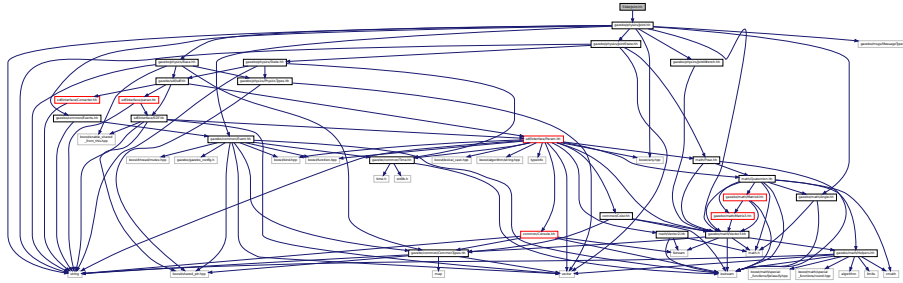
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::common**  
*Common namespace.*

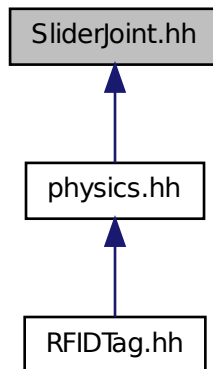
## 11.145 SliderJoint.hh File Reference

```
#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. 107) slider joint.*

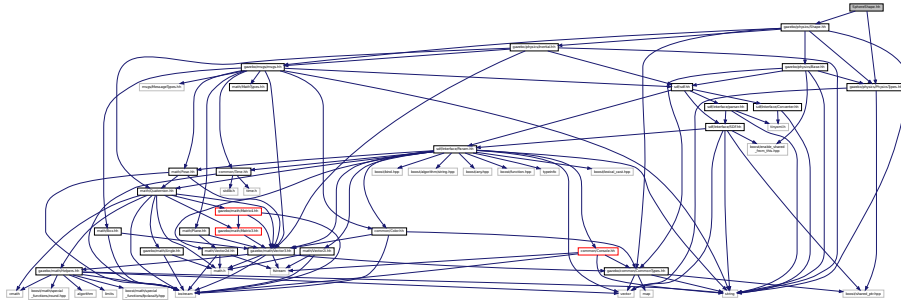
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*

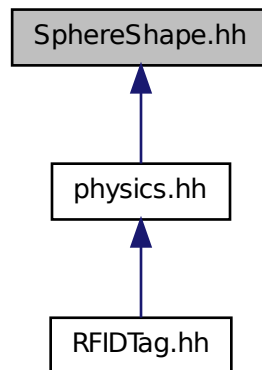
## 11.146 SphereShape.hh File Reference

```
#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

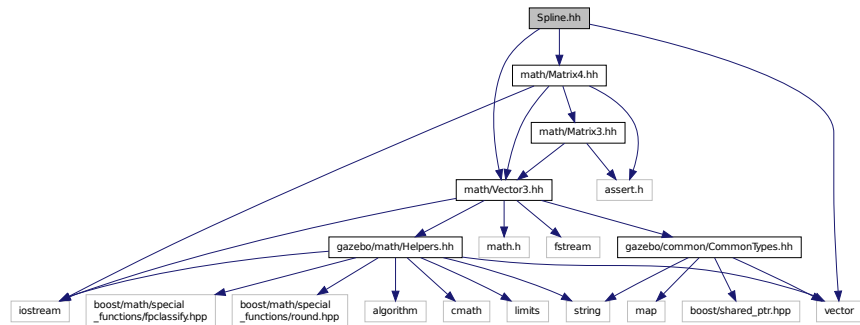
- class **gazebo::physics::SphereShape**  
*Sphere collision shape.*

### Namespaces

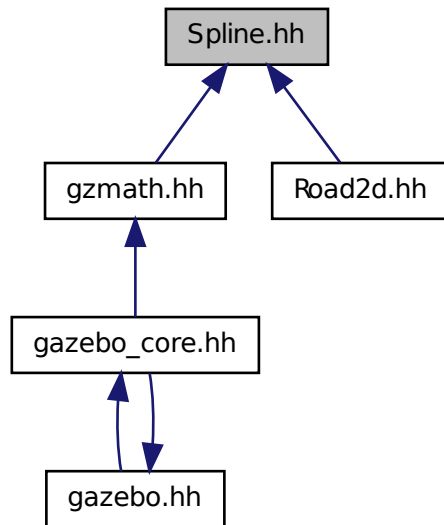
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*

## 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.*

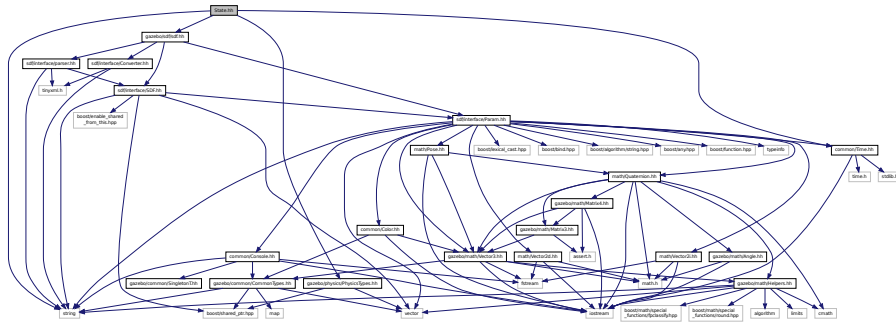
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::math**  
*Math namespace.*

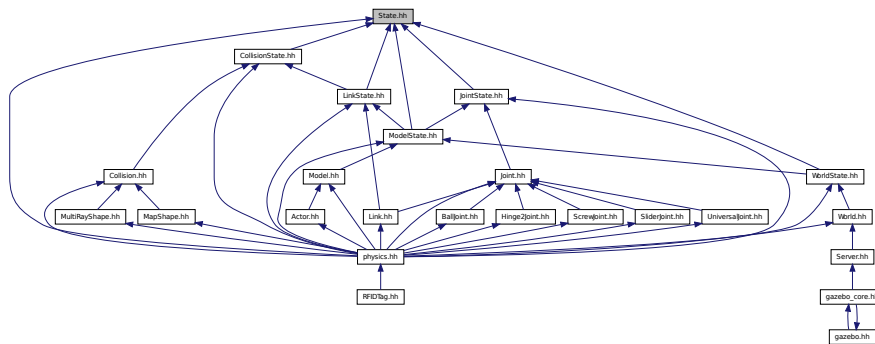
## 11.148 State.hh File Reference

```
#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. 729) of an entity.



## Namespaces

- namespace **gazebo**

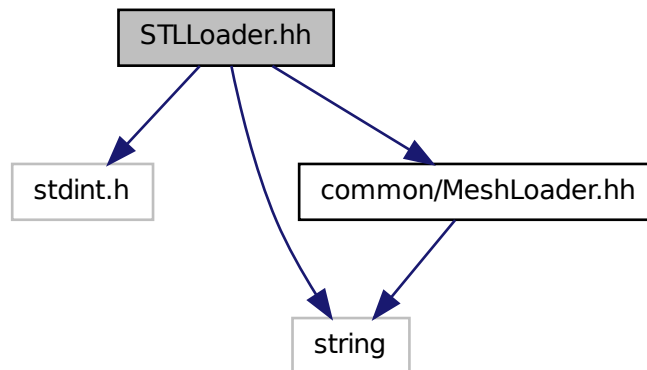
*Forward declarations for the common classes.*

- namespace **gazebo::physics**

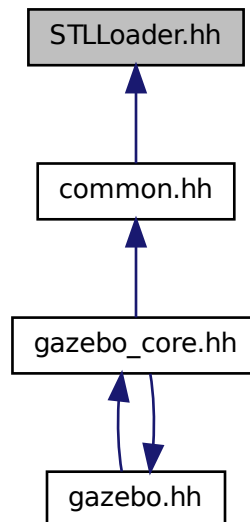
*namespace for physics*

## 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`

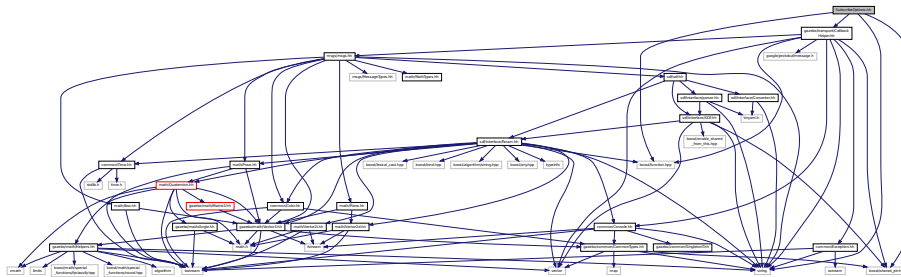
11.149.1.2 `#define FACE_MAX 200000`

11.149.1.3 `#define LINE_MAX_LEN 256`

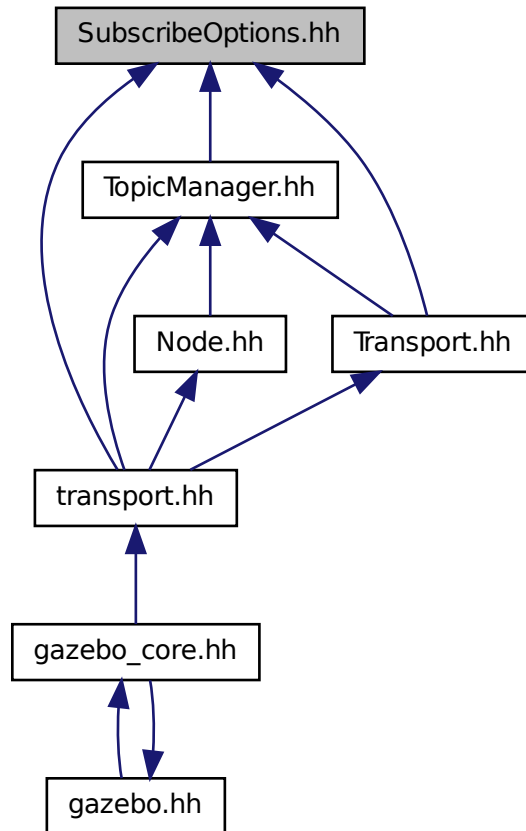
11.149.1.4 `#define ORDER_MAX 10`

## 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.*

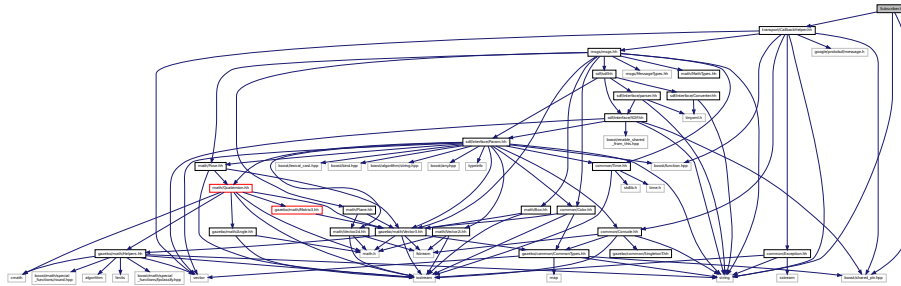
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::transport**

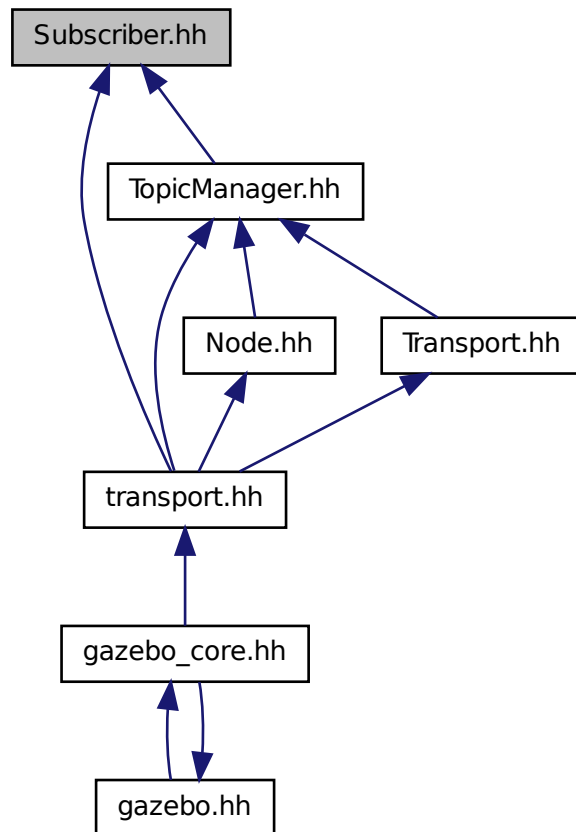
## 11.151 Subscriber.hh File Reference

```
#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. 107) subscriber to a topic.*

## Namespaces

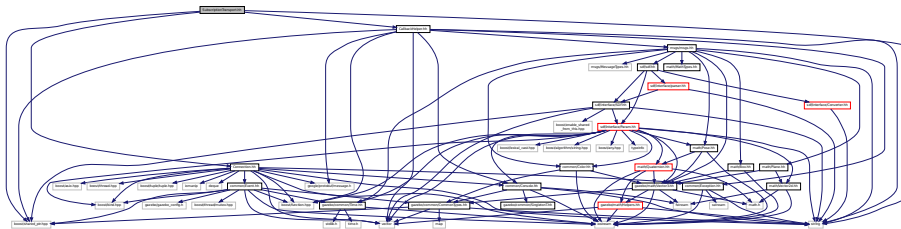
- namespace **gazebo**

*Forward declarations for the common classes.*

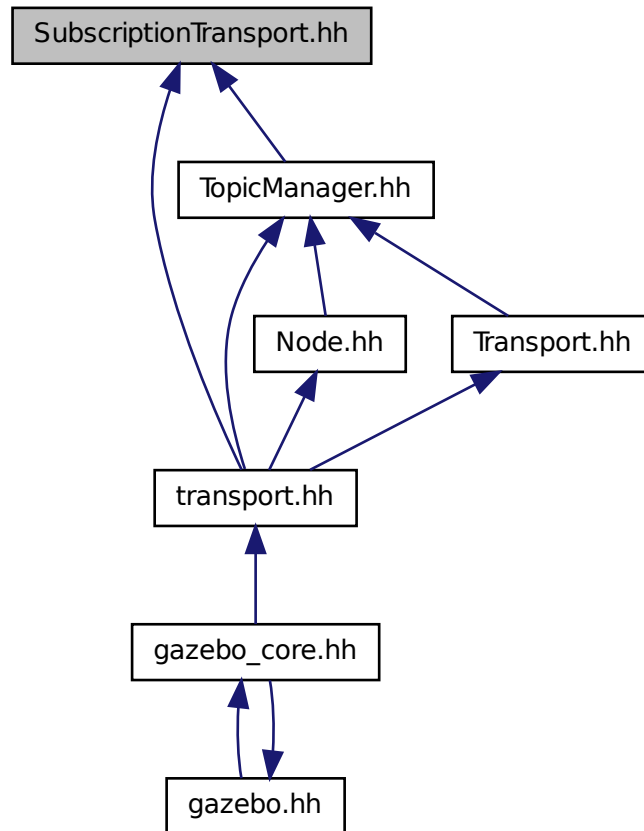
- namespace **gazebo::transport**

## 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*

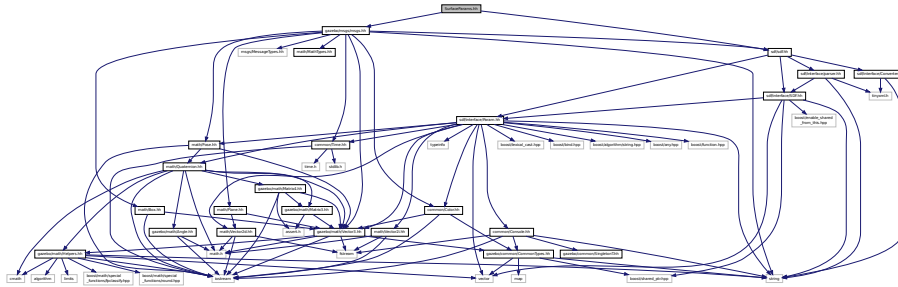
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::transport**

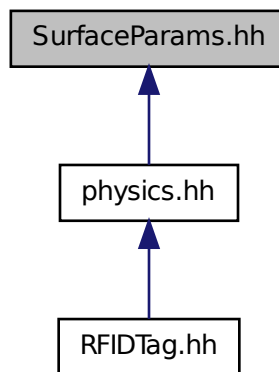
## 11.153 SurfaceParams.hh File Reference

```
#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. 750) defines various Surface contact parameters.

## Namespaces

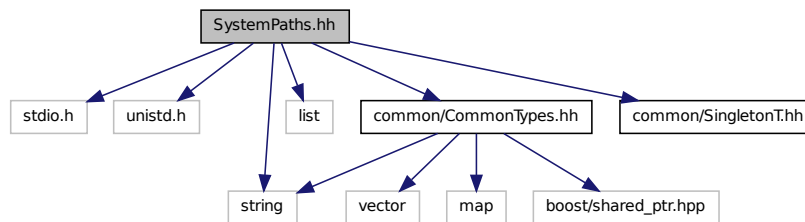
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*



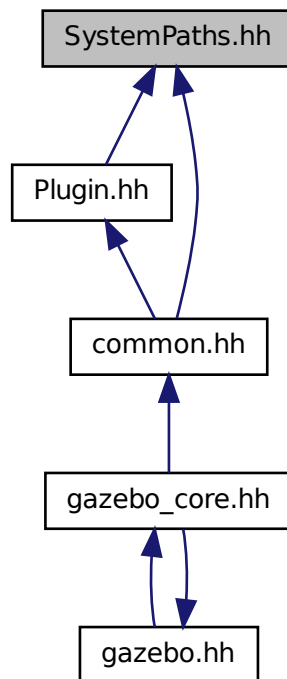
## 11.154 SystemPaths.hh File Reference

```
#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

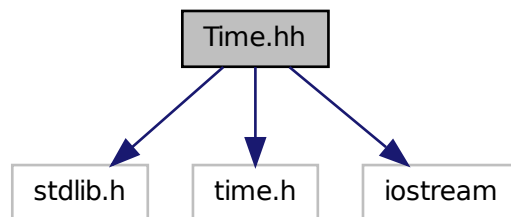
#### 11.154.1.1 #define GetCurrentDir getcwd

#### 11.154.1.2 #define LINUX

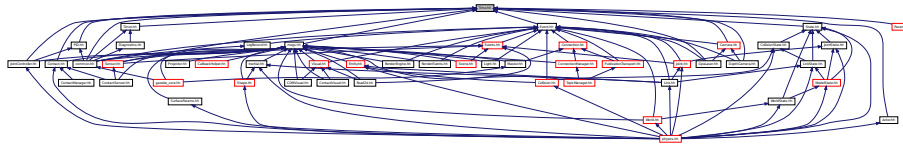
### 11.155 Time.hh File Reference

```
#include <stdlib.h>
#include <time.h>
#include <iostream>
```

Include dependency graph for Time.hh:



This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::common::Time**

**A** (p. 107) **Time** (p. 760) class, can be used to hold wall- or sim-time.

## Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

- namespace **gazebo::common**

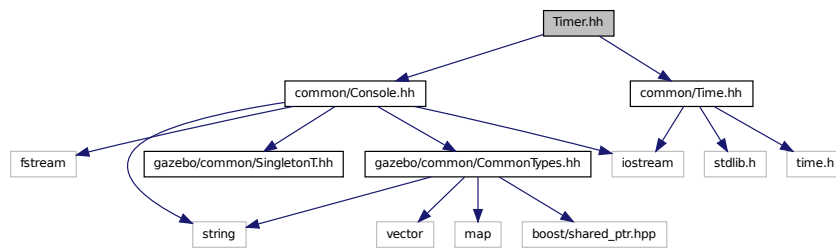
*Common namespace.*

## 11.156 Timer.hh File Reference

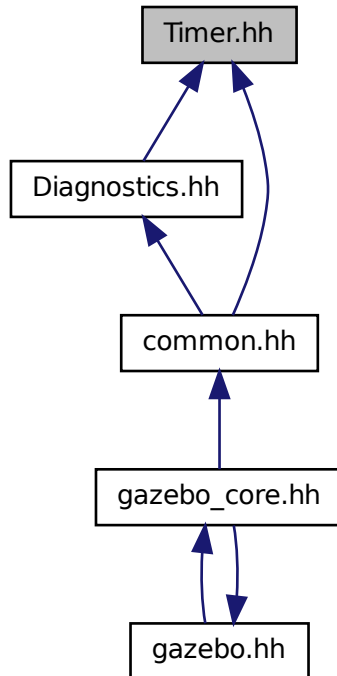
```
#include "common/Console.hh"
```

```
#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. 107) timer class, used to time things in real world walltime.*

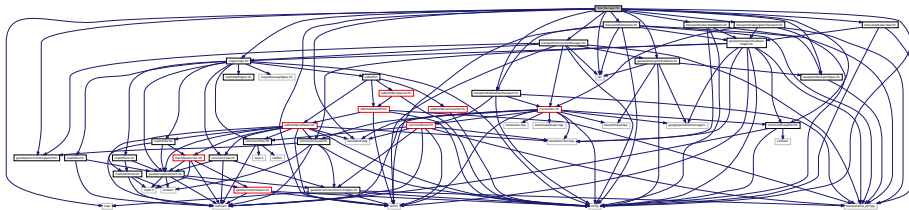
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::common**  
*Common namespace.*

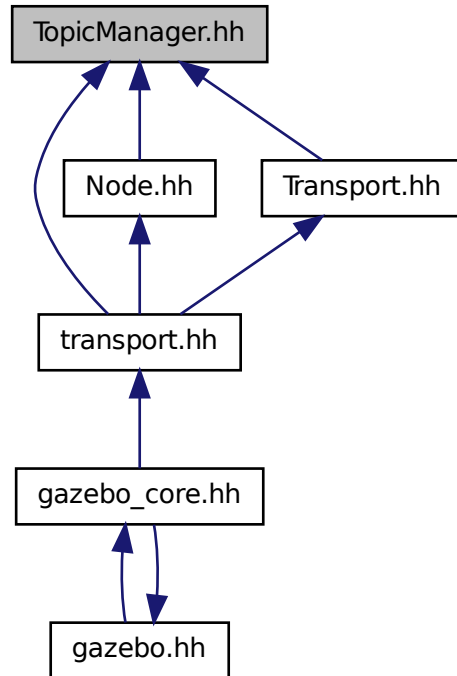
## 11.157 TopicManager.hh File Reference

```
#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.*

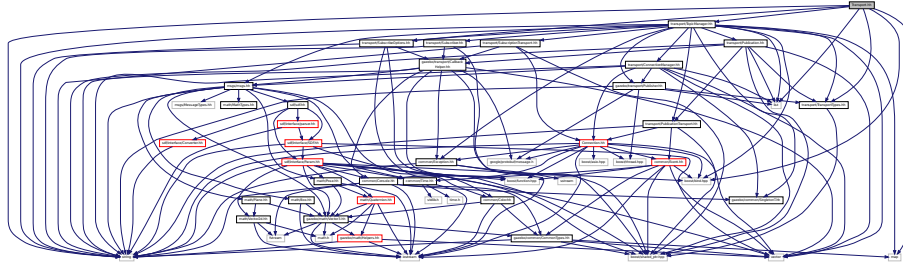
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::transport**

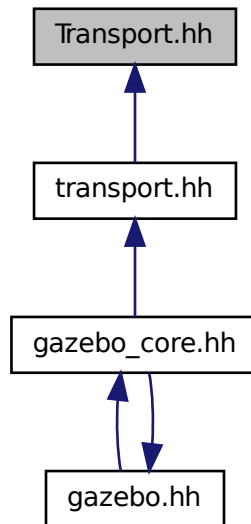
## 11.158 Transport.hh File Reference

```
#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.*

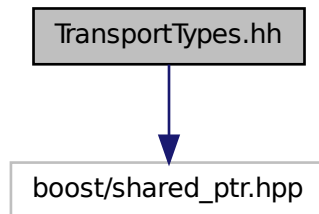
## 11.159 TransportTypes.hh File Reference

Forward declarations for transport.

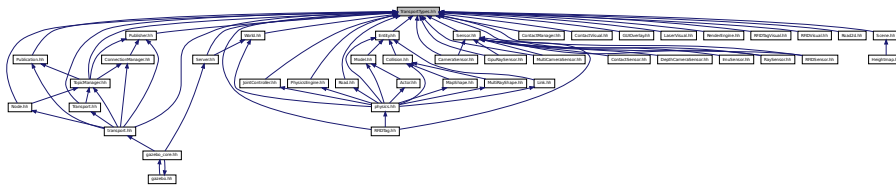


```
#include <boost/shared_ptr.hpp>
```

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**

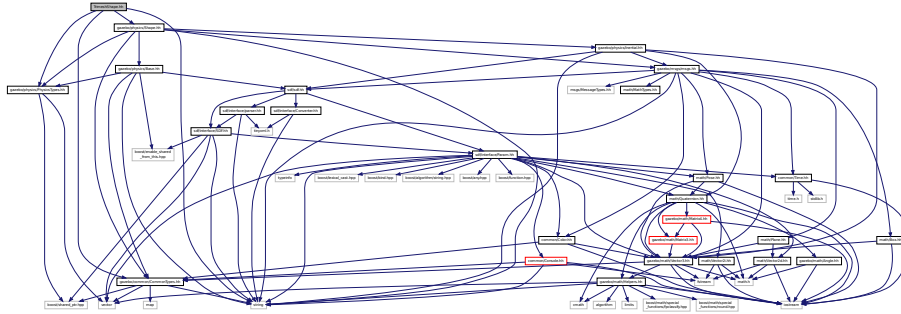
### 11.159.1 Detailed Description

Forward declarations for transport.

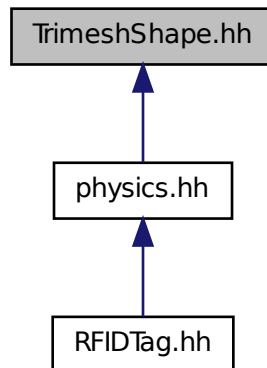
## 11.160 TrimeshShape.hh File Reference

```
#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

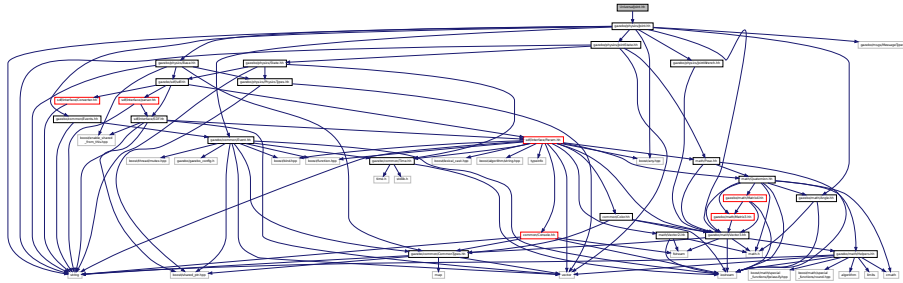
- namespace **gazebo**  
*Forward declarations for the common classes.*

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

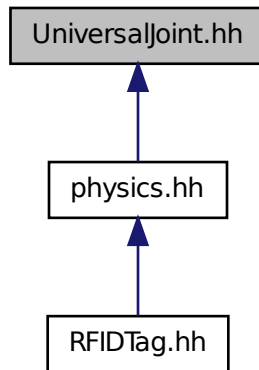
## 11.161 UniversalJoint.hh File Reference

```
#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. 107) universal joint.*

### Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

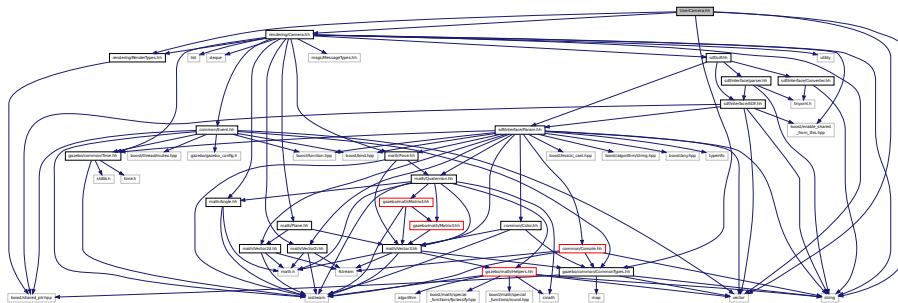
- namespace **gazebo::physics**

*namespace for physics*

## 11.162 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. 107) camera used for user visualization of a scene.*

### Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

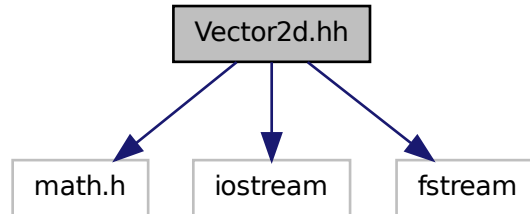
- namespace **gazebo::rendering**

*Rendering namespace.*

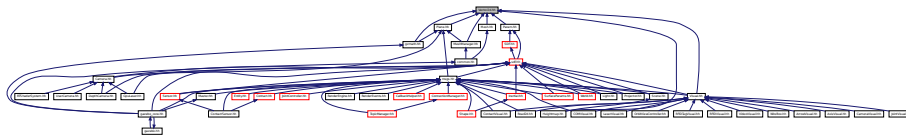
## 11.163 Vector2d.hh File Reference

```
#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**

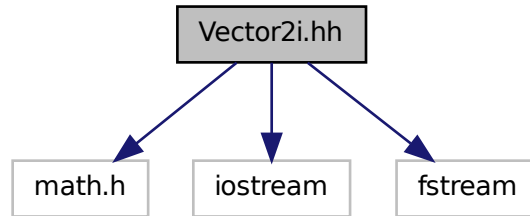
*Math namespace.*

## 11.164 Vector2i.hh File Reference

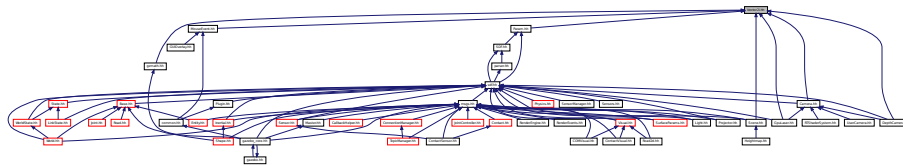
```

#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

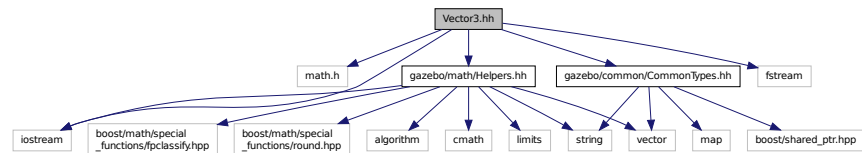
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::math**  
*Math namespace.*

## 11.165 Vector3.hh File Reference

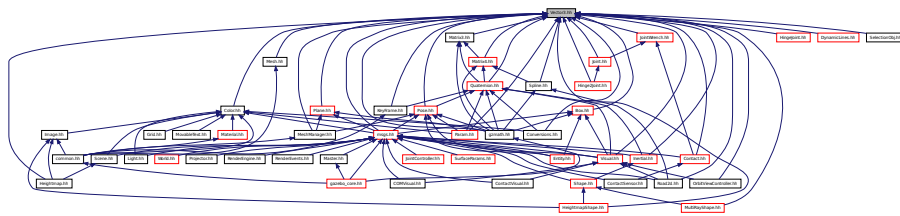
```

#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. 821) class represents the generic vector containing 3 elements.

## Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::math**

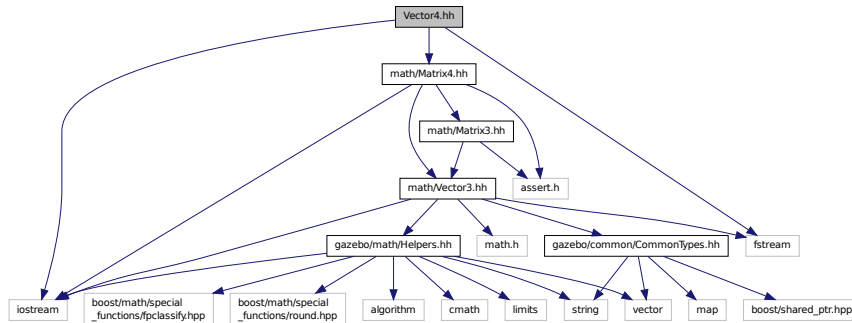
Math namespace.

## 11.166 Vector4.hh File Reference

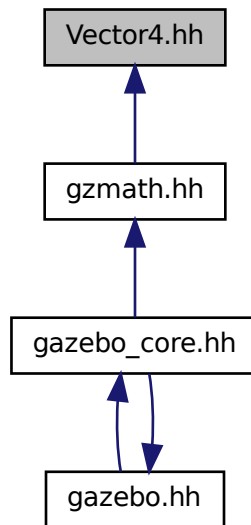
```

#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.*

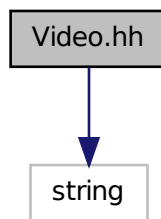
- namespace **gazebo::math**

*Math namespace.*

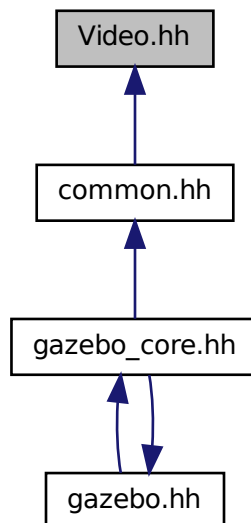
## 11.167 Video.hh File Reference

```
#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.*

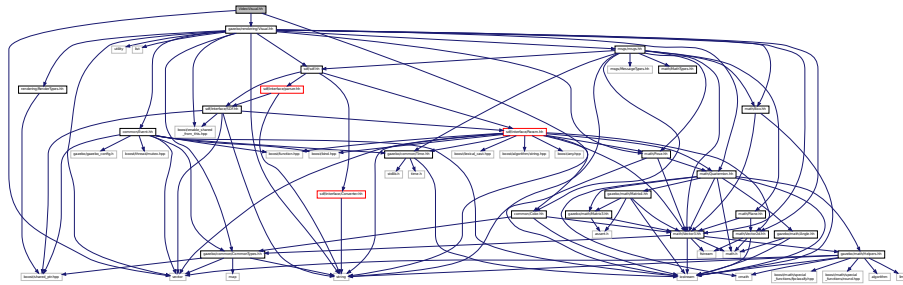
## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::common**  
*Common namespace.*

## 11.168 VideoVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for VideoVisual.hh:



## Classes

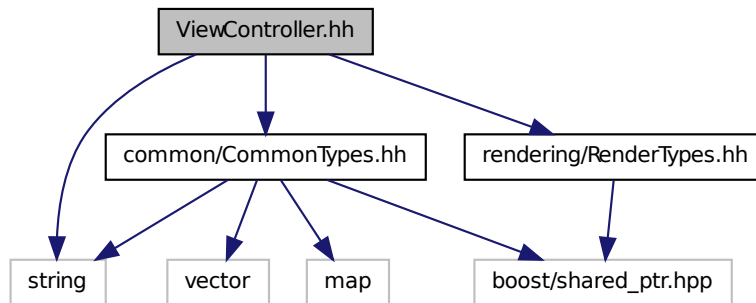
- class **gazebo::rendering::VideoVisual**  
*A (p. 107) visual element that displays a video as a texture.*

## Namespaces

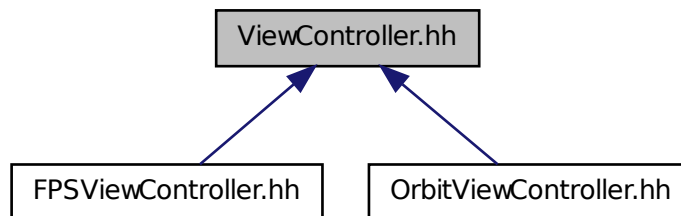
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::common**  
*Common namespace.*
- namespace **gazebo::rendering**  
*Rendering namespace.*

## 11.169 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

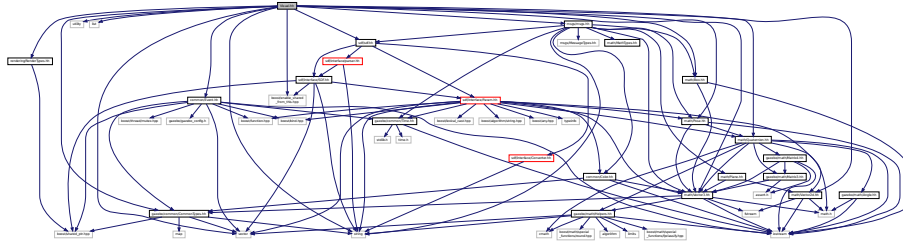
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::rendering**

*Rendering namespace.*

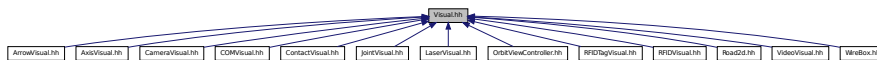
## 11.170 Visual.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include <utility>
#include <list>
#include <vector>
#include "common/Event.hh"
#include "math/Box.hh"
#include "math/Pose.hh"
#include "math/Quaternion.hh"
#include "math/Vector3.hh"
#include "math/Vector2d.hh"
#include "sdf/sdf.hh"
#include "msgs/msgs.hh"
#include "rendering/RenderTypes.hh"
#include "common/CommonTypes.hh"
```

Include dependency graph for Visual.hh:



This graph shows which files directly or indirectly include this file:



## Classes

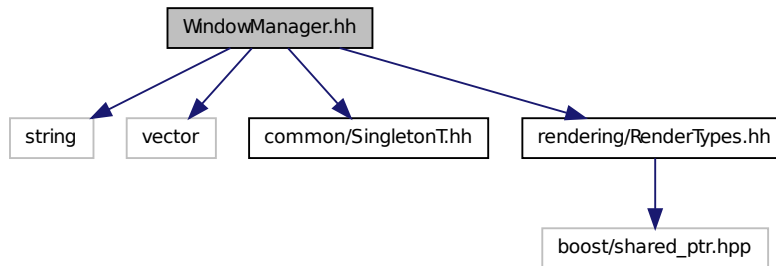
- class **gazebo::rendering::Visual**  
A (p. 107) renderable object.

## Namespaces

- namespace **gazebo**  
Forward declarations for the common classes.
- namespace **gazebo::rendering**  
Rendering namespace.
- namespace **Ogre**

## 11.171 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 manage render windows.*

### Namespaces

- namespace **gazebo**

*Forward declarations for the common classes.*

- namespace **gazebo::rendering**

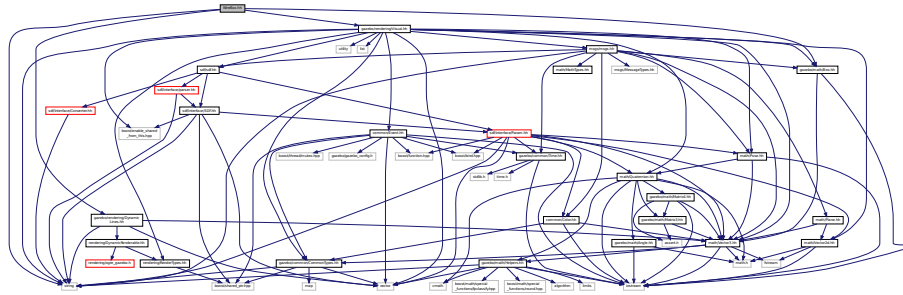
*Rendering namespace.*

- namespace **Ogre**

## 11.172 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.*

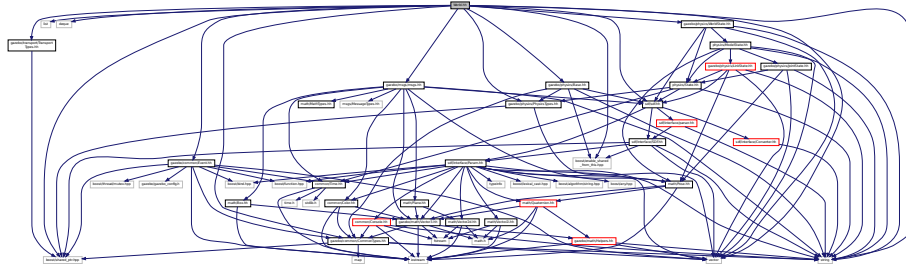
- namespace **gazebo::rendering**

*Rendering namespace.*

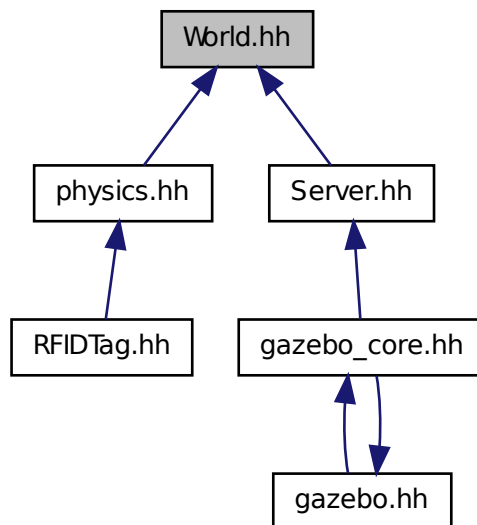
## 11.173 World.hh File Reference

```
#include <vector>
#include <list>
#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/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.*

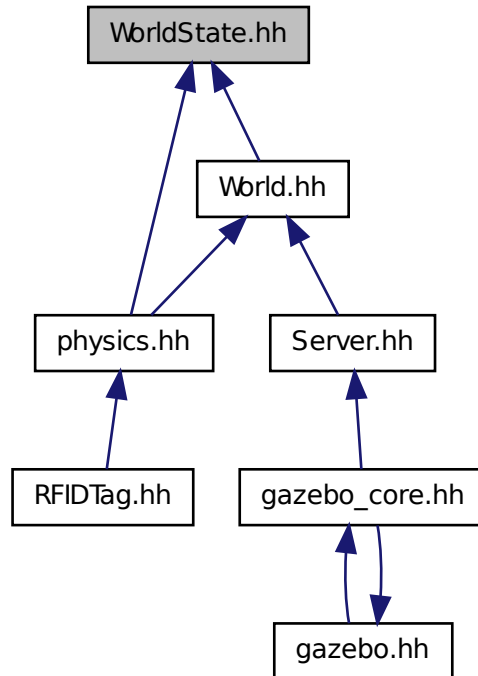
## Namespaces

- namespace **boost**
- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*





This graph shows which files directly or indirectly include this file:



## Classes

- class **gazebo::physics::WorldState**  
*Store state information of a **physics::World** (p. 875) object.*

## Namespaces

- namespace **gazebo**  
*Forward declarations for the common classes.*
- namespace **gazebo::physics**  
*namespace for physics*

# Index

- ~Actor
  - gazebo::physics::Actor, 110
- ~Angle
  - gazebo::math::Angle, 116
- ~Animation
  - gazebo::common::Animation, 123
- ~ArrowVisual
  - gazebo::rendering::ArrowVisual, 127
- ~AssertionInternalError
  - gazebo::common::AssertionInternalError, 129
- ~AxisVisual
  - gazebo::rendering::AxisVisual, 130
- ~BVHLoader
  - gazebo::common::BVHLoader, 152
- ~BallJoint
  - gazebo::physics::BallJoint, 133
- ~Base
  - gazebo::physics::Base, 137
- ~Box
  - gazebo::math::Box, 146
- ~BoxShape
  - gazebo::physics::BoxShape, 151
- ~COMVisual
  - gazebo::rendering::COMVisual, 215
- ~CallbackHelper
  - gazebo::transport::CallbackHelper, 154
- ~Camera
  - gazebo::rendering::Camera, 164
- ~CameraSensor
  - gazebo::sensors::CameraSensor, 184
- ~CameraVisual
  - gazebo::rendering::CameraVisual, 188
- ~ColladaLoader
  - gazebo::common::ColladaLoader, 189
- ~Collision
  - gazebo::physics::Collision, 192
- ~CollisionState
  - gazebo::physics::CollisionState, 201
- ~Color
  - gazebo::common::Color, 206
- ~Connection
  - gazebo::event::Connection, 216
  - gazebo::transport::Connection, 219
- ~Contact
  - gazebo::physics::Contact, 230
- ~ContactManager
  - gazebo::physics::ContactManager, 233
- ~ContactSensor
  - gazebo::sensors::ContactSensor, 236
- ~ContactVisual
  - gazebo::rendering::ContactVisual, 240
- ~CylinderShape
  - gazebo::physics::CylinderShape, 244
- ~DepthCamera
  - gazebo::rendering::DepthCamera, 247
- ~DepthCameraSensor
  - gazebo::sensors::DepthCameraSensor, 252
- ~DiagnosticTimer
  - gazebo::common::DiagnosticTimer, 258
- ~DynamicLines
  - gazebo::rendering::DynamicLines, 260
- ~DynamicRenderable
  - gazebo::rendering::DynamicRenderable, 264
- ~Element
  - sdf::Element, 269
- ~Entity
  - gazebo::physics::Entity, 277
- ~Event
  - gazebo::event::Event, 287
- ~EventT
  - Events, 38
- ~Exception
  - gazebo::common::Exception, 311
- ~FPSViewController
  - gazebo::rendering::FPSViewController, 314
- ~GUIOverlay
  - gazebo::rendering::GUIOverlay, 338
- ~GazeboGenerator
  - google::protobuf::compiler::cpp::GazeboGenerator, 316
- ~GpuLaser
  - gazebo::rendering::GpuLaser, 318
- ~GpuRaySensor
  - gazebo::sensors::GpuRaySensor, 323
- ~Grid
  - gazebo::rendering::Grid, 333
- ~Gripper
  - gazebo::physics::Gripper, 336
- ~GzTerrainMatGen
  - gazebo::rendering::GzTerrainMatGen, 342

- ~Heightmap
  - gazebo::rendering::Heightmap, 343
- ~HeightmapShape
  - gazebo::physics::HeightmapShape, 346
- ~Hinge2Joint
  - gazebo::physics::Hinge2Joint, 349
- ~HingeJoint
  - gazebo::physics::HingeJoint, 351
- ~IOManager
  - gazebo::transport::IOManager, 370
- ~Image
  - gazebo::common::Image, 354
- ~ImuSensor
  - gazebo::sensors::ImuSensor, 359
- ~Inertial
  - gazebo::physics::Inertial, 363
- ~InternalError
  - gazebo::common::InternalError, 369
- ~Joint
  - gazebo::physics::Joint, 375
- ~JointState
  - gazebo::physics::JointState, 389
- ~JointVisual
  - gazebo::rendering::JointVisual, 392
- ~KeyFrame
  - gazebo::common::KeyFrame, 395
- ~LaserVisual
  - gazebo::rendering::LaserVisual, 397
- ~Light
  - gazebo::rendering::Light, 399
- ~Link
  - gazebo::physics::Link, 408
- ~LinkState
  - gazebo::physics::LinkState, 423
- ~Master
  - gazebo::Master, 433
- ~Material
  - gazebo::common::Material, 437
- ~Matrix3
  - gazebo::math::Matrix3, 445
- ~Matrix4
  - gazebo::math::Matrix4, 450
- ~Mesh
  - gazebo::common::Mesh, 456
- ~MeshCSG
  - gazebo::common::MeshCSG, 461
- ~MeshLoader
  - gazebo::common::MeshLoader, 463
- ~Model
  - gazebo::physics::Model, 472
- ~ModelPlugin
  - gazebo::ModelPlugin, 484
- ~ModelState
  - gazebo::physics::ModelState, 487
- ~MovableText
  - gazebo::rendering::MovableText, 497
- ~MultiCameraSensor
  - gazebo::sensors::MultiCameraSensor, 503
- ~MultiRayShape
  - gazebo::physics::MultiRayShape, 509
- ~Node
  - gazebo::transport::Node, 516
- ~NodeAnimation
  - gazebo::common::NodeAnimation, 521
- ~NodeTransform
  - gazebo::common::NodeTransform, 527
- ~NumericAnimation
  - gazebo::common::NumericAnimation, 531
- ~NumericKeyFrame
  - gazebo::common::NumericKeyFrame, 533
- ~OrbitViewController
  - gazebo::rendering::OrbitViewController, 535
- ~PID
  - gazebo::common::PID, 559
- ~Param
  - sdf::Param, 539
- ~ParamT
  - sdf::ParamT, 544
- ~PhysicsEngine
  - gazebo::physics::PhysicsEngine, 549
- ~Plane
  - gazebo::math::Plane, 564
- ~PlaneShape
  - gazebo::physics::PlaneShape, 566
- ~Pose
  - gazebo::math::Pose, 574
- ~PoseAnimation
  - gazebo::common::PoseAnimation, 582
- ~PoseKeyFrame
  - gazebo::common::PoseKeyFrame, 584
- ~Projector
  - gazebo::rendering::Projector, 586
- ~Publication
  - gazebo::transport::Publication, 588
- ~PublicationTransport
  - gazebo::transport::PublicationTransport, 593
- ~Publisher
  - gazebo::transport::Publisher, 595
- ~Quaternion
  - gazebo::math::Quaternion, 601
- ~RFIDSensor
  - gazebo::sensors::RFIDSensor, 633
- ~RFIDTag
  - gazebo::sensors::RFIDTag, 635
- ~RFIDTagVisual
  - gazebo::rendering::RFIDTagVisual, 638
- ~RFIDVisual
  - gazebo::rendering::RFIDVisual, 639

- ~RaySensor
  - gazebo::sensors::RaySensor, 616
- ~RayShape
  - gazebo::physics::RayShape, 624
- ~Road
  - gazebo::physics::Road, 640
- ~Road2d
  - gazebo::rendering::Road2d, 641
- ~RotationSpline
  - gazebo::math::RotationSpline, 643
- ~SM2Profile
  - gazebo::rendering::GzTerrainMatGen::SM2Profile, 721
- ~STLLoader
  - gazebo::common::STLLoader, 733
- ~Scene
  - gazebo::rendering::Scene, 653
- ~ScrewJoint
  - gazebo::physics::ScrewJoint, 666
- ~SelectionObj
  - gazebo::rendering::SelectionObj, 670
- ~Sensor
  - gazebo::sensors::Sensor, 674
- ~SensorPlugin
  - gazebo::SensorPlugin, 686
- ~Server
  - gazebo::Server, 687
- ~Shape
  - gazebo::physics::Shape, 694
- ~SingletonT
  - SingletonT, 697
- ~Skeleton
  - gazebo::common::Skeleton, 699
- ~SkeletonAnimation
  - gazebo::common::SkeletonAnimation, 705
- ~SkeletonNode
  - gazebo::common::SkeletonNode, 711
- ~SliderJoint
  - gazebo::physics::SliderJoint, 718
- ~SphereShape
  - gazebo::physics::SphereShape, 723
- ~Spline
  - gazebo::math::Spline, 725
- ~State
  - gazebo::physics::State, 730
- ~SubMesh
  - gazebo::common::SubMesh, 736
- ~Subscriber
  - gazebo::transport::Subscriber, 746
- ~SubscriptionTransport
  - gazebo::transport::SubscriptionTransport, 748
- ~SurfaceParams
  - gazebo::physics::SurfaceParams, 750
- ~SystemPlugin
  - gazebo::SystemPlugin, 759
- ~Time
  - gazebo::common::Time, 764
- ~Timer
  - gazebo::common::Timer, 781
- ~TrimeshShape
  - gazebo::physics::TrimeshShape, 790
- ~URDF2Gazebo
  - urdf2gazebo::URDF2Gazebo, 794
- ~UniversalJoint
  - gazebo::physics::UniversalJoint, 793
- ~UserCamera
  - gazebo::rendering::UserCamera, 797
- ~Vector2d
  - gazebo::math::Vector2d, 805
- ~Vector2i
  - gazebo::math::Vector2i, 813
- ~Vector3
  - gazebo::math::Vector3, 823
- ~Vector4
  - gazebo::math::Vector4, 836
- ~Video
  - gazebo::common::Video, 843
- ~VideoVisual
  - gazebo::rendering::VideoVisual, 845
- ~ViewController
  - gazebo::rendering::ViewController, 847
- ~Visual
  - gazebo::rendering::Visual, 855
- ~WireBox
  - gazebo::rendering::WireBox, 874
- ~World
  - gazebo::physics::World, 877
- ~WorldPlugin
  - gazebo::WorldPlugin, 887
- ~WorldState
  - gazebo::physics::WorldState, 889
- \_setupGeometry
  - gazebo::rendering::MovableText, 497
- \_updateColors
  - gazebo::rendering::MovableText, 497
- A, 107
- a
  - gazebo::common::Color, 213
- ABGR
  - gazebo::common::Color, 205
- ACTOR
  - gazebo::physics::Base, 136
- ADD
  - gazebo::common::Material, 437
- ARGB
  - gazebo::common::Color, 205
- AcceptCallback

- gazebo::transport::Connection, 219
- active
  - gazebo::physics::Actor, 112
  - gazebo::sensors::Sensor, 679
- Actor
  - gazebo::physics::Actor, 110
- Actor.hh, 893
- Actor\_V
  - gazebo::physics, 93
- ActorPtr
  - gazebo::physics, 93
- Add
  - gazebo::common::LogRecord, 431
- add\_plugin
  - gazebo, 81
- add\_search\_path\_suffix
  - Common, 32
- AddAnimation
  - gazebo::common::Skeleton, 699
- AddAttribute
  - sdf::Element, 269
- AddCallback
  - gazebo::transport::PublicationTransport, 593
- AddChild
  - gazebo::common::SkeletonNode, 711
  - gazebo::physics::Base, 137
- AddChildJoint
  - gazebo::physics::Link, 408
- AddContact
  - gazebo::physics::Collision, 193
- AddElement
  - sdf::Element, 269
- AddElementDescription
  - sdf::Element, 269
- addEntity
  - gazebo::event::Events, 300
- AddForce
  - gazebo::physics::Link, 408
- AddForceAtRelativePosition
  - gazebo::physics::Link, 408
- AddForceAtWorldPosition
  - gazebo::physics::Link, 408
- AddGazeboPaths
  - gazebo::common::SystemPaths, 755
- AddIndex
  - gazebo::common::SubMesh, 736
- AddJoint
  - gazebo::physics::JointController, 386
- AddKeyFrame
  - gazebo::common::NodeAnimation, 521, 522
  - gazebo::common::SkeletonAnimation, 705
- AddMaterial
  - gazebo::common::Mesh, 456
- AddMesh
  - gazebo::common::MeshManager, 465
- addNestedModel
  - sdf, 104
- AddNode
  - gazebo::transport::TopicManager, 783
- AddNodeAssignment
  - gazebo::common::SubMesh, 737
- AddNormal
  - gazebo::common::SubMesh, 737
- AddOgrePaths
  - gazebo::common::SystemPaths, 755
- AddParentJoint
  - gazebo::physics::Link, 409
- AddPluginPaths
  - gazebo::common::SystemPaths, 755
- AddPoint
  - gazebo::math::RotationSpline, 643
  - gazebo::math::Spline, 725
  - gazebo::rendering::DynamicLines, 260
- AddPublisher
  - gazebo::transport::Publication, 589
- AddRawTransform
  - gazebo::common::SkeletonNode, 711
- AddRay
  - gazebo::physics::MultiRayShape, 509
- AddRelativeForce
  - gazebo::physics::Link, 409
- AddRelativeTorque
  - gazebo::physics::Link, 409
- AddResourcePath
  - gazebo::rendering::RenderEngine, 629
- AddScene
  - gazebo::rendering::RTShaderSystem, 647
- AddSearchPathSuffix
  - gazebo::common::SystemPaths, 755
- AddSubMesh
  - gazebo::common::Mesh, 456
- AddSubscription
  - gazebo::transport::Publication, 589
- AddTag
  - gazebo::sensors::RFIDSensor, 633
- addTechnique
  - gazebo::rendering::GzTerrainMatGen::SM2Profile, 721
- AddTexCoord
  - gazebo::common::SubMesh, 737
- AddTime
  - gazebo::common::Animation, 123
- AddTorque
  - gazebo::physics::Link, 409
- AddTransport
  - gazebo::transport::Publication, 589
- AddType
  - gazebo::physics::Base, 137

- AddValue
  - sdf::Element, 269
- AddVertNodeWeight
  - gazebo::common::Skeleton, 699
- AddVertex
  - gazebo::common::SubMesh, 737
- AddVisual
  - gazebo::rendering::Scene, 653
- Advertise
  - gazebo::transport::ConnectionManager, 225
  - gazebo::transport::Node, 516
  - gazebo::transport::TopicManager, 784
- alt
  - gazebo::common::MouseEvent, 493
- ambient
  - gazebo::common::Material, 443
- anchorLink
  - gazebo::physics::Joint, 384
- anchorPos
  - gazebo::physics::Joint, 384
- Angle
  - gazebo::math::Angle, 115
- Angle.hh, 894
  - GZ\_DTOR, 896
  - GZ\_NORMALIZE, 896
  - GZ\_RTOD, 896
- angularAccel
  - gazebo::physics::Link, 420
- animState
  - gazebo::rendering::Camera, 180
- Animation
  - gazebo::common::Animation, 123
- animation
  - gazebo::physics::Entity, 284
- Animation.hh, 897
- AnimationComplete
  - gazebo::rendering::Camera, 164
  - gazebo::rendering::UserCamera, 797
- animationConnection
  - gazebo::physics::Entity, 284
- AnimationPtr
  - gazebo::common, 84
- animationStartPose
  - gazebo::physics::Entity, 284
- animations
  - gazebo::common::SkeletonAnimation, 707
- anims
  - gazebo::common::Skeleton, 703
- ApplyDamping
  - gazebo::physics::Joint, 375
- applyDamping
  - gazebo::physics::Joint, 384
- ApplyShadows
  - gazebo::rendering::RTShaderSystem, 648
- AreConnected
  - gazebo::physics::Joint, 375
- ArrowVisual
  - gazebo::rendering::ArrowVisual, 126
- ArrowVisual.hh, 898
- ArrowVisualPtr
  - gazebo::rendering, 97
- Assert.hh, 899
  - GZ\_ASSERT, 900
- AssertionInternalError
  - gazebo::common::AssertionInternalError, 128
- AsyncRead
  - gazebo::transport::Connection, 219
- Attach
  - gazebo::physics::Joint, 375
  - gazebo::rendering::SelectionObj, 670
- AttachAxes
  - gazebo::rendering::Visual, 855
- AttachCameraToImage
  - gazebo::rendering::GUIOverlay, 338
- AttachEntity
  - gazebo::rendering::RTShaderSystem, 648
- AttachLineVertex
  - gazebo::rendering::Visual, 855
- AttachMesh
  - gazebo::rendering::Visual, 855
- AttachObject
  - gazebo::rendering::Visual, 855
- AttachStaticModel
  - gazebo::physics::Link, 409
  - gazebo::physics::Model, 472
- AttachToVisual
  - gazebo::rendering::Camera, 164
- AttachToVisualImpl
  - gazebo::rendering::Camera, 164, 165
  - gazebo::rendering::UserCamera, 797
- AttachViewport
  - gazebo::rendering::RTShaderSystem, 648
- AttachVisual
  - gazebo::rendering::Visual, 855
- attachedModels
  - gazebo::physics::Model, 481
- attachedModelsOffset
  - gazebo::physics::Link, 420
  - gazebo::physics::Model, 481
- Attribute
  - gazebo::physics::Joint, 374
- autoCalc
  - gazebo::math::RotationSpline, 645
  - gazebo::math::Spline, 728
- autoStart
  - gazebo::physics::Actor, 112
- AxisVisual
  - gazebo::rendering::AxisVisual, 130

- AxisVisual.hh, 900
- AxisVisualPtr
  - gazebo::rendering, 97
- b
  - gazebo::common::Color, 213
- BALL\_JOINT
  - gazebo::physics::Base, 137
- BASE
  - gazebo::physics::Base, 136
- BAYER\_GBRG8
  - gazebo::common::Image, 354
- BAYER\_GRBG8
  - gazebo::common::Image, 354
- BAYER\_RRGB8
  - gazebo::common::Image, 354
- BAYER\_RGGR8
  - gazebo::common::Image, 354
- BGR\_INT16
  - gazebo::common::Image, 353
- BGR\_INT32
  - gazebo::common::Image, 353
- BGR\_INT8
  - gazebo::common::Image, 353
- BGRA
  - gazebo::common::Color, 205
- BGRA\_INT8
  - gazebo::common::Image, 353
- BLEND\_COUNT
  - gazebo::common::Material, 437
- BLINN
  - gazebo::common::Material, 437
- BOX\_SHAPE
  - gazebo::physics::Base, 137
- BVHLoader
  - gazebo::common::BVHLoader, 152
- BVHLoader.hh, 905
  - X\_POSITION, 906
  - X\_ROTATION, 906
  - Y\_POSITION, 906
  - Y\_ROTATION, 906
  - Z\_POSITION, 906
  - Z\_ROTATION, 906
- BallJoint
  - gazebo::physics::BallJoint, 132
- BallJoint.hh, 901
- Base
  - gazebo::physics::Base, 137
- Base.hh, 902
- Base\_V
  - gazebo::physics, 93
- BasePtr
  - gazebo::physics, 93
- bayerFrameBuffer
  - gazebo::rendering::Camera, 180
- bindShapeTransform
  - gazebo::common::Skeleton, 703
- Black
  - gazebo::common::Color, 213
- BlendMode
  - gazebo::common::Material, 437
- blendMode
  - gazebo::common::Material, 443
- BlendModeStr
  - gazebo::common::Material, 443
- Blue
  - gazebo::common::Color, 213
- body1Force
  - gazebo::physics::JointWrench, 393
- body1Torque
  - gazebo::physics::JointWrench, 393
- body2Force
  - gazebo::physics::JointWrench, 394
- body2Torque
  - gazebo::physics::JointWrench, 394
- bonePosePub
  - gazebo::physics::Actor, 112
- BooleanOperation
  - gazebo::common::MeshCSG, 461
- boost, 79
- bounce
  - gazebo::physics::SurfaceParams, 750
- bounceThreshold
  - gazebo::physics::SurfaceParams, 751
- Box
  - gazebo::math::Box, 146
- Box.hh, 903
- BoxShape
  - gazebo::physics::BoxShape, 151
- BoxShape.hh, 903
- BoxShapePtr
  - gazebo::physics, 93
- build
  - gazebo::common::Animation, 124
- BuildInterpolationSplines
  - gazebo::common::PoseAnimation, 582
- BuildNodeMap
  - gazebo::common::Skeleton, 700
- button
  - gazebo::common::MouseEvent, 493
- ButtonCallback
  - gazebo::rendering::GUIOverlay, 339
- Buttons
  - gazebo::common::MouseEvent, 493
- buttons
  - gazebo::common::MouseEvent, 493
- CFM

- gazebo::physics::Joint, 374
- COLLISION
  - gazebo::physics::Base, 136
- COMVisual
  - gazebo::rendering::COMVisual, 215
- COMVisual.hh, 919
- COMVisualPtr
  - gazebo::rendering, 97
- COR3\_MAX
  - STLLoader.hh, 1060
- CYLINDER\_SHAPE
  - gazebo::physics::Base, 137
- CallbackHelper
  - gazebo::transport::CallbackHelper, 154
- CallbackHelper.hh, 906
- CallbackHelperPtr
  - Transport, 74
- CallbackHelperT
  - gazebo::transport::CallbackHelperT, 156
- Camera
  - gazebo::rendering::Camera, 163
- camera
  - gazebo::rendering::Camera, 180
  - gazebo::rendering::ViewController, 849
- Camera.hh, 908
- cameraCount
  - gazebo::sensors::GpuRaySensor, 330
- cameraElem
  - gazebo::sensors::GpuRaySensor, 330
- CameraPtr
  - gazebo::rendering, 97
- CameraSensor
  - gazebo::sensors::CameraSensor, 184
- CameraSensor.hh, 909
- CameraSensor\_V
  - gazebo::sensors, 99
- CameraSensorPtr
  - gazebo::sensors, 99
- CameraVisual
  - gazebo::rendering::CameraVisual, 188
- CameraVisual.hh, 910
- CameraVisualPtr
  - gazebo::rendering, 97
- Cancel
  - gazebo::transport::Connection, 219
- captureData
  - gazebo::rendering::Camera, 180
- cegui.h, 910
- cfm
  - gazebo::physics::SurfaceParams, 751
- cgVisuals
  - gazebo::physics::Link, 420
- chfov
  - gazebo::sensors::GpuRaySensor, 330
- childLink
  - gazebo::physics::Joint, 384
- children
  - gazebo::common::SkeletonNode, 716
  - gazebo::physics::Base, 144
- childrenEnd
  - gazebo::physics::Base, 144
- 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, 643
  - gazebo::math::Spline, 725
  - gazebo::physics::ContactManager, 233
  - gazebo::physics::World, 878
  - gazebo::rendering::DynamicLines, 261
  - gazebo::rendering::RTShaderSystem, 648
  - gazebo::rendering::Scene, 654
  - gazebo::rendering::SelectionObj, 670
  - sdf::Plugin, 569
- clear\_buffers
  - Transport, 75
- ClearBuffers
  - gazebo::transport::TopicManager, 784
- ClearElements
  - sdf::Element, 269
- ClearGazeboPaths
  - gazebo::common::SystemPaths, 755
- ClearOgrePaths
  - gazebo::common::SystemPaths, 755
- ClearParent
  - gazebo::rendering::Visual, 856
- ClearPluginPaths
  - gazebo::common::SystemPaths, 755
- Clone
  - gazebo::rendering::Visual, 856



- sdf::Element, 269
  - sdf::Param, 539
  - sdf::ParamT, 545
- CloneVisual
  - gazebo::rendering::Scene, 654
- coeffs
  - gazebo::math::Spline, 728
- ColladaLoader
  - gazebo::common::ColladaLoader, 189
- ColladaLoader.hh, 911
- Collision
  - gazebo::physics::Collision, 192
- Collision.hh, 913
- collision1
  - gazebo::physics::Contact, 231
- collision2
  - gazebo::physics::Contact, 231
- Collision\_V
  - gazebo::physics, 93
- collisionParent
  - gazebo::physics::Shape, 695
- CollisionPtr
  - gazebo::physics, 93
- CollisionState
  - gazebo::physics::CollisionState, 200
- CollisionState.hh, 914
- Color
  - gazebo::common::Color, 205, 206
- Color.hh, 916
- ColorErr
  - Common, 33
- ColorMsg
  - Common, 33
- Common, 27
  - add\_search\_path\_suffix, 32
  - ColorErr, 33
  - ColorMsg, 33
  - DIAG\_TIMER, 31
  - DiagnosticTimerPtr, 32
  - DownloadDependencies, 33
  - find\_file, 33
  - find\_file\_path, 34
  - GetDBConfig, 34
  - GetManifest, 34
  - GetModelConfig, 34
  - GetModelFile, 34
  - GetModelName, 35
  - GetModelPath, 35
  - GetModels, 35
  - GetURI, 36
  - gzclr\_end, 31
  - gzclr\_start, 31
  - gzdbg, 31
  - gzerr, 31
  - gzlog, 31
  - gzmsg, 31
  - gzthrow, 32
  - gzwarn, 32
  - HasModel, 36
  - Init, 36
  - Log, 36
  - MODEL\_PLUGIN, 32
  - NullStream, 32
  - PixelFormatNames, 37
  - PluginType, 32
  - SENSOR\_PLUGIN, 32
  - SYSTEM\_PLUGIN, 32
  - SetQuiet, 36
  - VISUAL\_PLUGIN, 32
  - WORLD\_PLUGIN, 32
- Common.hh, 916
- common/Plugin.hh
  - GZ\_REGISTER\_MODEL\_PLUGIN, 1014
  - GZ\_REGISTER\_SENSOR\_PLUGIN, 1014
  - GZ\_REGISTER\_SYSTEM\_PLUGIN, 1015
  - GZ\_REGISTER\_VISUAL\_PLUGIN, 1015
  - GZ\_REGISTER\_WORLD\_PLUGIN, 1015
- CommonTypes.hh, 918
  - GAZEBO\_DEPRECATED, 919
  - GAZEBO\_FORCEINLINE, 919
  - NULL, 919
- Connect
  - Events, 38
  - gazebo::transport::Connection, 219
- ConnectAddEntity
  - gazebo::event::Events, 292
- ConnectContact
  - gazebo::physics::Collision, 193
- ConnectCreateEntity
  - gazebo::event::Events, 292
- ConnectCreateScene
  - gazebo::rendering::Events, 288
- ConnectDeleteEntity
  - gazebo::event::Events, 293
- ConnectDiagTimerStart
  - gazebo::event::Events, 293
- ConnectDiagTimerStop
  - gazebo::event::Events, 293
- ConnectEnabled
  - gazebo::physics::Link, 410
- ConnectJointUpdate
  - gazebo::physics::Joint, 375
- ConnectNewDepthFrame
  - gazebo::rendering::DepthCamera, 248
- ConnectNewImageFrame
  - gazebo::rendering::Camera, 165
- ConnectNewLaserFrame
  - gazebo::rendering::GpuLaser, 318

- gazebo::sensors::GpuRaySensor, 324
- ConnectNewLaserScans
  - gazebo::physics::MultiRayShape, 509
- ConnectNewRGBPointCloud
  - gazebo::rendering::DepthCamera, 248
- ConnectPause
  - gazebo::event::Events, 294
- ConnectPostRender
  - gazebo::event::Events, 294
- ConnectPreRender
  - gazebo::event::Events, 294
- ConnectPubToSub
  - gazebo::transport::TopicManager, 784
- ConnectRemoveScene
  - gazebo::rendering::Events, 288
- ConnectRender
  - gazebo::event::Events, 295
- ConnectSetSelectedEntity
  - gazebo::event::Events, 295
- ConnectStep
  - gazebo::event::Events, 295
- ConnectStop
  - gazebo::event::Events, 295
- ConnectSubToPub
  - gazebo::transport::TopicManager, 784
- ConnectSubscribers
  - gazebo::transport::TopicManager, 784
- ConnectToRemoteHost
  - gazebo::transport::ConnectionManager, 225
- ConnectToShutdown
  - gazebo::transport::Connection, 220
- ConnectUpdated
  - gazebo::sensors::Sensor, 674
- ConnectWorldCreated
  - gazebo::event::Events, 296
- ConnectWorldUpdateEnd
  - gazebo::event::Events, 296
- ConnectWorldUpdateStart
  - gazebo::event::Events, 296
- Connection
  - gazebo::event::Connection, 216
  - gazebo::transport::Connection, 219
- Connection.hh, 920
  - HEADER\_LENGTH, 922
- Connection\_V
  - gazebo::event, 85
- ConnectionCount
  - Events, 39
- ConnectionManager.hh, 922
- ConnectionPtr
  - gazebo::event, 85
  - gazebo::transport, 102
- connections
  - gazebo::physics::Entity, 284
- gazebo::rendering::Camera, 180
- gazebo::sensors::Sensor, 679
- Console.hh, 923
- ConstUrdflinkPtr
  - Gazebo\_parser, 68
- Contact
  - gazebo::physics::Contact, 230
- Contact.hh, 925
  - MAX\_COLLIDE\_RETURNS, 926
  - MAX\_CONTACT\_JOINTS, 926
- contactFiducial
  - gazebo::physics::RayShape, 626
- contactLen
  - gazebo::physics::RayShape, 626
- ContactManager
  - gazebo::physics::ContactManager, 233
- contactManager
  - gazebo::physics::PhysicsEngine, 556
- ContactManager.hh, 926
- ContactPtr
  - gazebo::physics, 93
- contactRetro
  - gazebo::physics::RayShape, 627
- ContactSensor
  - gazebo::sensors::ContactSensor, 236
- ContactSensor.hh, 927
- ContactSensor\_V
  - gazebo::sensors, 99
- ContactSensorPtr
  - gazebo::sensors, 99
- ContactVisual
  - gazebo::rendering::ContactVisual, 239
- ContactVisual.hh, 928
- ContactVisualPtr
  - gazebo::rendering, 97
- control
  - gazebo::common::MouseEvent, 493
- Conversions.hh, 929
- Convert
  - gazebo::rendering::Conversions, 241, 242
  - Messages, 56–59
  - sdf::Converter, 242
- ConvertPixelFormat
  - gazebo::common::Image, 354
- Converter.hh, 929
- CoordPoseSolve
  - gazebo::math::Pose, 574
- CoordPositionAdd
  - gazebo::math::Pose, 575
- CoordPositionSub
  - gazebo::math::Pose, 575
- CoordRotationAdd
  - gazebo::math::Pose, 575
- CoordRotationSub

- gazebo::math::Pose, 576
- Copy
  - sdf::Element, 270
- copyChildren
  - sdf, 104
- CopyNormals
  - gazebo::common::SubMesh, 738
- CopyVertices
  - gazebo::common::SubMesh, 738
- Correct
  - gazebo::math::Pose, 576
  - gazebo::math::Quaternion, 601
  - gazebo::math::Vector3, 823
- count
  - gazebo::physics::Contact, 231
- Create
  - gazebo::PluginT, 571
- create\_scene
  - Rendering, 66
- create\_sensor
  - Sensors, 71
- create\_world
  - Classes for physics and dynamics, 44
- CreateBoolean
  - gazebo::common::MeshCSG, 461
- CreateBox
  - gazebo::common::MeshManager, 465
- CreateCamera
  - gazebo::common::MeshManager, 465
  - gazebo::rendering::Scene, 654
- CreateCollision
  - gazebo::physics::PhysicsEngine, 549
- CreateCone
  - gazebo::common::MeshManager, 465
- CreateCylinder
  - gazebo::common::MeshManager, 465
- CreateDepthCamera
  - gazebo::rendering::Scene, 654
- CreateDepthTexture
  - gazebo::rendering::DepthCamera, 248
- CreateDynamicLine
  - gazebo::rendering::Visual, 856
- CreateGrid
  - gazebo::rendering::Scene, 655
- CreateJoint
  - gazebo::physics::PhysicsEngine, 550
- CreateKeyFrame
  - gazebo::common::NumericAnimation, 531
  - gazebo::common::PoseAnimation, 582
- CreateLaserTexture
  - gazebo::rendering::GpuLaser, 318
- CreateLink
  - gazebo::physics::PhysicsEngine, 550
- CreatePlane
  - gazebo::common::MeshManager, 466
  - gazebo::physics::PlaneShape, 567
- CreateRenderTexture
  - gazebo::rendering::Camera, 165
- CreateRequest
  - Messages, 59
- CreateScene
  - gazebo::rendering::RenderEngine, 629
- createScene
  - gazebo::rendering::Events, 289
- CreateSensor
  - gazebo::sensors::SensorManager, 683
- CreateShape
  - gazebo::physics::PhysicsEngine, 550
- CreateSphere
  - gazebo::common::MeshManager, 466
- CreateTimer
  - gazebo::common::DiagnosticManager, 255
- CreateTube
  - gazebo::common::MeshManager, 466
- CreateUserCamera
  - gazebo::rendering::Scene, 655
- CreateVertexDeclaration
  - gazebo::rendering::DynamicLines, 261
  - gazebo::rendering::DynamicRenderable, 264
- CreateWindow
  - gazebo::rendering::GUIOverlay, 339
  - gazebo::rendering::WindowManager, 871
- Cross
  - gazebo::math::Vector2d, 805
  - gazebo::math::Vector2i, 814
  - gazebo::math::Vector3, 824
- cvfov
  - gazebo::sensors::GpuRaySensor, 330
- CylinderShape
  - gazebo::physics::CylinderShape, 244
- CylinderShape.hh, 930
- CylinderShapePtr
  - gazebo::physics, 93
- d
  - gazebo::math::Plane, 564
- DEFERRED
  - gazebo::rendering::RenderEngine, 629
- DIAG\_TIMER
  - Common, 31
- DIFFERENCE
  - gazebo::common::MeshCSG, 461
- dampingCoefficient
  - gazebo::physics::Joint, 384
- data
  - sdf::Plugin, 569
- DebugPrint
  - gazebo::physics::PhysicsEngine, 550

- DebugString
  - gazebo::physics::Contact, 230
- DecCount
  - gazebo::transport::IOManager, 370
- DecodeTopicName
  - gazebo::transport::Node, 516
- defaultValue
  - sdf::ParamT, 546
- defaultVpParams
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 689
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 691
- Degree
  - gazebo::math::Angle, 116
- DeleteDynamicLine
  - gazebo::rendering::Visual, 856
- deleteEntity
  - gazebo::event::Events, 300
- DepthCamera
  - gazebo::rendering::DepthCamera, 247
- DepthCamera.hh, 931
- DepthCameraPtr
  - gazebo::rendering, 97
- DepthCameraSensor
  - gazebo::sensors::DepthCameraSensor, 252
- DepthCameraSensor.hh, 932
- DepthCameraSensor\_V
  - gazebo::sensors, 99
- DepthCameraSensorPtr
  - gazebo::sensors, 99
- depthTarget
  - gazebo::rendering::DepthCamera, 250
- depthTexture
  - gazebo::rendering::DepthCamera, 250
- depthViewport
  - gazebo::rendering::DepthCamera, 250
- depths
  - gazebo::physics::Contact, 231
- description
  - sdf::Param, 542
- Detach
  - gazebo::physics::Joint, 376
- DetachAllStaticModels
  - gazebo::physics::Link, 410
- DetachEntity
  - gazebo::rendering::RTShaderSystem, 648
- DetachObjects
  - gazebo::rendering::Visual, 856
- DetachStaticModel
  - gazebo::physics::Link, 410
  - gazebo::physics::Model, 472
- DetachViewport
  - gazebo::rendering::RTShaderSystem, 648
- DetachVisual
  - gazebo::rendering::Visual, 856, 857
- diagTimerStart
  - gazebo::event::Events, 300
- diagTimerStop
  - gazebo::event::Events, 300
- DiagnosticTimer
  - gazebo::common::DiagnosticTimer, 257
- DiagnosticTimerPtr
  - Common, 32
- Diagnostics.hh, 933
- diffuse
  - gazebo::common::Material, 443
- dirtyPose
  - gazebo::physics::Entity, 285
- dirtyPoses
  - gazebo::physics::World, 886
- DisableAllModels
  - gazebo::physics::World, 878
- DisableTrackVisual
  - gazebo::rendering::Visual, 857
- Disconnect
  - Events, 39, 40
  - gazebo::event::Event, 287
- DisconnectAddEntity
  - gazebo::event::Events, 297
- DisconnectContact
  - gazebo::physics::Collision, 193
- DisconnectCreateEntity
  - gazebo::event::Events, 297
- DisconnectCreateScene
  - gazebo::rendering::Events, 289
- DisconnectDeleteEntity
  - gazebo::event::Events, 297
- DisconnectDiagTimerStart
  - gazebo::event::Events, 297
- DisconnectDiagTimerStop
  - gazebo::event::Events, 297
- DisconnectEnabled
  - gazebo::physics::Link, 410
- DisconnectJointUpdate
  - gazebo::physics::Joint, 376
- DisconnectNewDepthFrame
  - gazebo::rendering::DepthCamera, 248
- DisconnectNewImageFrame
  - gazebo::rendering::Camera, 165
- DisconnectNewLaserFrame
  - gazebo::rendering::GpuLaser, 319
  - gazebo::sensors::GpuRaySensor, 324
- DisconnectNewLaserScans
  - gazebo::physics::MultiRayShape, 510
- DisconnectNewRGBPointCloud
  - gazebo::rendering::DepthCamera, 248
- DisconnectPause

- gazebo::event::Events, 298
- DisconnectPostRender
  - gazebo::event::Events, 298
- DisconnectPreRender
  - gazebo::event::Events, 298
- DisconnectPubFromSub
  - gazebo::transport::TopicManager, 784
- DisconnectRemoveScene
  - gazebo::rendering::Events, 289
- DisconnectRender
  - gazebo::event::Events, 298
- DisconnectSetSelectedEntity
  - gazebo::event::Events, 298
- DisconnectShutdown
  - gazebo::transport::Connection, 220
- DisconnectStep
  - gazebo::event::Events, 299
- DisconnectStop
  - gazebo::event::Events, 299
- DisconnectSubFromPub
  - gazebo::transport::TopicManager, 785
- DisconnectUpdated
  - gazebo::sensors::Sensor, 674
- DisconnectWorldCreated
  - gazebo::event::Events, 299
- DisconnectWorldUpdateEnd
  - gazebo::event::Events, 299
- DisconnectWorldUpdateStart
  - gazebo::event::Events, 299
- Distance
  - gazebo::math::Plane, 564
  - gazebo::math::Vector2d, 805
  - gazebo::math::Vector2i, 814
  - gazebo::math::Vector3, 824
  - gazebo::math::Vector4, 836
- Dot
  - gazebo::math::Quaternion, 601
  - gazebo::math::Vector3, 824
- Double
  - gazebo::common::Time, 764
- DownloadDependencies
  - Common, 33
- dragging
  - gazebo::common::MouseEvent, 493
- DrawLine
  - gazebo::rendering::Scene, 655
- dummyContext
  - gazebo::rendering::RenderEngine, 631
- dummyDisplay
  - gazebo::rendering::RenderEngine, 631
- dummyWindowId
  - gazebo::rendering::RenderEngine, 631
- duration
  - gazebo::physics::TrajectoryInfo, 788
- DynamicLines
  - gazebo::rendering::DynamicLines, 260
- DynamicLines.hh, 935
- DynamicLinesPtr
  - gazebo::rendering, 97
- DynamicRenderable
  - gazebo::rendering::DynamicRenderable, 264
- DynamicRenderable.hh, 935
- ENTITY
  - gazebo::physics::Base, 136
- ERP
  - gazebo::physics::Joint, 374
- Element
  - sdf::Element, 269
- ElementPtr
  - sdf, 104
- ElementPtr\_V
  - sdf, 104
- emissive
  - gazebo::common::Material, 443
- Enable
  - gazebo::rendering::Grid, 334
- EnableAllModels
  - gazebo::physics::World, 878
- EnablePhysicsEngine
  - gazebo::physics::World, 878
- EnableSaveFrame
  - gazebo::rendering::Camera, 166
- EnableTrackVisual
  - gazebo::rendering::Visual, 857
- EnableViewController
  - gazebo::rendering::UserCamera, 798
- enabled
  - gazebo::rendering::ViewController, 849
- EncodeTopicName
  - gazebo::transport::Node, 516
- endTime
  - gazebo::physics::TrajectoryInfo, 788
- EnqueueMsg
  - gazebo::physics::World, 878
  - gazebo::transport::Connection, 220
- Entity
  - gazebo::physics::Entity, 277
- Entity.hh, 936
- entityCreated
  - gazebo::event::Events, 300
- EntityPtr
  - gazebo::physics, 93
- EntityType
  - gazebo::physics::Base, 136
- EntityTypename
  - Classes for physics and dynamics, 46
- Equal

- gazebo::math::Vector3, 824
- equal
  - Math, 50
- erp
  - gazebo::physics::SurfaceParams, 751
- EulerToQuaternion
  - gazebo::math::Quaternion, 602
- Event.hh, 938
- eventConnections
  - gazebo::transport::ConnectionManager, 228
- EventType
  - gazebo::common::MouseEvent, 493
- Events, 38
  - ~EventT, 38
  - Connect, 38
  - ConnectionCount, 39
  - Disconnect, 39, 40
- Events.hh, 938
- Exception
  - gazebo::common::Exception, 311
- Exception.hh, 939
  
- FACE\_MAX
  - STLLoader.hh, 1060
- FLAT
  - gazebo::common::Material, 437
- FMAX
  - gazebo::physics::Joint, 374
- FORWARD
  - gazebo::rendering::RenderEngine, 629
- FPSViewController
  - gazebo::rendering::FPSViewController, 314
- FPSViewController.hh, 941
- FUDGE\_FACTOR
  - gazebo::physics::Joint, 374
- fakeAnchor
  - gazebo::physics::ScrewJoint, 667
  - gazebo::physics::SliderJoint, 719
- far
  - gazebo::sensors::GpuRaySensor, 331
- fdir1
  - gazebo::physics::SurfaceParams, 751
- filename
  - gazebo::PluginT, 571
  - sdf::Plugin, 569
- FillArrays
  - gazebo::common::Mesh, 456
  - gazebo::common::SubMesh, 738
- FillHardwareBuffers
  - gazebo::rendering::DynamicLines, 261
  - gazebo::rendering::DynamicRenderable, 264
- FillMsg
  - gazebo::physics::BoxShape, 151
  - gazebo::physics::Collision, 193
  - gazebo::physics::Contact, 230
  - gazebo::physics::CylinderShape, 244
  - gazebo::physics::HeightmapShape, 346
  - gazebo::physics::Joint, 376
  - gazebo::physics::Link, 410
  - gazebo::physics::Model, 473
  - gazebo::physics::MultiRayShape, 510
  - gazebo::physics::PlaneShape, 567
  - gazebo::physics::RayShape, 624
  - gazebo::physics::Shape, 694
  - gazebo::physics::SphereShape, 723
  - gazebo::physics::SurfaceParams, 750
  - gazebo::physics::TrimeshShape, 790
  - gazebo::rendering::Light, 399
  - gazebo::sensors::Sensor, 674
- FillSDF
  - gazebo::physics::CollisionState, 201
  - gazebo::physics::JointState, 389
  - gazebo::physics::LinkState, 423
  - gazebo::physics::ModelState, 487
  - gazebo::physics::WorldState, 889
- find\_file
  - Common, 33
  - gazebo, 81
- find\_file\_path
  - Common, 34
- FindFile
  - gazebo::common::SystemPaths, 755
- FindFileURI
  - gazebo::common::SystemPaths, 756
- FindPublication
  - gazebo::transport::TopicManager, 785
- Fini
  - gazebo::Master, 434
  - gazebo::physics::Actor, 110
  - gazebo::physics::Base, 138
  - gazebo::physics::Collision, 193
  - gazebo::physics::Entity, 277
  - gazebo::physics::Link, 410
  - gazebo::physics::Model, 473
  - gazebo::physics::PhysicsEngine, 550
  - gazebo::physics::World, 878
  - gazebo::rendering::Camera, 166
  - gazebo::rendering::DepthCamera, 249
  - gazebo::rendering::GpuLaser, 319
  - gazebo::rendering::RenderEngine, 630
  - gazebo::rendering::RTShaderSystem, 649
  - gazebo::rendering::UserCamera, 798
  - gazebo::rendering::Visual, 857
  - gazebo::rendering::WindowManager, 872
  - gazebo::sensors::CameraSensor, 184
  - gazebo::sensors::ContactSensor, 236
  - gazebo::sensors::DepthCameraSensor, 252
  - gazebo::sensors::GpuRaySensor, 324

- gazebo::sensors::ImuSensor, 359
- gazebo::sensors::MultiCameraSensor, 503
- gazebo::sensors::RaySensor, 617
- gazebo::sensors::RFIDSensor, 633
- gazebo::sensors::RFIDTag, 635
- gazebo::sensors::Sensor, 675
- gazebo::sensors::SensorManager, 683
- gazebo::Server, 687
- gazebo::transport::ConnectionManager, 225
- gazebo::transport::Node, 517
- gazebo::transport::PublicationTransport, 593
- gazebo::transport::TopicManager, 785
- fini
  - Classes for physics and dynamics, 44
  - gazebo, 81
  - Rendering, 66
  - Sensors, 71
  - Transport, 75
- Float
  - gazebo::common::Time, 764
- FogFromSDF
  - Messages, 59
- forceApplied
  - gazebo::physics::Joint, 385
- g
  - gazebo::common::Color, 213
- GAZEBO\_DEPRECATED
  - CommonTypes.hh, 919
- GAZEBO\_FORCEINLINE
  - CommonTypes.hh, 919
- GOURAUD
  - gazebo::common::Material, 437
- GPtrArray
  - MeshCSG.hh, 981
- GUIFromSDF
  - Messages, 60
- GUIOverlay
  - gazebo::rendering::GUIOverlay, 338
- GUIOverlay.hh, 948
- GUIPluginPtr
  - gazebo, 81
- GZ\_ALL\_COLLIDE
  - PhysicsTypes.hh, 1008
- GZ\_ASSERT
  - Assert.hh, 900
- GZ\_DBL\_MAX
  - Helpers.hh, 952
- GZ\_DBL\_MIN
  - Helpers.hh, 952
- GZ\_DTOR
  - Angle.hh, 896
- GZ\_FIXED\_COLLIDE
  - PhysicsTypes.hh, 1008
- GZ\_FLT\_MAX
  - Helpers.hh, 952
- GZ\_FLT\_MIN
  - Helpers.hh, 952
- GZ\_GHOST\_COLLIDE
  - PhysicsTypes.hh, 1008
- GZ\_LOG\_VERSION
  - LogRecord.hh, 973
- GZ\_MODEL\_DB\_MANIFEST\_FILENAME
  - ModelDatabase.hh, 986
- GZ\_MODEL\_MANIFEST\_FILENAME
  - ModelDatabase.hh, 986
- GZ\_NONE\_COLLIDE
  - PhysicsTypes.hh, 1008
- GZ\_NORMALIZE
  - Angle.hh, 896
- GZ\_REGISTER\_MODEL\_PLUGIN
  - common/Plugin.hh, 1014
- GZ\_REGISTER\_PHYSICS\_ENGINE
  - Classes for physics and dynamics, 44
- GZ\_REGISTER\_SENSOR\_PLUGIN
  - common/Plugin.hh, 1014
- GZ\_REGISTER\_STATIC\_MSG
  - Messages, 56
- GZ\_REGISTER\_STATIC\_SENSOR
  - Sensors, 70
- GZ\_REGISTER\_SYSTEM\_PLUGIN
  - common/Plugin.hh, 1015
- GZ\_REGISTER\_VISUAL\_PLUGIN
  - common/Plugin.hh, 1015
- GZ\_REGISTER\_WORLD\_PLUGIN
  - common/Plugin.hh, 1015
- GZ\_RTOD
  - Angle.hh, 896
- GZ\_SENSOR\_COLLIDE
  - PhysicsTypes.hh, 1008
- GZ\_VISIBILITY\_ALL
  - RenderTypes.hh, 1031
- GZ\_VISIBILITY\_GUI
  - RenderTypes.hh, 1031
- GZ\_VISIBILITY\_NOT\_SELECTABLE
  - RenderTypes.hh, 1031
- GZ\_VISIBILITY\_SELECTION
  - RenderTypes.hh, 1031
- gazebo, 79
  - add\_plugin, 81
  - find\_file, 81
  - fini, 81
  - GUIPluginPtr, 81
  - init, 81
  - load, 81
  - ModelPluginPtr, 81
  - print\_version, 81
  - run, 81

- SensorPluginPtr, 81
- stop, 81
- SystemPluginPtr, 81
- VisualPluginPtr, 81
- WorldPluginPtr, 81
- gazebo.hh, 942
- gazebo::Master, 433
  - ~Master, 433
  - Fini, 434
  - Init, 434
  - Master, 433
  - Run, 434
  - RunOnce, 434
  - RunThread, 434
  - Stop, 434
- gazebo::ModelPlugin, 483
  - ~ModelPlugin, 484
  - Init, 484
  - Load, 484
  - ModelPlugin, 484
  - Reset, 484
- gazebo::PluginT
  - Create, 571
  - filename, 571
  - GetFilename, 571
  - GetHandle, 571
  - GetType, 571
  - handle, 571
  - TPtr, 570
  - type, 571
- gazebo::PluginT< T >, 570
- gazebo::SensorPlugin, 685
  - ~SensorPlugin, 686
  - Init, 686
  - Load, 686
  - Reset, 686
  - SensorPlugin, 686
- gazebo::Server, 687
  - ~Server, 687
  - Fini, 687
  - GetInitialized, 687
  - Init, 687
  - LoadFile, 687
  - LoadString, 687
  - ParseArgs, 687
  - PrintUsage, 687
  - Run, 687
  - Server, 687
  - SetParams, 687
  - Stop, 687
  - systemPluginsArgc, 688
  - systemPluginsArgv, 688
- gazebo::SystemPlugin, 758
  - ~SystemPlugin, 759
  - Init, 759
  - Load, 759
  - Reset, 759
  - SystemPlugin, 758
- gazebo::VisualPlugin, 869
  - Init, 870
  - Load, 870
  - Reset, 870
  - VisualPlugin, 870
- gazebo::WorldPlugin, 886
  - ~WorldPlugin, 887
  - Init, 887
  - Load, 887
  - Reset, 887
  - WorldPlugin, 887
- gazebo::common, 81
  - AnimationPtr, 84
  - NodeMap, 84
  - NodeMapIter, 84
  - NumericAnimationPtr, 84
  - Param\_V, 84
  - PoseAnimationPtr, 84
  - RawNodeAnim, 84
  - RawNodeWeights, 84
  - RawSkeletonAnim, 84
  - StrStr\_M, 84
- gazebo::common::Animation, 121
  - ~Animation, 123
  - AddTime, 123
  - Animation, 123
  - build, 124
  - GetKeyFrame, 123
  - GetKeyFrameCount, 123
  - GetKeyFramesAtTime, 123
  - GetLength, 124
  - GetTime, 124
  - KeyFrame\_V, 122
  - keyFrames, 124
  - length, 125
  - loop, 125
  - name, 125
  - SetLength, 124
  - SetTime, 124
  - timePos, 125
- gazebo::common::AssertionInternalError, 127
  - ~AssertionInternalError, 129
  - AssertionInternalError, 128
- gazebo::common::BVHLoader, 152
  - ~BVHLoader, 152
  - BVHLoader, 152
  - Load, 152
- gazebo::common::ColladaLoader, 188
  - ~ColladaLoader, 189
  - ColladaLoader, 189



- Load, 189
- gazebo::common::Color, 203
  - ~Color, 206
  - a, 213
  - ABGR, 205
  - ARGB, 205
  - b, 213
  - BGRA, 205
  - Black, 213
  - Blue, 213
  - Color, 205, 206
  - g, 213
  - GetAsABGR, 206
  - GetAsARGB, 206
  - GetAsBGRA, 206
  - GetAsHSV, 207
  - GetAsRGBA, 207
  - GetAsYUV, 207
  - Green, 213
  - operator<<, 213
  - operator>>, 213
  - operator\*, 207, 208
  - operator\*=: 208
  - operator+, 208
  - operator+=: 209
  - operator-, 209
  - operator-=, 209
  - operator/, 210
  - operator/=, 210
  - operator=, 210
  - operator==, 211
  - operator[], 211
  - Purple, 213
  - r, 213
  - RGBA, 205
  - Red, 214
  - Reset, 211
  - Set, 211
  - SetFromABGR, 211
  - SetFromARGB, 212
  - SetFromBGRA, 212
  - SetFromHSV, 212
  - SetFromRGBA, 212
  - SetFromYUV, 212
  - White, 214
  - Yellow, 214
- gazebo::common::Console, 228
- gazebo::common::DiagnosticManager, 254
  - CreateTimer, 255
  - GetEnabled, 255
  - GetLabel, 255
  - GetTime, 255, 256
  - GetTimerCount, 256
  - SetEnabled, 256
  - TimerStart, 256
  - TimerStop, 256
- gazebo::common::DiagnosticTimer, 257
  - ~DiagnosticTimer, 258
  - DiagnosticTimer, 257
  - GetName, 258
- gazebo::common::Exception, 310
  - ~Exception, 311
  - Exception, 311
  - GetErrorFile, 311
  - GetErrorStr, 312
  - operator<<, 312
  - Print, 312
- gazebo::common::Image, 352
  - ~Image, 354
  - BAYER\_GBRG8, 354
  - BAYER\_GRBG8, 354
  - BAYER\_RGGB8, 354
  - BAYER\_RGGR8, 354
  - BGR\_INT16, 353
  - BGR\_INT32, 353
  - BGR\_INT8, 353
  - BGRA\_INT8, 353
  - ConvertPixelFormat, 354
  - GetAvgColor, 354
  - GetBPP, 355
  - GetData, 355
  - GetFilename, 355
  - GetHeight, 355
  - GetMaxColor, 355
  - GetPitch, 355
  - GetPixel, 355
  - GetPixelFormat, 356
  - GetRGBData, 356
  - GetWidth, 356
  - Image, 354
  - L\_INT16, 353
  - L\_INT8, 353
  - Load, 356
  - PIXEL\_FORMAT\_COUNT, 354
  - PixelFormat, 353
  - R\_FLOAT16, 354
  - R\_FLOAT32, 354
  - RGB\_FLOAT16, 354
  - RGB\_FLOAT32, 354
  - RGB\_INT16, 353
  - RGB\_INT32, 353
  - RGB\_INT8, 353
  - RGBA\_INT8, 353
  - Rescale, 356
  - SavePNG, 357
  - SetFromData, 357
  - UNKNOWN\_PIXEL\_FORMAT, 353
  - Valid, 357

- gazebo::common::InternalError, 368
  - ~InternalError, 369
  - InternalError, 369
- gazebo::common::KeyFrame, 394
  - ~KeyFrame, 395
  - GetTime, 395
  - KeyFrame, 395
  - time, 395
- gazebo::common::LogPlay, 427
  - GetChunk, 428
  - GetChunkCount, 428
  - GetEncoding, 428
  - GetGazeboVersion, 428
  - GetLogVersion, 428
  - GetRandSeed, 428
  - IsOpen, 429
  - Open, 429
  - Step, 429
- gazebo::common::LogRecord, 430
  - Add, 431
  - GetEncoding, 431
  - GetRunning, 432
  - Init, 432
  - Remove, 432
  - Start, 432
  - Stop, 432
- gazebo::common::Material, 434
  - ~Material, 437
  - ADD, 437
  - ambient, 443
  - BLEND\_COUNT, 437
  - BLINN, 437
  - BlendMode, 437
  - blendMode, 443
  - BlendModeStr, 443
  - diffuse, 443
  - emissive, 443
  - FLAT, 437
  - GOURAUD, 437
  - GetAmbient, 438
  - GetBlendFactors, 438
  - GetBlendMode, 438
  - GetDepthWrite, 438
  - GetDiffuse, 438
  - GetEmissive, 438
  - GetLighting, 439
  - GetName, 439
  - GetPointSize, 439
  - GetShadeMode, 439
  - GetShininess, 439
  - GetSpecular, 439
  - GetTextureImage, 440
  - GetTransparency, 440
  - MODULATE, 437
  - Material, 437
  - name, 443
  - operator<<, 442
  - PHONG, 437
  - pointSize, 443
  - REPLACE, 437
  - SHADE\_COUNT, 437
  - SetAmbient, 440
  - SetBlendFactors, 440
  - SetBlendMode, 440
  - SetDepthWrite, 441
  - SetDiffuse, 441
  - SetEmissive, 441
  - SetLighting, 441
  - SetPointSize, 441
  - SetShadeMode, 441
  - SetShininess, 441
  - SetSpecular, 442
  - SetTextureImage, 442
  - SetTransparency, 442
  - ShadeMode, 437
  - shadeMode, 443
  - ShadeModeStr, 443
  - shininess, 443
  - specular, 443
  - texImage, 443
  - transparency, 443
- gazebo::common::Mesh, 454
  - ~Mesh, 456
  - AddMaterial, 456
  - AddSubMesh, 456
  - FillArrays, 456
  - GenSphericalTexCoord, 457
  - GetAABB, 457
  - GetIndexCount, 457
  - GetMaterial, 457
  - GetMaterialCount, 457
  - GetMax, 458
  - GetMin, 458
  - GetName, 458
  - GetNormalCount, 458
  - GetPath, 458
  - GetSkeleton, 458
  - GetSubMesh, 459
  - GetSubMeshCount, 459
  - GetTexCoordCount, 459
  - GetVertexCount, 459
  - HasSkeleton, 459
  - Mesh, 456
  - RecalculateNormals, 459
  - Scale, 460
  - SetName, 460
  - SetPath, 460
  - SetScale, 460

- SetSkeleton, 460
- gazebo::common::MeshCSG, 460
  - ~MeshCSG, 461
  - BooleanOperation, 461
  - CreateBoolean, 461
  - DIFFERENCE, 461
  - INTERSECTION, 461
  - MeshCSG, 461
  - UNION, 461
- gazebo::common::MeshLoader, 462
  - ~MeshLoader, 463
  - Load, 463
  - MeshLoader, 463
- gazebo::common::MeshManager, 463
  - AddMesh, 465
  - CreateBox, 465
  - CreateCamera, 465
  - CreateCone, 465
  - CreateCylinder, 465
  - CreatePlane, 466
  - CreateSphere, 466
  - CreateTube, 466
  - GenSphericalTexCoord, 467
  - GetMesh, 467
  - GetMeshAABB, 467
  - HasMesh, 467
  - IsValidFilename, 467
  - Load, 468
- gazebo::common::ModelDatabase, 482
- gazebo::common::MouseEvent, 491
  - alt, 493
  - button, 493
  - Buttons, 493
  - buttons, 493
  - control, 493
  - dragging, 493
  - EventType, 493
  - LEFT, 493
  - MIDDLE, 493
  - MOVE, 493
  - MouseEvent, 493
  - moveScale, 494
  - NO\_BUTTON, 493
  - NO\_EVENT, 493
  - PRESS, 493
  - pos, 494
  - pressPos, 494
  - prevPos, 494
  - RELEASE, 493
  - RIGHT, 493
  - SCROLL, 493
  - scroll, 494
  - shift, 494
  - type, 494
- gazebo::common::NodeAnimation, 520
  - ~NodeAnimation, 521
  - AddKeyFrame, 521, 522
  - GetFrameAt, 522
  - GetFrameCount, 522
  - GetKeyFrame, 522
  - GetLength, 523
  - GetName, 523
  - GetTimeAtX, 523
  - keyFrames, 524
  - length, 524
  - name, 524
  - NodeAnimation, 521
  - Scale, 523
  - SetName, 523
- gazebo::common::NodeAssignment, 524
  - nodeIndex, 525
  - vertexIndex, 525
  - weight, 525
- gazebo::common::NodeTransform, 525
  - ~NodeTransform, 527
  - Get, 527
  - GetSID, 527
  - GetType, 527
  - MATRIX, 526
  - NodeTransform, 527
  - operator\*, 528
  - operator(), 527
  - PrintSource, 528
  - ROTATE, 526
  - RecalculateMatrix, 528
  - SCALE, 526
  - Set, 528
  - SetComponent, 528
  - SetSID, 529
  - SetSourceValues, 529
  - SetType, 529
  - sid, 529
  - source, 529
  - TRANSLATE, 526
  - transform, 530
  - TransformType, 526
  - type, 530
- gazebo::common::NumericAnimation, 530
  - ~NumericAnimation, 531
  - CreateKeyFrame, 531
  - GetInterpolatedKeyFrame, 531
  - NumericAnimation, 531
- gazebo::common::NumericKeyFrame, 532
  - ~NumericKeyFrame, 533
  - GetValue, 533
  - NumericKeyFrame, 532
  - SetValue, 533
  - value, 533

- gazebo::common::PID, 558
  - ~PID, 559
  - GetCmd, 560
  - GetErrors, 560
  - Init, 560
  - operator=, 560
  - PID, 559
  - Reset, 560
  - SetCmd, 561
  - SetCmdMax, 561
  - SetCmdMin, 561
  - SetDGain, 561
  - SetIGain, 561
  - SetIMax, 561
  - SetIMin, 562
  - SetPGain, 562
  - Update, 562
- gazebo::common::PoseAnimation, 580
  - ~PoseAnimation, 582
  - BuildInterpolationSplines, 582
  - CreateKeyFrame, 582
  - GetInterpolatedKeyFrame, 582
  - PoseAnimation, 581
- gazebo::common::PoseKeyFrame, 583
  - ~PoseKeyFrame, 584
  - GetRotation, 584
  - GetTranslation, 584
  - PoseKeyFrame, 584
  - rotate, 585
  - SetRotation, 584
  - SetTranslation, 584
  - translate, 585
- gazebo::common::STLLoader, 732
  - ~STLLoader, 733
  - Load, 733
  - STLLoader, 733
- gazebo::common::Skeleton, 697
  - ~Skeleton, 699
  - AddAnimation, 699
  - AddVertNodeWeight, 699
  - anims, 703
  - bindShapeTransform, 703
  - BuildNodeMap, 700
  - GetAnimation, 700
  - GetBindShapeTransform, 700
  - GetNodeByHandle, 700
  - GetNodeById, 700
  - GetNodeByName, 701
  - GetNodes, 701
  - GetNumAnimations, 701
  - GetNumJoints, 701
  - GetNumNodes, 701
  - GetNumVertNodeWeights, 701
  - GetRootNode, 702
  - GetVertNodeWeight, 702
  - nodes, 703
  - PrintTransforms, 702
  - rawNW, 703
  - root, 703
  - Scale, 702
  - SetBindShapeTransform, 702
  - SetNumVertAttached, 703
  - SetRootNode, 703
  - Skeleton, 699
- gazebo::common::SkeletonAnimation, 704
  - ~SkeletonAnimation, 705
  - AddKeyFrame, 705
  - animations, 707
  - GetLength, 705
  - GetName, 706
  - GetNodeCount, 706
  - GetNodePoseAt, 706
  - GetPoseAt, 706
  - GetPoseAtX, 707
  - HasNode, 707
  - length, 707
  - name, 708
  - Scale, 707
  - SetName, 707
  - SkeletonAnimation, 705
- gazebo::common::SkeletonNode, 708
  - ~SkeletonNode, 711
  - AddChild, 711
  - AddRawTransform, 711
  - children, 716
  - GetChild, 711
  - GetChildById, 711
  - GetChildByName, 712
  - GetChildCount, 712
  - GetHandle, 712
  - GetId, 712
  - GetInverseBindTransform, 712
  - GetModelTransform, 712
  - GetName, 713
  - GetNumRawTrans, 713
  - GetParent, 713
  - GetRawTransform, 713
  - GetRawTransforms, 713
  - GetTransform, 714
  - GetTransforms, 714
  - handle, 716
  - id, 716
  - initialTransform, 716
  - invBindTransform, 716
  - IsJoint, 714
  - IsRootNode, 714
  - JOINT, 710
  - modelTransform, 716

- NODE, 710
- name, 716
- parent, 717
- rawTransforms, 717
- Reset, 714
- SetHandle, 714
- SetId, 714
- SetInitialTransform, 715
- SetInverseBindTransform, 715
- SetModelTransform, 715
- SetName, 715
- SetParent, 715
- SetTransform, 715
- SetType, 716
- SkeletonNode, 710, 711
- SkeletonNodeType, 710
- transform, 717
- type, 717
- UpdateChildrenTransforms, 716
- gazebo::common::SubMesh, 734
  - ~SubMesh, 736
  - AddIndex, 736
  - AddNodeAssignment, 737
  - AddNormal, 737
  - AddTexCoord, 737
  - AddVertex, 737
  - CopyNormals, 738
  - CopyVertices, 738
  - FillArrays, 738
  - GenSphericalTexCoord, 738
  - GetIndex, 738
  - GetIndexCount, 739
  - GetMaterialIndex, 739
  - GetMax, 739
  - GetMaxIndex, 739
  - GetMin, 739
  - GetNodeAssignment, 739
  - GetNodeAssignmentsCount, 739
  - GetNormal, 739
  - GetNormalCount, 740
  - GetPrimitiveType, 740
  - GetTexCoord, 740
  - GetTexCoordCount, 740
  - GetVertex, 740
  - GetVertexCount, 740
  - GetVertexIndex, 741
  - HasVertex, 741
- LINES, 736
- LINESTRIPS, 736
- POINTS, 736
- PrimitiveType, 736
- RecalculateNormals, 741
- Scale, 741
- SetIndexCount, 741
- SetMaterialIndex, 741
- SetNormal, 742
- SetNormalCount, 742
- SetPrimitiveType, 742
- SetScale, 742
- SetSubMeshCenter, 742
- SetTexCoord, 742
- SetTexCoordCount, 743
- SetVertex, 743
- SetVertexCount, 743
- SubMesh, 736
- TRIANGLES, 736
- TRIFANS, 736
- TRISTRIPS, 736
- gazebo::common::SystemPaths, 753
  - AddGazeboPaths, 755
  - AddOgrePaths, 755
  - AddPluginPaths, 755
  - AddSearchPathSuffix, 755
  - ClearGazeboPaths, 755
  - ClearOgrePaths, 755
  - ClearPluginPaths, 755
  - FindFile, 755
  - FindFileURI, 756
  - gazeboPathsFromEnv, 757
  - GetGazeboPaths, 756
  - GetLogPath, 756
  - GetModelPaths, 756
  - GetOgrePaths, 756
  - GetPluginPaths, 757
  - GetWorldPathExtension, 757
  - modelPathsFromEnv, 757
  - ogrePathsFromEnv, 757
  - pluginPathsFromEnv, 757
- gazebo::common::Time, 759
  - ~Time, 764
  - Double, 764
  - Float, 764
  - GetWallTime, 764
  - MSleep, 765
  - MicToNano, 764
  - MilToNano, 765
  - NSleep, 765
  - nsec, 779
  - operator<, 772, 773
  - operator<<, 779
  - operator<=, 773, 774
  - operator>, 776, 777
  - operator>>, 779
  - operator>=, 777, 778
  - operator\*, 766, 767
  - operator\*=, 767, 768
  - operator+, 768
  - operator+=, 769

- operator-, 769, 770
- operator=, 770, 771
- operator/, 771
- operator/=, 772
- operator=, 774, 775
- operator==, 775, 776
- sec, 779
- SecToNano, 778
- Set, 778, 779
- SetToWallTime, 779
- Time, 763, 764
- gazebo::common::Timer, 780
  - ~Timer, 781
  - GetElapsed, 781
  - operator<<, 781
  - Start, 781
  - Timer, 781
- gazebo::common::Video, 843
  - ~Video, 843
  - GetHeight, 844
  - GetNextFrame, 844
  - GetWidth, 844
  - Load, 844
  - Video, 843
- gazebo::event, 84
  - Connection\_V, 85
  - ConnectionPtr, 85
- gazebo::event::Connection, 216
  - ~Connection, 216
  - Connection, 216
  - GetId, 217
- gazebo::event::Event, 285
  - ~Event, 287
  - Disconnect, 287
- gazebo::event::EventT
  - operator(), 304–306
  - Signal, 307–309
- gazebo::event::EventT< T >, 302
- gazebo::event::Events, 290
  - addEntity, 300
  - ConnectAddEntity, 292
  - ConnectCreateEntity, 292
  - ConnectDeleteEntity, 293
  - ConnectDiagTimerStart, 293
  - ConnectDiagTimerStop, 293
  - ConnectPause, 294
  - ConnectPostRender, 294
  - ConnectPreRender, 294
  - ConnectRender, 295
  - ConnectSetSelectedEntity, 295
  - ConnectStep, 295
  - ConnectStop, 295
  - ConnectWorldCreated, 296
  - ConnectWorldUpdateEnd, 296
  - ConnectWorldUpdateStart, 296
  - deleteEntity, 300
  - diagTimerStart, 300
  - diagTimerStop, 300
  - DisconnectAddEntity, 297
  - DisconnectCreateEntity, 297
  - DisconnectDeleteEntity, 297
  - DisconnectDiagTimerStart, 297
  - DisconnectDiagTimerStop, 297
  - DisconnectPause, 298
  - DisconnectPostRender, 298
  - DisconnectPreRender, 298
  - DisconnectRender, 298
  - DisconnectSetSelectedEntity, 298
  - DisconnectStep, 299
  - DisconnectStop, 299
  - DisconnectWorldCreated, 299
  - DisconnectWorldUpdateEnd, 299
  - DisconnectWorldUpdateStart, 299
  - entityCreated, 300
  - pause, 300
  - postRender, 300
  - preRender, 300
  - render, 301
  - setSelectedEntity, 301
  - step, 301
  - stop, 301
  - worldCreated, 301
  - worldUpdateEnd, 301
  - worldUpdateStart, 301
- gazebo::math, 85
  - GeneratorType, 87
  - NRealGen, 87
  - NormalRealDist, 87
  - UIntGen, 87
  - URealGen, 87
  - UniformIntDist, 87
  - UniformRealDist, 87
- gazebo::math::Angle, 114
  - ~Angle, 116
  - Angle, 115
  - Degree, 116
  - Normalize, 116
  - operator<, 118
  - operator<<, 120
  - operator<=, 119
  - operator>, 119
  - operator>>, 120
  - operator>=, 119
  - operator\*, 116
  - operator\*=, 117
  - operator+, 117
  - operator+=, 117
  - operator-, 117

- operator=, 118
- operator/, 118
- operator/=, 118
- operator==, 119
- Radian, 120
- SetFromDegree, 120
- SetFromRadian, 120
- gazebo::math::Box, 144
  - ~Box, 146
  - Box, 146
  - GetCenter, 146
  - GetSize, 146
  - GetXLength, 146
  - GetYLength, 147
  - GetZLength, 147
  - max, 149
  - Merge, 147
  - min, 149
  - operator<<, 148
  - operator+, 147
  - operator+=", 147
  - operator-, 148
  - operator=, 148
  - operator==, 148
- gazebo::math::Matrix3, 444
  - ~Matrix3, 445
  - m, 447
  - Matrix3, 445
  - operator<<, 447
  - operator==, 445
  - operator[], 446
  - SetCol, 446
  - SetFromAxes, 446
  - SetFromAxis, 446
- gazebo::math::Matrix4, 447
  - ~Matrix4, 450
  - GetAsPose, 450
  - GetEulerRotation, 450
  - GetRotation, 450
  - GetTranslation, 450
  - IDENTITY, 454
  - Inverse, 450
  - IsAffine, 450
  - m, 454
  - Matrix4, 449
  - operator<<, 454
  - operator\*, 451
  - operator=, 451, 452
  - operator==, 452
  - operator[], 452
  - Set, 453
  - SetScale, 453
  - SetTranslate, 453
  - TransformAffine, 453
- ZERO, 454
- gazebo::math::Plane, 562
  - ~Plane, 564
  - d, 564
  - Distance, 564
  - normal, 565
  - operator=, 564
  - Plane, 563
  - Set, 564
  - size, 565
- gazebo::math::Pose, 572
  - ~Pose, 574
  - CoordPoseSolve, 574
  - CoordPositionAdd, 575
  - CoordPositionSub, 575
  - CoordRotationAdd, 575
  - CoordRotationSub, 576
  - Correct, 576
  - GetInverse, 576
  - IsFinite, 576
  - operator<<, 579
  - operator>>, 579
  - operator\*, 577
  - operator+, 577
  - operator+=", 577
  - operator-, 577
  - operator=, 578
  - operator==, 578
  - pos, 580
  - Pose, 574
  - Reset, 578
  - rot, 580
  - RotatePositionAboutOrigin, 578
  - Round, 579
  - Set, 579
  - Zero, 580
- gazebo::math::Quaternion, 597
  - ~Quaternion, 601
  - Correct, 601
  - Dot, 601
  - EulerToQuaternion, 602
  - GetAsAxis, 602
  - GetAsEuler, 602
  - GetAsMatrix3, 602
  - GetAsMatrix4, 603
  - GetExp, 603
  - GetInverse, 603
  - GetLog, 603
  - GetPitch, 603
  - GetRoll, 603
  - GetXAxis, 604
  - GetYAxis, 604
  - GetYaw, 604
  - GetZAxis, 604

- Invert, 604
- IsFinite, 604
- Normalize, 604
- operator<<, 610
- operator>>, 610
- operator\*, 605
- operator\*=: 606
- operator+, 606
- operator+=, 606
- operator-, 606
- operator-=, 607
- operator=, 607
- operator==, 607
- Quaternion, 600, 601
- RotateVector, 607
- RotateVectorReverse, 608
- Round, 608
- Scale, 608
- Set, 608
- SetFromAxis, 608, 609
- SetFromEuler, 609
- SetToIdentity, 609
- Slerp, 609
- Squad, 610
- w, 610
- x, 610
- y, 611
- z, 611
- gazebo::math::Rand, 611
  - GetDbfNormal, 612
  - GetDbfUniform, 612
  - GetIntNormal, 612
  - GetIntUniform, 612
  - GetSeed, 612
  - SetSeed, 612
- gazebo::math::RotationSpline, 642
  - ~RotationSpline, 643
  - AddPoint, 643
  - autoCalc, 645
  - Clear, 643
  - GetNumPoints, 643
  - GetPoint, 643
  - Interpolate, 644
  - points, 645
  - RecalcTangents, 644
  - RotationSpline, 643
  - SetAutoCalculate, 645
  - tangents, 645
  - UpdatePoint, 645
- gazebo::math::Spline, 724
  - ~Spline, 725
  - AddPoint, 725
  - autoCalc, 728
  - Clear, 725
  - coeffs, 728
  - GetPoint, 725
  - GetPointCount, 726
  - GetTangent, 726
  - GetTension, 726
  - Interpolate, 726
  - points, 728
  - RecalcTangents, 726
  - SetAutoCalculate, 727
  - SetTension, 727
  - Spline, 725
  - tangents, 728
  - tension, 728
  - UpdatePoint, 727
- gazebo::math::Vector2d, 803
  - ~Vector2d, 805
  - Cross, 805
  - Distance, 805
  - IsFinite, 806
  - Normalize, 806
  - operator<<, 810
  - operator>>, 811
  - operator\*, 806
  - operator\*=: 807
  - operator+, 807
  - operator+=, 807
  - operator-, 808
  - operator-=, 808
  - operator/, 808
  - operator/=, 809
  - operator=, 809
  - operator==, 810
  - operator[], 810
  - Set, 810
  - Vector2d, 805
  - x, 811
  - y, 811
- gazebo::math::Vector2i, 811
  - ~Vector2i, 813
  - Cross, 814
  - Distance, 814
  - IsFinite, 814
  - Normalize, 814
  - operator<<, 819
  - operator>>, 819
  - operator\*, 815
  - operator\*=: 815
  - operator+, 816
  - operator+=, 816
  - operator-, 816
  - operator-=, 816
  - operator/, 817
  - operator/=, 817, 818
  - operator=, 818



- operator==, 819
- operator[], 819
- Set, 819
- Vector2i, 813
- x, 820
- y, 820
- gazebo::math::Vector3, 820
  - ~Vector3, 823
  - Correct, 823
  - Cross, 824
  - Distance, 824
  - Dot, 824
  - Equal, 824
  - GetAbs, 825
  - GetDistToLine, 825
  - GetLength, 825
  - GetMax, 825
  - GetMin, 825
  - GetNormal, 826
  - GetPerpendicular, 826
  - GetRounded, 826
  - GetSquaredLength, 826
  - GetSum, 826
  - IsFinite, 827
  - Normalize, 827
  - operator<<, 832
  - operator>>, 833
  - operator\*, 827, 832
  - operator\*=: 828
  - operator+, 828
  - operator+=, 828
  - operator-, 829
  - operator-=, 829
  - operator/, 829, 830
  - operator/=, 830
  - operator=, 830, 831
  - operator==, 831
  - operator[], 831
  - Round, 831
  - Set, 831
  - SetToMax, 832
  - SetToMin, 832
  - Vector3, 823
  - x, 833
  - y, 833
  - z, 833
  - Zero, 833
- gazebo::math::Vector4, 833
  - ~Vector4, 836
  - Distance, 836
  - GetLength, 836
  - GetSquaredLength, 836
  - IsFinite, 836
  - Normalize, 837
  - operator<<, 842
  - operator>>, 842
  - operator\*, 837, 838
  - operator\*=: 838
  - operator+, 838
  - operator+=, 839
  - operator-, 839
  - operator-=, 839
  - operator/, 839, 840
  - operator/=, 840
  - operator=, 841
  - operator==, 841
  - operator[], 841
  - Set, 841
  - Vector4, 835, 836
  - w, 842
  - x, 842
  - y, 842
  - z, 843
- gazebo::msgs, 87
  - MsgFactoryFn, 89
- gazebo::msgs::MsgFactory, 500
  - GetMsgTypes, 501
  - NewMsg, 501
  - RegisterMsg, 501
- gazebo::physics, 89
  - Actor\_V, 93
  - ActorPtr, 93
  - Base\_V, 93
  - BasePtr, 93
  - BoxShapePtr, 93
  - Collision\_V, 93
  - CollisionPtr, 93
  - ContactPtr, 93
  - CylinderShapePtr, 93
  - EntityPtr, 93
  - HeightmapShapePtr, 93
  - InertialPtr, 93
  - Joint\_V, 93
  - JointController\_V, 93
  - JointControllerPtr, 93
  - JointPtr, 93
  - Link\_V, 93
  - LinkPtr, 93
  - MeshShapePtr, 93
  - Model\_V, 93
  - ModelPtr, 94
  - MultiRayShapePtr, 94
  - PhysicsEnginePtr, 94
  - RayShapePtr, 94
  - RoadPtr, 94
  - ShapePtr, 94
  - SphereShapePtr, 94
  - SurfaceParamsPtr, 94

- WorldPtr, 94
- gazebo::physics::Actor, 107
  - ~Actor, 110
  - active, 112
  - Actor, 110
  - autoStart, 112
  - bonePosePub, 112
  - Fini, 110
  - GetSDF, 110
  - Init, 111
  - interpolateX, 112
  - isActive, 111
  - lastPos, 112
  - lastScriptTime, 112
  - lastTraj, 112
  - Load, 111
  - loop, 112
  - mainLink, 112
  - mesh, 112
  - oldAction, 112
  - pathLength, 112
  - Play, 111
  - playStartTime, 113
  - prevFrameTime, 113
  - scriptLength, 113
  - skelAnimation, 113
  - skelNodesMap, 113
  - skeleton, 113
  - skinFile, 113
  - skinScale, 113
  - startDelay, 113
  - Stop, 111
  - trajInfo, 113
  - trajectories, 113
  - Update, 111
  - UpdateParameters, 111
  - visualName, 114
- gazebo::physics::BallJoint
  - ~BallJoint, 133
  - BallJoint, 132
  - GetAngleCount, 133
  - GetHighStop, 133
  - GetLowStop, 133
  - Load, 133
  - SetAxis, 133
  - SetHighStop, 133
  - SetLowStop, 133
- gazebo::physics::BallJoint< T >, 131
- gazebo::physics::Base, 133
  - ~Base, 137
  - ACTOR, 136
  - AddChild, 137
  - AddType, 137
  - BALL\_JOINT, 137
  - BASE, 136
  - BOX\_SHAPE, 137
  - Base, 137
  - COLLISION, 136
  - CYLINDER\_SHAPE, 137
  - children, 144
  - childrenEnd, 144
  - ENTITY, 136
  - EntityType, 136
  - Fini, 138
  - GetById, 138
  - GetByName, 138
  - GetChild, 138
  - GetChildCount, 139
  - GetId, 139
  - GetName, 139
  - GetParent, 139
  - GetParentId, 139
  - GetSDF, 140
  - GetSaveable, 139
  - GetScopedName, 140
  - GetType, 140
  - GetWorld, 140
  - HEIGHTMAP\_SHAPE, 137
  - HINGE2\_JOINT, 137
  - HINGE\_JOINT, 137
  - HasType, 140
  - Init, 141
  - IsSelected, 141
  - JOINT, 137
  - LIGHT, 137
  - LINK, 136
  - Load, 141
  - MAP\_SHAPE, 137
  - MODEL, 136
  - MULTIRAY\_SHAPE, 137
  - operator==, 141
  - PLANE\_SHAPE, 137
  - parent, 144
  - Print, 142
  - RAY\_SHAPE, 137
  - RemoveChild, 142
  - RemoveChildren, 142
  - Reset, 142
  - SCREW\_JOINT, 137
  - SHAPE, 137
  - SLIDER\_JOINT, 137
  - SPHERE\_SHAPE, 137
  - sdf, 144
  - SetName, 142
  - SetParent, 143
  - SetSaveable, 143
  - SetSelected, 143
  - SetWorld, 143

- TRIMESH\_SHAPE, 137
- UNIVERSAL\_JOINT, 137
- Update, 143
- UpdateParameters, 144
- VISUAL, 137
- world, 144
- gazebo::physics::BoxShape, 149
  - ~BoxShape, 151
  - BoxShape, 151
  - FillMsg, 151
  - GetSize, 151
  - Init, 151
  - ProcessMsg, 151
  - SetSize, 151
- gazebo::physics::Collision, 190
  - ~Collision, 192
  - AddContact, 193
  - Collision, 192
  - ConnectContact, 193
  - DisconnectContact, 193
  - FillMsg, 193
  - Fin, 193
  - GetBoundingBox, 193
  - GetContactsEnabled, 193
  - GetLaserRetro, 194
  - GetLink, 194
  - GetModel, 194
  - GetRelativeAngularAccel, 194
  - GetRelativeAngularVel, 194
  - GetRelativeLinearAccel, 194
  - GetRelativeLinearVel, 195
  - GetShape, 195
  - GetShapeType, 195
  - GetState, 195
  - GetSurface, 195
  - GetWorldAngularAccel, 196
  - GetWorldAngularVel, 196
  - GetWorldLinearAccel, 196
  - GetWorldLinearVel, 196
  - Init, 196
  - IsPlaceable, 196
  - link, 198
  - Load, 197
  - placeable, 199
  - ProcessMsg, 197
  - SetCategoryBits, 197
  - SetCollideBits, 197
  - SetCollision, 197
  - SetContactsEnabled, 198
  - SetLaserRetro, 198
  - SetShape, 198
  - SetState, 198
  - shape, 199
  - UpdateParameters, 198
- gazebo::physics::CollisionState, 199
  - ~CollisionState, 201
  - CollisionState, 200
  - FillSDF, 201
  - GetPose, 201
  - IsZero, 201
  - Load, 201
  - operator<<, 202
  - operator+, 201
  - operator-, 202
  - operator=, 202
- gazebo::physics::Contact, 229
  - ~Contact, 230
  - collision1, 231
  - collision2, 231
  - Contact, 230
  - count, 231
  - DebugString, 230
  - depths, 231
  - FillMsg, 230
  - normals, 231
  - operator=, 230, 231
  - positions, 231
  - Reset, 231
  - time, 232
  - wrench, 232
- gazebo::physics::ContactManager, 232
  - ~ContactManager, 233
  - Clear, 233
  - ContactManager, 233
  - GetContact, 233
  - GetContactCount, 233
  - GetContacts, 233
  - Init, 233
  - NewContact, 234
  - PublishContacts, 234
  - ResetCount, 234
- gazebo::physics::CylinderShape, 243
  - ~CylinderShape, 244
  - CylinderShape, 244
  - FillMsg, 244
  - GetLength, 244
  - GetRadius, 244
  - Init, 245
  - ProcessMsg, 245
  - SetLength, 245
  - SetRadius, 245
  - SetSize, 245
- gazebo::physics::Entity, 274
  - ~Entity, 277
  - animation, 284
  - animationConnection, 284
  - animationStartPose, 284
  - connections, 284

- dirtyPose, 285
- Entity, 277
- Fini, 277
- GetBoundingBox, 277
- GetChildCollision, 277
- GetChildLink, 278
- GetCollisionBoundingBox, 278
- GetDirtyPose, 278
- GetInitialRelativePose, 278
- GetNearestEntityBelow, 278
- GetParentModel, 279
- GetRelativeAngularAccel, 279
- GetRelativeAngularVel, 279
- GetRelativeLinearAccel, 279
- GetRelativeLinearVel, 279
- GetRelativePose, 280
- GetWorldAngularAccel, 280
- GetWorldAngularVel, 280
- GetWorldLinearAccel, 280
- GetWorldLinearVel, 280
- GetWorldPose, 281
- IsCanonicalLink, 281
- IsStatic, 281
- Load, 281
- node, 285
- OnPoseChange, 281
- parentEntity, 285
- PlaceOnEntity, 282
- PlaceOnNearestEntityBelow, 282
- poseMsg, 285
- prevAnimationTime, 285
- requestPub, 285
- Reset, 282
- SetAnimation, 282
- SetCanonicalLink, 282
- SetInitialRelativePose, 283
- SetName, 283
- SetRelativePose, 283
- SetStatic, 283
- SetWorldPose, 283
- SetWorldTwist, 284
- StopAnimation, 284
- UpdateParameters, 284
- visPub, 285
- visualMsg, 285
- gazebo::physics::Gripper, 336
  - ~Gripper, 336
  - Gripper, 336
  - Init, 337
  - Load, 337
- gazebo::physics::HeightmapShape, 344
  - ~HeightmapShape, 346
  - FillMsg, 346
  - GetHeight, 346
  - GetMaxHeight, 346
  - GetMinHeight, 346
  - GetPos, 346
  - GetSize, 347
  - GetSubSampling, 347
  - GetURI, 347
  - GetVertexCount, 347
  - HeightmapShape, 345
  - heights, 348
  - img, 348
  - Init, 347
  - Load, 347
  - ProcessMsg, 348
  - scale, 348
  - subSampling, 348
  - vertSize, 348
- gazebo::physics::Hinge2Joint
  - ~Hinge2Joint, 349
  - GetAngleCount, 350
  - Hinge2Joint, 349
  - Load, 350
- gazebo::physics::Hinge2Joint< T >, 348
- gazebo::physics::HingeJoint
  - ~HingeJoint, 351
  - GetAngleCount, 351
  - HingeJoint, 351
  - Init, 351
  - Load, 351
- gazebo::physics::HingeJoint< T >, 350
- gazebo::physics::Inertial, 360
  - ~Inertial, 363
  - GetCoG, 363
  - GetlXX, 363
  - GetlXY, 363
  - GetlXZ, 363
  - GetlYY, 363
  - GetlYZ, 363
  - GetlZZ, 364
  - GetMass, 364
  - GetPose, 364
  - GetPrincipalMoments, 364
  - GetProductsofInertia, 364
  - Inertial, 362
  - Load, 364
  - operator<<, 368
  - operator+, 365
  - operator+=", 365
  - operator=, 365
  - ProcessMsg, 365
  - Reset, 365
  - Rotate, 366
  - SetCoG, 366
  - SetlXX, 366
  - SetlXY, 367

- SetIXZ, 367
- SetIYY, 367
- SetIYZ, 367
- SetIZZ, 367
- SetInertiaMatrix, 366
- SetMass, 367
- UpdateParameters, 367
- gazebo::physics::Joint, 371
  - ~Joint, 375
  - anchorLink, 384
  - anchorPos, 384
  - ApplyDamping, 375
  - applyDamping, 384
  - AreConnected, 375
  - Attach, 375
  - Attribute, 374
  - CFM, 374
  - childLink, 384
  - ConnectJointUpdate, 375
  - dampingCoefficient, 384
  - Detach, 376
  - DisconnectJointUpdate, 376
  - ERP, 374
  - FMAX, 374
  - FUDGE\_FACTOR, 374
  - FillMsg, 376
  - forceApplied, 385
  - GetAnchor, 376
  - GetAngle, 376
  - GetAngleCount, 377
  - GetAngleImpl, 377
  - GetChild, 377
  - GetForce, 377
  - GetForceTorque, 378
  - GetGlobalAxis, 378
  - GetHighStop, 378
  - GetJointLink, 378
  - GetLinkForce, 379
  - GetLinkTorque, 379
  - GetLocalAxis, 379
  - GetLowStop, 379
  - GetMaxForce, 380
  - GetParent, 380
  - GetVelocity, 380
  - HI\_STOP, 374
  - Init, 380
  - Joint, 375
  - LO\_STOP, 374
  - Load, 380, 381
  - model, 385
  - parentLink, 385
  - Reset, 381
  - STOP\_CFM, 374
  - STOP\_ERP, 374
  - SUSPENSION\_CFM, 374
  - SUSPENSION\_ERP, 374
  - SetAnchor, 381
  - SetAngle, 381
  - SetAttribute, 382
  - SetAxis, 382
  - SetDamping, 382
  - SetForce, 382
  - SetHighStop, 383
  - SetLowStop, 383
  - SetMaxForce, 383
  - SetModel, 383
  - SetState, 383
  - SetVelocity, 384
  - Update, 384
  - UpdateParameters, 384
  - VEL, 374
- gazebo::physics::JointController, 385
  - AddJoint, 386
  - JointController, 386
  - Reset, 386
  - SetJointPosition, 386
  - SetJointPositions, 386
  - Update, 387
- gazebo::physics::JointState, 387
  - ~JointState, 389
  - FillSDF, 389
  - GetAngle, 389
  - GetAngleCount, 389
  - GetAngles, 389
  - IsZero, 389
  - JointState, 388
  - Load, 390
  - operator<<, 391
  - operator+, 390
  - operator-, 390
  - operator=, 390
- gazebo::physics::JointWrench, 392
  - body1Force, 393
  - body1Torque, 393
  - body2Force, 394
  - body2Torque, 394
  - operator=, 393
- gazebo::physics::Link, 403
  - ~Link, 408
  - AddChildJoint, 408
  - AddForce, 408
  - AddForceAtRelativePosition, 408
  - AddForceAtWorldPosition, 408
  - AddParentJoint, 409
  - AddRelativeForce, 409
  - AddRelativeTorque, 409
  - AddTorque, 409
  - angularAccel, 420

AttachStaticModel, 409  
 attachedModelsOffset, 420  
 cgVisuals, 420  
 ConnectEnabled, 410  
 DetachAllStaticModels, 410  
 DetachStaticModel, 410  
 DisconnectEnabled, 410  
 FillMsg, 410  
 Fini, 410  
 GetAngularDamping, 411  
 GetBoundingBox, 411  
 GetChildJointsLinks, 411  
 GetCollision, 411  
 GetCollisionById, 412  
 GetCollisions, 412  
 GetEnabled, 412  
 GetGravityMode, 412  
 GetInertial, 412  
 GetKinematic, 412  
 GetLinearDamping, 413  
 GetModel, 413  
 GetParentJointsLinks, 413  
 GetRelativeAngularAccel, 413  
 GetRelativeAngularVel, 413  
 GetRelativeForce, 413  
 GetRelativeLinearAccel, 414  
 GetRelativeLinearVel, 414  
 GetRelativeTorque, 414  
 GetSelfCollide, 414  
 GetSensorCount, 414  
 GetSensorName, 414  
 GetWorldAngularAccel, 415  
 GetWorldForce, 415  
 GetWorldLinearAccel, 415  
 GetWorldTorque, 415  
 inertial, 421  
 Init, 416  
 linearAccel, 421  
 Link, 408  
 Load, 416  
 OnPoseChange, 416  
 ProcessMsg, 416  
 RemoveChildJoint, 416  
 RemoveParentJoint, 416  
 Reset, 417  
 SetAngularAccel, 417  
 SetAngularDamping, 417  
 SetAngularVel, 417  
 SetAutoDisable, 417  
 SetCollideMode, 417  
 SetEnabled, 418  
 SetForce, 418  
 SetGravityMode, 418  
 SetInertial, 418  
 SetKinematic, 418  
 SetLaserRetro, 418  
 SetLinearAccel, 419  
 SetLinearDamping, 419  
 SetLinearVel, 419  
 SetSelected, 419  
 SetSelfCollide, 419  
 SetState, 419  
 SetTorque, 420  
 Update, 420  
 UpdateMass, 420  
 UpdateParameters, 420  
 UpdateSurface, 420  
 visuals, 421  
 gazebo::physics::LinkState, 421  
   ~LinkState, 423  
   FillSDF, 423  
   GetAcceleration, 423  
   GetCollisionState, 423, 424  
   GetCollisionStateCount, 424  
   GetCollisionStates, 424  
   GetPose, 425  
   GetVelocity, 425  
   GetWrench, 425  
   IsZero, 425  
   LinkState, 423  
   Load, 425  
   operator<<, 426  
   operator+, 425  
   operator-, 426  
   operator=, 426  
 gazebo::physics::Model, 468  
   ~Model, 472  
   AttachStaticModel, 472  
   attachedModels, 481  
   attachedModelsOffset, 481  
   DetachStaticModel, 472  
   FillMsg, 473  
   Fini, 473  
   GetAutoDisable, 473  
   GetBoundingBox, 473  
   GetJoint, 473  
   GetJointController, 474  
   GetJointCount, 474  
   GetJoints, 474  
   GetLink, 474  
   GetLinkById, 474  
   GetLinks, 474  
   GetPluginCount, 475  
   GetRelativeAngularAccel, 475  
   GetRelativeAngularVel, 475  
   GetRelativeLinearAccel, 475  
   GetRelativeLinearVel, 475  
   GetSDF, 476

- GetSensorCount, 476
- GetWorldAngularAccel, 476
- GetWorldAngularVel, 476
- GetWorldLinearAccel, 476
- GetWorldLinearVel, 477
- Init, 477
- Load, 477
- LoadJoints, 477
- LoadPlugins, 477
- Model, 472
- OnPoseChange, 477
- ProcessMsg, 478
- RemoveChild, 478
- Reset, 478
- SetAngularAccel, 478
- SetAngularVel, 478
- SetAutoDisable, 478
- SetCollideMode, 478
- SetEnabled, 479
- SetGravityMode, 479
- SetJointAnimation, 479
- SetJointPosition, 479
- SetJointPositions, 479
- SetLaserRetro, 480
- SetLinearAccel, 480
- SetLinearVel, 480
- SetLinkWorldPose, 480
- SetState, 481
- StopAnimation, 481
- Update, 481
- UpdateParameters, 481
- gazebo::physics::ModelState, 485
  - ~ModelState, 487
  - FillSDF, 487
  - GetJointState, 487
  - GetJointStateCount, 488
  - GetJointStates, 488
  - GetLinkState, 488
  - GetLinkStateCount, 489
  - GetLinkStates, 489
  - GetPose, 489
  - HasJointState, 489
  - HasLinkState, 490
  - IsZero, 490
  - Load, 490
  - ModelState, 486
  - operator<<, 491
  - operator+, 490
  - operator-, 490
  - operator=, 491
- gazebo::physics::MultiRayShape, 506
  - ~MultiRayShape, 509
  - AddRay, 509
  - ConnectNewLaserScans, 509
  - DisconnectNewLaserScans, 510
  - FillMsg, 510
  - GetFiducial, 510
  - GetMaxAngle, 510
  - GetMaxRange, 510
  - GetMinAngle, 511
  - GetMinRange, 511
  - GetRange, 511
  - GetResRange, 511
  - GetRetro, 511
  - GetSampleCount, 512
  - GetScanResolution, 512
  - GetVerticalMaxAngle, 512
  - GetVerticalMinAngle, 512
  - GetVerticalSampleCount, 512
  - GetVerticalScanResolution, 512
  - horzElem, 513
  - Init, 512
  - MultiRayShape, 509
  - newLaserScans, 513
  - offset, 513
  - ProcessMsg, 513
  - rangeElem, 513
  - rayElem, 513
  - rays, 514
  - scanElem, 514
  - Update, 513
  - UpdateRays, 513
  - vertElem, 514
- gazebo::physics::PhysicsEngine, 546
  - ~PhysicsEngine, 549
  - contactManager, 556
  - CreateCollision, 549
  - CreateJoint, 550
  - CreateLink, 550
  - CreateShape, 550
  - DebugPrint, 550
  - Fini, 550
  - GetAutoDisableFlag, 550
  - GetContactManager, 550
  - GetContactMaxCorrectingVel, 551
  - GetContactSurfaceLayer, 551
  - GetGravity, 551
  - GetMaxContacts, 551
  - GetPhysicsUpdateMutex, 551
  - GetSORPGSIters, 551
  - GetSORPGSPreconIters, 552
  - GetSORPGSW, 552
  - GetStepTime, 552
  - GetUpdatePeriod, 552
  - GetUpdateRate, 552
  - GetWorldCFM, 552
  - GetWorldERP, 553
  - Init, 553

- InitForThread, 553
- Load, 553
- node, 556
- OnPhysicsMsg, 553
- OnRequest, 553
- PhysicsEngine, 549
- physicsSub, 556
- physicsUpdateMutex, 557
- requestSub, 557
- Reset, 554
- responsePub, 557
- sdf, 557
- SetAutoDisableFlag, 554
- SetContactMaxCorrectingVel, 554
- SetContactSurfaceLayer, 554
- SetGravity, 554
- SetMaxContacts, 554
- SetSORPGSIters, 555
- SetSORPGSPreconIters, 555
- SetSORPGSW, 555
- SetSeed, 555
- SetStepTime, 555
- SetUpdateRate, 556
- SetWorldCFM, 556
- SetWorldERP, 556
- UpdateCollision, 556
- UpdatePhysics, 556
- world, 557
- gazebo::physics::PhysicsFactory, 557
  - NewPhysicsEngine, 558
  - RegisterAll, 558
  - RegisterPhysicsEngine, 558
- gazebo::physics::PlaneShape, 565
  - ~PlaneShape, 566
  - CreatePlane, 567
  - FillMsg, 567
  - GetNormal, 567
  - GetSize, 567
  - Init, 567
  - PlaneShape, 566
  - ProcessMsg, 567
  - SetAltitude, 567
  - SetNormal, 568
  - SetSize, 568
- gazebo::physics::RayShape, 621
  - ~RayShape, 624
  - contactFiducial, 626
  - contactLen, 626
  - contactRetro, 627
  - FillMsg, 624
  - GetFiducial, 624
  - GetGlobalPoints, 624
  - GetIntersection, 624
  - GetLength, 625
  - GetRelativePoints, 625
  - GetRetro, 625
  - globalEndPos, 627
  - globalStartPos, 627
  - Init, 625
  - ProcessMsg, 625
  - RayShape, 623, 624
  - relativeEndPos, 627
  - relativeStartPos, 627
  - SetFiducial, 625
  - SetLength, 626
  - SetPoints, 626
  - SetRetro, 626
  - Update, 626
- gazebo::physics::Road, 639
  - ~Road, 640
  - Init, 641
  - Load, 641
  - Road, 640
- gazebo::physics::ScrewJoint
  - ~ScrewJoint, 666
  - fakeAnchor, 667
  - GetAnchor, 666
  - GetAngleCount, 666
  - Load, 667
  - ScrewJoint, 666
  - SetAnchor, 667
  - SetThreadPitch, 667
  - threadPitch, 667
- gazebo::physics::ScrewJoint< T >, 665
- gazebo::physics::Shape, 692
  - ~Shape, 694
  - collisionParent, 695
  - FillMsg, 694
  - Init, 694
  - ProcessMsg, 694
  - Shape, 693
- gazebo::physics::SliderJoint
  - ~SliderJoint, 718
  - fakeAnchor, 719
  - GetAnchor, 718
  - GetAngleCount, 719
  - Load, 719
  - SetAnchor, 719
  - SliderJoint, 718
- gazebo::physics::SliderJoint< T >, 717
- gazebo::physics::SphereShape, 721
  - ~SphereShape, 723
  - FillMsg, 723
  - GetRadius, 723
  - Init, 723
  - ProcessMsg, 723
  - SetRadius, 723
  - SphereShape, 723



- gazebo::physics::State, 728
  - ~State, 730
  - GetName, 730
  - GetRealTime, 730
  - GetSimTime, 731
  - GetWallTime, 731
  - Load, 731
  - name, 732
  - operator-, 731
  - operator=, 731
  - realTime, 732
  - SetName, 732
  - simTime, 732
  - State, 730
  - wallTime, 732
- gazebo::physics::SurfaceParams, 749
  - ~SurfaceParams, 750
  - bounce, 750
  - bounceThreshold, 751
  - cfm, 751
  - erp, 751
  - fdir1, 751
  - FillMsg, 750
  - kd, 751
  - kp, 751
  - Load, 750
  - maxVel, 752
  - minDepth, 752
  - mu1, 752
  - mu2, 752
  - ProcessMsg, 750
  - slip1, 752
  - slip2, 753
  - SurfaceParams, 750
- gazebo::physics::TrajectoryInfo, 788
  - duration, 788
  - endTime, 788
  - id, 788
  - startTime, 788
  - translated, 788
  - type, 788
- gazebo::physics::TrimeshShape, 789
  - ~TrimeshShape, 790
  - FillMsg, 790
  - GetFilename, 791
  - GetSize, 791
  - Init, 791
  - mesh, 792
  - ProcessMsg, 791
  - SetFilename, 791
  - SetScale, 791
  - TrimeshShape, 790
  - Update, 791
- gazebo::physics::UniversalJoint
  - ~UniversalJoint, 793
  - GetAngleCount, 793
  - Load, 793
  - UniversalJoint, 793
- gazebo::physics::UniversalJoint< T >, 792
- gazebo::physics::World, 874
  - ~World, 877
  - Clear, 878
  - dirtyPoses, 886
  - DisableAllModels, 878
  - EnableAllModels, 878
  - EnablePhysicsEngine, 878
  - EnqueueMsg, 878
  - Finis, 878
  - GetByName, 878
  - GetEnablePhysicsEngine, 879
  - GetEntity, 879
  - GetEntityBelowPoint, 879
  - GetModel, 879, 880
  - GetModelBelowPoint, 880
  - GetModelCount, 880
  - GetModels, 880
  - GetName, 881
  - GetPauseTime, 881
  - GetPhysicsEngine, 881
  - GetRealTime, 881
  - GetSelectedEntity, 881
  - GetSetWorldPoseMutex, 881
  - GetSimTime, 882
  - GetStartTime, 882
  - Init, 882
  - InsertModelFile, 882
  - InsertModelSDF, 882
  - InsertModelString, 882
  - IsLoaded, 883
  - IsPaused, 883
  - Load, 883
  - LoadPlugin, 883
  - PrintEntityTree, 883
  - RemovePlugin, 884
  - Reset, 884
  - ResetEntities, 884
  - ResetTime, 884
  - Run, 884
  - Save, 884
  - SetPaused, 884
  - SetSimTime, 885
  - SetState, 885
  - StepWorld, 885
  - Stop, 885
  - StripWorldName, 885
  - UpdateStateSDF, 885
  - World, 877
- gazebo::physics::WorldState, 887

- ~WorldState, 889
- FillSDF, 889
- GetModelState, 890
- GetModelStateCount, 890
- GetModelStates, 890
- HasModelState, 891
- IsZero, 891
- Load, 891
- operator<<, 892
- operator+, 891
- operator-, 891
- operator=, 892
- WorldState, 889
- gazebo::rendering, 94
  - ArrowVisualPtr, 97
  - AxisVisualPtr, 97
  - COMVisualPtr, 97
  - CameraPtr, 97
  - CameraVisualPtr, 97
  - ContactVisualPtr, 97
  - DepthCameraPtr, 97
  - DynamicLinesPtr, 97
  - GpuLaserPtr, 97
  - JointVisualPtr, 97
  - LaserVisualPtr, 97
  - LightPtr, 97
  - RENDERING\_LINE\_LIST, 97
  - RENDERING\_LINE\_STRIP, 97
  - RENDERING\_MESH\_RESOURCE, 97
  - RENDERING\_POINT\_LIST, 97
  - RENDERING\_TRIANGLE\_FAN, 97
  - RENDERING\_TRIANGLE\_LIST, 97
  - RENDERING\_TRIANGLE\_STRIP, 97
  - RFIDTagVisualPtr, 97
  - RFIDVisualPtr, 97
  - RenderOpType, 97
  - ScenePtr, 97
  - UserCameraPtr, 97
  - VisualPtr, 97
- gazebo::rendering::ArrowVisual, 125
  - ~ArrowVisual, 127
  - ArrowVisual, 126
  - Load, 127
  - ShowRotation, 127
- gazebo::rendering::AxisVisual, 129
  - ~AxisVisual, 130
  - AxisVisual, 130
  - Load, 130
  - ScaleXAxis, 130
  - ScaleYAxis, 130
  - ScaleZAxis, 131
  - SetAxisMaterial, 131
  - ShowRotation, 131
- gazebo::rendering::COMVisual, 214
  - ~COMVisual, 215
  - COMVisual, 215
  - Load, 215
- gazebo::rendering::Camera, 157
  - ~Camera, 164
  - animState, 180
  - AnimationComplete, 164
  - AttachToVisual, 164
  - AttachToVisualImpl, 164, 165
  - bayerFrameBuffer, 180
  - Camera, 163
  - camera, 180
  - captureData, 180
  - ConnectNewImageFrame, 165
  - connections, 180
  - CreateRenderTexture, 165
  - DisconnectNewImageFrame, 165
  - EnableSaveFrame, 166
  - Fini, 166
  - GetAspectRatio, 166
  - GetAvgFPS, 166
  - GetCameraToViewportRay, 166
  - GetDirection, 166
  - GetFarClip, 167
  - GetFrameFilename, 167
  - GetHFOV, 167
  - GetImageByteSize, 167
  - GetImageData, 168
  - GetImageDepth, 168
  - GetImageFormat, 168
  - GetImageHeight, 168
  - GetImageWidth, 168
  - GetInitialized, 168
  - GetLastRenderWallTime, 169
  - GetName, 169
  - GetNearClip, 169
  - GetOgreCamera, 169
  - GetPitchNode, 169
  - GetRenderRate, 169
  - GetRenderTexture, 170
  - GetRight, 170
  - GetScene, 170
  - GetSceneNode, 170
  - GetTextureHeight, 170
  - GetTextureWidth, 170
  - GetTriangleCount, 171
  - GetUp, 171
  - GetVFOV, 171
  - GetViewport, 171
  - GetViewportHeight, 171
  - GetViewportWidth, 171
  - GetWindowId, 172
  - GetWorldPointOnPlane, 172
  - GetWorldPose, 172

- GetWorldPosition, 172
- GetWorldRotation, 172
- GetZValue, 173
- imageFormat, 181
- imageHeight, 181
- imageWidth, 181
- Init, 173
- initialized, 181
- IsAnimating, 173
- IsInitialized, 173
- IsVisible, 173, 174
- lastRenderWallTime, 181
- Load, 174
- MoveToPosition, 174
- MoveToPositions, 174
- name, 181
- newData, 181
- newImageFrame, 181
- onAnimationComplete, 181
- pitchNode, 181
- PostRender, 175
- prevAnimTime, 181
- Render, 175
- RenderImpl, 175
- renderTarget, 182
- renderTexture, 182
- requests, 182
- RotatePitch, 175
- RotateYaw, 175
- saveCount, 182
- SaveFrame, 175, 176
- saveFrameBuffer, 182
- scene, 182
- sceneNode, 182
- sdf, 182
- SetAspectRatio, 176
- SetCaptureData, 176
- SetClipDist, 176
- SetHFOV, 177
- SetImageHeight, 177
- SetImageSize, 177
- SetImageWidth, 177
- SetName, 177
- SetRenderRate, 177
- SetRenderTarget, 178
- SetSaveFramePathname, 178
- SetScene, 178
- SetSceneNode, 178
- SetWindowId, 178
- SetWorldPose, 178
- SetWorldPosition, 179
- SetWorldRotation, 179
- ShowWireframe, 179
- textureHeight, 182
- textureWidth, 182
- ToggleShowWireframe, 179
- TrackVisual, 179
- TrackVisualImpl, 179, 180
- Translate, 180
- Update, 180
- viewport, 182
- windowId, 182
- gazebo::rendering::CameraVisual, 187
  - ~CameraVisual, 188
  - CameraVisual, 188
  - Load, 188
- gazebo::rendering::ContactVisual, 238
  - ~ContactVisual, 240
  - ContactVisual, 239
  - SetEnabled, 240
- gazebo::rendering::Conversions, 240
  - Convert, 241, 242
- gazebo::rendering::DepthCamera, 246
  - ~DepthCamera, 247
  - ConnectNewDepthFrame, 248
  - ConnectNewRGBPointCloud, 248
  - CreateDepthTexture, 248
  - DepthCamera, 247
  - depthTarget, 250
  - depthTexture, 250
  - depthViewport, 250
  - DisconnectNewDepthFrame, 248
  - DisconnectNewRGBPointCloud, 248
  - Fini, 249
  - GetDepthData, 249
  - Init, 249
  - Load, 249
  - PostRender, 249
  - SetDepthTarget, 249
- gazebo::rendering::DynamicLines, 258
  - ~DynamicLines, 260
  - AddPoint, 260
  - Clear, 261
  - CreateVertexDeclaration, 261
  - DynamicLines, 260
  - FillHardwareBuffers, 261
  - GetMovableType, 261
  - getMovableType, 261
  - GetPoint, 261
  - GetPointCount, 261
  - SetPoint, 262
  - Update, 262
- gazebo::rendering::DynamicRenderable, 262
  - ~DynamicRenderable, 264
  - CreateVertexDeclaration, 264
  - DynamicRenderable, 264
  - FillHardwareBuffers, 264
  - getBoundingRadius, 264

- GetOperationType, 265
- getSquaredViewDepth, 265
- indexBufferCapacity, 266
- Init, 265
- PrepareHardwareBuffers, 265
- SetOperationType, 266
- vertexBufferCapacity, 266
- gazebo::rendering::Events, 288
  - ConnectCreateScene, 288
  - ConnectRemoveScene, 288
  - createScene, 289
  - DisconnectCreateScene, 289
  - DisconnectRemoveScene, 289
  - removeScene, 289
- gazebo::rendering::FPSViewController, 312
  - ~FPSViewController, 314
  - FPSViewController, 314
  - GetTypeString, 314
  - HandleKeyPressEvent, 314
  - HandleKeyReleaseEvent, 314
  - HandleMouseEvent, 314
  - Init, 315
  - Update, 315
- gazebo::rendering::GUIOverlay, 337
  - ~GUIOverlay, 338
  - AttachCameraToImage, 338
  - ButtonCallback, 339
  - CreateWindow, 339
  - GUIOverlay, 338
  - HandleKeyPressEvent, 339
  - HandleKeyReleaseEvent, 340
  - HandleMouseEvent, 340
  - Hide, 340
  - Init, 340
  - IsInitialized, 340
  - LoadLayout, 340
  - Resize, 341
  - Show, 341
  - Update, 341
- gazebo::rendering::GpuLaser, 316
  - ~GpuLaser, 318
  - ConnectNewLaserFrame, 318
  - CreateLaserTexture, 318
  - DisconnectNewLaserFrame, 319
  - Fini, 319
  - GetLaserData, 319
  - GpuLaser, 318
  - Init, 319
  - Load, 319
  - notifyRenderSingleObject, 319
  - PostRender, 319
  - SetParentSensor, 319
  - SetRangeCount, 320
- gazebo::rendering::Grid, 332
  - ~Grid, 333
  - Enable, 334
  - GetCellCount, 334
  - GetCellLength, 334
  - GetColor, 334
  - GetHeight, 334
  - GetLineWidth, 334
  - GetSceneNode, 334
  - Grid, 333
  - Init, 335
  - SetCellCount, 335
  - SetCellLength, 335
  - SetColor, 335
  - SetHeight, 335
  - SetLineWidth, 335
  - SetUserData, 336
- gazebo::rendering::GzTerrainMatGen, 341
  - ~GzTerrainMatGen, 342
  - GzTerrainMatGen, 342
- gazebo::rendering::GzTerrainMatGen::SM2Profile, 719
  - ~SM2Profile, 721
  - addTechnique, 721
  - generate, 721
  - generateForCompositeMap, 721
  - SM2Profile, 721
  - UpdateParams, 721
  - UpdateParamsForCompositeMap, 721
- gazebo::rendering::GzTerrainMatGen::SM2Profile::~-
  - ShaderHelperCg, 688
  - defaultVpParams, 689
  - generateFragmentProgram, 689
  - generateVertexProgram, 689
  - generateVertexProgramSource, 689
  - generateVpDynamicShadows, 689
  - generateVpDynamicShadowsParams, 689
  - generateVpFooter, 689
  - generateVpHeader, 689
- gazebo::rendering::GzTerrainMatGen::SM2Profile::~-
  - ShaderHelperGLSL, 690
  - defaultVpParams, 691
  - generateFpDynamicShadows, 691
  - generateFpDynamicShadowsHelpers, 691
  - generateFpDynamicShadowsParams, 691
  - generateFpFooter, 691
  - generateFpHeader, 691
  - generateFpLayer, 691
  - generateFragmentProgram, 692
  - generateFragmentProgramSource, 692
  - generateVertexProgram, 692
  - generateVertexProgramSource, 692
  - generateVpDynamicShadows, 692
  - generateVpDynamicShadowsParams, 692
  - generateVpFooter, 692
  - generateVpHeader, 692

- updateParams, 692
- updateVpParams, 692
- gazebo::rendering::Heightmap, 342
  - ~Heightmap, 343
  - GetHeight, 343
  - GetOgreTerrain, 343
  - Heightmap, 343
  - Load, 343
  - LoadFromMsg, 343
- gazebo::rendering::JointVisual, 391
  - ~JointVisual, 392
  - JointVisual, 392
  - Load, 392
- gazebo::rendering::LaserVisual, 395
  - ~LaserVisual, 397
  - LaserVisual, 396
  - SetEmissive, 397
- gazebo::rendering::Light, 397
  - ~Light, 399
  - FillMsg, 399
  - GetDiffuseColor, 399
  - GetDirection, 399
  - GetName, 399
  - GetPosition, 399
  - GetSpecularColor, 400
  - GetType, 400
  - Light, 399
  - Load, 400
  - LoadFromMsg, 400
  - OnPoseChange, 400
  - SetAttenuation, 400
  - SetCastShadows, 401
  - SetDiffuseColor, 401
  - SetDirection, 401
  - SetLightType, 401
  - SetName, 401
  - SetPosition, 401
  - SetRange, 402
  - SetSelected, 402
  - SetSpecularColor, 402
  - SetSpotFalloff, 402
  - SetSpotInnerAngle, 402
  - SetSpotOuterAngle, 402
  - ShowVisual, 403
  - ToggleShowVisual, 403
  - UpdateFromMsg, 403
- gazebo::rendering::MovableText, 494
  - ~MovableText, 497
  - \_setupGeometry, 497
  - \_updateColors, 497
  - GetAABB, 497
  - GetBaseline, 497
  - getBoundingRadius, 497
  - GetCharHeight, 497
  - GetColor, 497
  - GetFont, 498
  - getLights, 498
  - getMaterial, 498
  - getRenderOperation, 498
  - GetShowOnTop, 498
  - GetSpaceWidth, 498
  - getSquaredViewDepth, 498
  - GetText, 498
  - getWorldTransforms, 498
  - H\_CENTER, 496
  - H\_LEFT, 496
  - HorizAlign, 496
  - Load, 499
  - MovableText, 497
  - SetBaseline, 499
  - SetCharHeight, 499
  - SetColor, 499
  - SetFontName, 499
  - SetShowOnTop, 499
  - SetSpaceWidth, 500
  - SetText, 500
  - SetTextAlignment, 500
  - Update, 500
  - V\_ABOVE, 497
  - V\_BELOW, 497
  - VertAlign, 496
  - visitRenderables, 500
- gazebo::rendering::OrbitViewController, 533
  - ~OrbitViewController, 535
  - GetFocalPoint, 535
  - GetTypeString, 535
  - HandleKeyPressEvent, 535
  - HandleKeyReleaseEvent, 535
  - HandleMouseEvent, 536
  - Init, 536
  - OrbitViewController, 535
  - SetDistance, 536
  - SetFocalPoint, 536
  - Update, 537
- gazebo::rendering::Projector, 585
  - ~Projector, 586
  - GetParent, 586
  - Load, 586
  - Projector, 586
  - SetEnabled, 587
  - SetTexture, 587
  - Toggle, 587
- gazebo::rendering::RFIDTagVisual, 636
  - ~RFIDTagVisual, 638
  - RFIDTagVisual, 637
- gazebo::rendering::RFIDVisual, 638
  - ~RFIDVisual, 639
  - RFIDVisual, 639

- gazebo::rendering::RTShaderSystem, 646
  - AddScene, 647
  - ApplyShadows, 648
  - AttachEntity, 648
  - AttachViewport, 648
  - Clear, 648
  - DetachEntity, 648
  - DetachViewport, 648
  - Fini, 649
  - GenerateShaders, 649
  - GetPSSMShadowCameraSetup, 649
  - Init, 649
  - LightingModel, 647
  - RemoveScene, 649
  - RemoveShadows, 649
  - SSLM\_NormalMapLightingObjectSpace, 647
  - SSLM\_NormalMapLightingTangentSpace, 647
  - SSLM\_PerPixelLighting, 647
  - SSLM\_PerVertexLighting, 647
  - SetPerPixelLighting, 649
  - UpdateShaders, 650
- gazebo::rendering::RenderEngine, 627
  - AddResourcePath, 629
  - CreateScene, 629
  - DEFERRED, 629
  - dummyContext, 631
  - dummyDisplay, 631
  - dummyWindowId, 631
  - FORWARD, 629
  - Fini, 630
  - GetRenderPathType, 630
  - GetScene, 630
  - GetSceneCount, 630
  - Init, 631
  - Load, 631
  - NONE, 629
  - RENDER\_PATH\_COUNT, 629
  - RemoveScene, 631
  - RenderPathType, 629
  - root, 631
  - VERTEX, 629
- gazebo::rendering::Road2d, 641
  - ~Road2d, 641
  - Load, 642
  - Road2d, 641
- gazebo::rendering::Scene, 650
  - ~Scene, 653
  - AddVisual, 653
  - Clear, 654
  - CloneVisual, 654
  - CreateCamera, 654
  - CreateDepthCamera, 654
  - CreateGrid, 655
  - CreateUserCamera, 655
  - DrawLine, 655
  - GetAmbientColor, 655
  - GetBackgroundColor, 655
  - GetCamera, 656
  - GetCameraCount, 656
  - GetFirstContact, 656
  - GetGrid, 657
  - GetGridCount, 657
  - GetHeightBelowPoint, 657
  - GetHeightmap, 657
  - GetId, 658
  - GetIdString, 658
  - GetLight, 658
  - GetLightCount, 658
  - GetManager, 659
  - GetModelVisualAt, 659
  - GetName, 659
  - GetSelectedVisual, 659
  - GetShadowsEnabled, 659
  - GetUserCamera, 659
  - GetUserCameraCount, 660
  - GetVisual, 660
  - GetVisualAt, 660
  - GetVisualBelow, 661
  - GetVisualsBelowPoint, 661
  - GetWorldVisual, 661
  - Init, 661
  - Load, 661
  - PreRender, 662
  - PrintSceneGraph, 662
  - RemoveVisual, 662
  - Scene, 653
  - SelectVisual, 662
  - SetAmbientColor, 662
  - SetBackgroundColor, 662
  - SetFog, 662
  - SetGrid, 663
  - SetShadowsEnabled, 663
  - SetTransparent, 663
  - SetVisible, 663
  - ShowCOMs, 664
  - ShowCollisions, 663
  - ShowContacts, 664
  - ShowJoints, 664
  - skyx, 665
  - SnapVisualToNearestBelow, 664
  - StripSceneName, 664
- gazebo::rendering::SelectionObj, 669
  - ~SelectionObj, 670
  - Attach, 670
  - Clear, 670
  - GetVisualName, 670
  - Init, 670
  - IsActive, 670

- SelectionObj, 670
- SetActive, 670
- SetHighlight, 671
- gazebo::rendering::UserCamera, 795
  - ~UserCamera, 797
  - AnimationComplete, 797
  - AttachToVisualImpl, 797
  - EnableViewController, 798
  - Fini, 798
  - GetAvgFPS, 798
  - GetGUIOverlay, 798
  - GetTriangleCount, 799
  - GetViewControllerTypeString, 799
  - GetVisual, 799
  - HandleKeyPressEvent, 799
  - HandleKeyReleaseEvent, 800
  - HandleMouseEvent, 800
  - Init, 800
  - Load, 800
  - MoveToPosition, 800
  - MoveToVisual, 801
  - PostRender, 801
  - Resize, 801
  - SetFocalPoint, 801
  - SetRenderTarget, 801
  - SetViewController, 802
  - SetViewportDimensions, 802
  - SetWorldPose, 802
  - TrackVisualImpl, 802
  - Update, 803
  - UserCamera, 797
- gazebo::rendering::VideoVisual, 844
  - ~VideoVisual, 845
  - VideoVisual, 845
- gazebo::rendering::ViewController, 846
  - ~ViewController, 847
  - camera, 849
  - enabled, 849
  - GetTypeString, 847
  - HandleKeyPressEvent, 847
  - HandleKeyReleaseEvent, 848
  - HandleMouseEvent, 848
  - Init, 848
  - SetEnabled, 848
  - typeString, 849
  - Update, 849
  - ViewController, 847
- gazebo::rendering::Visual, 849
  - ~Visual, 855
  - AttachAxes, 855
  - AttachLineVertex, 855
  - AttachMesh, 855
  - AttachObject, 855
  - AttachVisual, 855
  - ClearParent, 856
  - Clone, 856
  - CreateDynamicLine, 856
  - DeleteDynamicLine, 856
  - DetachObjects, 856
  - DetachVisual, 856, 857
  - DisableTrackVisual, 857
  - EnableTrackVisual, 857
  - Fini, 857
  - GetAttachedObjectCount, 857
  - GetBoundingBox, 857
  - GetChild, 857
  - GetChildCount, 858
  - GetMaterialName, 858
  - GetMeshName, 858
  - GetName, 858
  - GetNormalMap, 858
  - GetParent, 858
  - GetPose, 859
  - GetPosition, 859
  - GetRootVisual, 859
  - GetRotation, 859
  - GetScale, 859
  - GetScene, 859
  - GetSceneNode, 860
  - GetShaderType, 860
  - GetTransparency, 860
  - GetVisibilityFlags, 860
  - GetVisible, 860
  - GetWorldPose, 861
  - HasAttachedObject, 861
  - Init, 861
  - InsertMesh, 861
  - IsPlane, 861
  - IsStatic, 862
  - Load, 862
  - LoadFromMsg, 862
  - LoadPlugin, 862
  - MakeStatic, 862
  - MoveToPosition, 862
  - MoveToPositions, 863
  - parent, 868
  - RemovePlugin, 863
  - scene, 868
  - sceneNode, 869
  - SetAmbient, 863
  - SetCastShadows, 863
  - SetDiffuse, 863
  - SetEmissive, 864
  - SetHighlighted, 864
  - SetMaterial, 864
  - SetName, 864
  - SetNormalMap, 864
  - SetPose, 864

- SetPosition, 865
- SetRibbonTrail, 865
- SetRotation, 865
- SetScale, 865
- SetScene, 865
- SetShaderType, 866
- SetSkeletonPose, 866
- SetSpecular, 866
- SetTransparency, 866
- SetVisibilityFlags, 866
- SetVisible, 867
- SetWorldPose, 867
- SetWorldPosition, 867
- SetWorldRotation, 867
- ShowBoundingBox, 867
- ShowCOM, 867
- ShowCollision, 867
- ShowJoints, 868
- ShowSkeleton, 868
- ToggleVisible, 868
- Update, 868
- UpdateFromMsg, 868
- Visual, 854
- gazebo::rendering::WindowManager, 870
  - CreateWindow, 871
  - Fini, 872
  - GetAvgFPS, 872
  - GetTriangleCount, 872
  - GetWindow, 872
  - Moved, 872
  - Resize, 873
  - SetCamera, 873
- gazebo::rendering::WireBox, 873
  - ~WireBox, 874
  - Init, 874
  - SetVisible, 874
  - WireBox, 874
- gazebo::sensors, 98
  - CameraSensor\_V, 99
  - CameraSensorPtr, 99
  - ContactSensor\_V, 99
  - ContactSensorPtr, 99
  - DepthCameraSensor\_V, 99
  - DepthCameraSensorPtr, 99
  - GpuRaySensor\_V, 100
  - GpuRaySensorPtr, 100
  - RFIDSensor\_V, 100
  - RFIDSensorPtr, 100
  - RFIDTag\_V, 100
  - RFIDTagPtr, 100
  - RaySensor\_V, 100
  - RaySensorPtr, 100
  - Sensor\_V, 100
  - SensorFactoryFn, 100
  - SensorPtr, 100
- gazebo::sensors::CameraSensor, 183
  - ~CameraSensor, 184
  - CameraSensor, 184
  - Fini, 184
  - GetCamera, 184
  - GetImageData, 185
  - GetImageHeight, 185
  - GetImageWidth, 185
  - GetTopic, 185
  - Init, 185
  - IsActive, 185
  - Load, 186
  - SaveFrame, 186
  - SetParent, 186
  - UpdateImpl, 187
- gazebo::sensors::ContactSensor, 234
  - ~ContactSensor, 236
  - ContactSensor, 236
  - Fini, 236
  - GetCollisionContactCount, 236
  - GetCollisionCount, 236
  - GetCollisionName, 237
  - GetContacts, 237
  - Init, 237
  - IsActive, 237
  - Load, 238
  - UpdateImpl, 238
- gazebo::sensors::DepthCameraSensor, 250
  - ~DepthCameraSensor, 252
  - DepthCameraSensor, 252
  - Fini, 252
  - GetDepthCamera, 252
  - Init, 252
  - Load, 252
  - SaveFrame, 253
  - SetActive, 253
  - SetParent, 253
  - UpdateImpl, 253
- gazebo::sensors::GpuRaySensor, 320
  - ~GpuRaySensor, 323
  - cameraCount, 330
  - cameraElem, 330
  - chfov, 330
  - ConnectNewLaserFrame, 324
  - cvfov, 330
  - DisconnectNewLaserFrame, 324
  - far, 331
  - Fini, 324
  - GetAngleMax, 324
  - GetAngleMin, 324
  - GetAngleResolution, 324
  - GetCameraCount, 324
  - GetCosHorzFOV, 325



- GetCosVertFOV, 325
- GetFiducial, 325
- GetHorzFOV, 325
- GetHorzHalfAngle, 325
- GetLaserCamera, 326
- GetRange, 326
- GetRangeCount, 326
- GetRangeCountRatio, 326
- GetRangeMax, 326
- GetRangeMin, 327
- GetRangeResolution, 327
- GetRanges, 327
- GetRayCount, 327
- GetRayCountRatio, 327
- GetRetro, 327
- GetVertFOV, 328
- GetVertHalfAngle, 328
- GetVerticalAngleMax, 328
- GetVerticalAngleMin, 328
- GetVerticalRangeCount, 328
- GetVerticalRayCount, 328
- GpuRaySensor, 323
- hfov, 331
- horzElem, 331
- horzHalfAngle, 331
- horzRangeCount, 331
- horzRayCount, 331
- Init, 329
- IsHorizontal, 329
- isHorizontal, 331
- Load, 329
- near, 331
- rangeCountRatio, 331
- rangeElem, 331
- rayCountRatio, 331
- scanElem, 332
- SetAngleMax, 329
- SetAngleMin, 329
- SetVerticalAngleMax, 330
- SetVerticalAngleMin, 330
- UpdateImpl, 330
- vertElem, 332
- vertHalfAngle, 332
- vertRangeCount, 332
- vertRayCount, 332
- vfov, 332
- gazebo::sensors::ImuSensor, 357
  - ~ImuSensor, 359
  - Fini, 359
  - GetAngularVelocity, 359
  - GetLinearAcceleration, 359
  - ImuSensor, 359
  - Init, 359
  - Load, 359, 360
  - UpdateImpl, 360
- gazebo::sensors::MultiCameraSensor, 502
  - ~MultiCameraSensor, 503
  - Fini, 503
  - GetCamera, 503
  - GetCameraCount, 504
  - GetImageData, 504
  - GetImageHeight, 504
  - GetImageWidth, 504
  - GetTopic, 505
  - Init, 505
  - Load, 505
  - MultiCameraSensor, 503
  - SaveFrame, 505
  - UpdateImpl, 506
- gazebo::sensors::RFIDSensor, 632
  - ~RFIDSensor, 633
  - AddTag, 633
  - Fini, 633
  - Init, 633
  - Load, 633
  - RFIDSensor, 633
  - UpdateImpl, 634
- gazebo::sensors::RFIDTag, 634
  - ~RFIDTag, 635
  - Fini, 635
  - GetTagPose, 635
  - Init, 636
  - Load, 636
  - RFIDTag, 635
  - UpdateImpl, 636
- gazebo::sensors::RaySensor, 615
  - ~RaySensor, 616
  - Fini, 617
  - GetAngleMax, 617
  - GetAngleMin, 617
  - GetAngleResolution, 617
  - GetFiducial, 617
  - GetLaserShape, 618
  - GetRange, 618
  - GetRangeCount, 618
  - GetRangeMax, 618
  - GetRangeMin, 618
  - GetRangeResolution, 619
  - GetRanges, 619
  - GetRayCount, 619
  - GetRetro, 619
  - GetTopic, 620
  - GetVerticalAngleMax, 620
  - GetVerticalAngleMin, 620
  - GetVerticalRangeCount, 620
  - GetVerticalRayCount, 620
  - Init, 620
  - IsActive, 620

- Load, 621
- RaySensor, 616
- UpdateImpl, 621
- gazebo::sensors::Sensor, 671
  - ~Sensor, 674
  - active, 679
  - ConnectUpdated, 674
  - connections, 679
  - DisconnectUpdated, 674
  - FillMsg, 674
  - Fini, 675
  - GetLastMeasurementTime, 675
  - GetLastUpdateTime, 675
  - GetName, 675
  - GetParentName, 675
  - GetPose, 675
  - GetScopedName, 676
  - GetTopic, 676
  - GetType, 676
  - GetUpdateRate, 676
  - GetVisualize, 676
  - GetWorldName, 676
  - Init, 677
  - IsActive, 677
  - lastMeasurementTime, 679
  - lastUpdateTime, 679
  - Load, 677
  - node, 679
  - parentName, 679
  - plugins, 679
  - pose, 679
  - poseSub, 679
  - sdf, 679
  - Sensor, 674
  - SetActive, 678
  - SetParent, 678
  - SetUpdateRate, 678
  - Update, 678
  - UpdateImpl, 678
  - updatePeriod, 680
  - world, 680
- gazebo::sensors::SensorFactory, 680
  - GetSensorTypes, 681
  - NewSensor, 681
  - RegisterAll, 681
  - RegisterSensor, 681
- gazebo::sensors::SensorManager, 682
  - CreateSensor, 683
  - Fini, 683
  - GetSensor, 683
  - GetSensorTypes, 684
  - GetSensors, 684
  - Init, 684
  - RemoveSensor, 684
  - RemoveSensors, 684
  - Run, 684
  - SensorsInitialized, 684
  - Stop, 684
  - Update, 685
- gazebo::transport, 100
  - ConnectionPtr, 102
  - NodePtr, 102
  - PublicationPtr, 102
  - PublicationTransportPtr, 102
  - PublisherPtr, 102
  - SubscriberPtr, 102
  - SubscriptionTransportPtr, 102
- gazebo::transport::CallbackHelper, 153
  - ~CallbackHelper, 154
  - CallbackHelper, 154
  - GetId, 154
  - GetLatching, 154
  - GetMsgType, 154
  - HandleData, 155
  - IsLocal, 155
  - latching, 155
- gazebo::transport::CallbackHelperT
  - CallbackHelperT, 156
  - GetMsgType, 157
  - HandleData, 157
  - IsLocal, 157
- gazebo::transport::CallbackHelperT< M >, 155
- gazebo::transport::Connection, 217
  - ~Connection, 219
  - AcceptCallback, 219
  - AsyncRead, 219
  - Cancel, 219
  - Connect, 219
  - ConnectToShutdown, 220
  - Connection, 219
  - DisconnectShutdown, 220
  - EnqueueMsg, 220
  - GetId, 220
  - GetLocalAddress, 221
  - GetLocalHostname, 221
  - GetLocalPort, 221
  - GetLocalURI, 221
  - GetRemoteAddress, 221
  - GetRemoteHostname, 221
  - GetRemotePort, 222
  - GetRemoteURI, 222
  - IsOpen, 222
  - Listen, 222
  - ProcessWriteQueue, 222
  - Read, 222
  - ReadCallback, 219
  - Shutdown, 223
  - StartRead, 223

- StopRead, 223
- ValidateIP, 223
- gazebo::transport::ConnectionManager, 223
  - Advertise, 225
  - ConnectToRemoteHost, 225
  - eventConnections, 228
  - Fini, 225
  - GetAllPublishers, 225
  - GetTopicNamespaces, 226
  - Init, 226
  - IsRunning, 226
  - RegisterTopicNamespace, 226
  - RemoveConnection, 226
  - Run, 226
  - RunUpdate, 227
  - Stop, 227
  - Subscribe, 227
  - Unadvertise, 227
  - Unsubscribe, 227
- gazebo::transport::IOManager, 369
  - ~IOManager, 370
  - DecCount, 370
  - GetCount, 370
  - GetIO, 370
  - IOManager, 370
  - IncCount, 370
  - Stop, 371
- gazebo::transport::Node, 514
  - ~Node, 516
  - Advertise, 516
  - DecodeTopicName, 516
  - EncodeTopicName, 516
  - Fini, 517
  - GetId, 517
  - GetMsgType, 517
  - GetTopicNamespace, 517
  - HandleData, 517
  - HasLatchedSubscriber, 518
  - Init, 518
  - InsertLatchedMsg, 518
  - Node, 516
  - ProcessIncoming, 518
  - ProcessPublishers, 518
  - RemoveCallback, 518
  - Subscribe, 518–520
- gazebo::transport::Publication, 587
  - ~Publication, 588
  - AddPublisher, 589
  - AddSubscription, 589
  - AddTransport, 589
  - GetCallbackCount, 589
  - GetLocallyAdvertised, 589
  - GetMsgType, 590
  - GetNodeCount, 590
  - GetRemoteSubscriptionCount, 590
  - GetTransportCount, 590
  - HasTransport, 590
  - LocalPublish, 591
  - Publication, 588
  - Publish, 591
  - RemoveSubscription, 591
  - RemoveTransport, 591
  - SetLocallyAdvertised, 591
- gazebo::transport::PublicationTransport, 592
  - ~PublicationTransport, 593
  - AddCallback, 593
  - Fini, 593
  - GetConnection, 593
  - GetMsgType, 593
  - GetTopic, 593
  - Init, 594
  - PublicationTransport, 593
- gazebo::transport::Publisher, 594
  - ~Publisher, 595
  - GetLatching, 595
  - GetMsgType, 595
  - GetOutgoingCount, 596
  - GetPrevMsg, 596
  - GetTopic, 596
  - HasConnections, 596
  - Publish, 596
  - Publisher, 595
  - SendMessage, 597
  - SetPublication, 597
  - WaitForConnection, 597
- gazebo::transport::RawCallbackHelper, 613
  - GetMsgType, 614
  - HandleData, 614
  - IsLocal, 614
  - RawCallbackHelper, 614
- gazebo::transport::SubscribeOptions, 743
  - GetLatching, 744
  - GetMsgType, 744
  - GetNode, 744
  - GetTopic, 744
  - Init, 745
  - SubscribeOptions, 744
- gazebo::transport::Subscriber, 745
  - ~Subscriber, 746
  - GetCallbackId, 746
  - GetTopic, 746
  - SetCallbackId, 746
  - Subscriber, 746
  - Unsubscribe, 746
- gazebo::transport::SubscriptionTransport, 747
  - ~SubscriptionTransport, 748
  - GetConnection, 748
  - HandleData, 748

- Init, 748
- IsLocal, 748
- SubscriptionTransport, 748
- gazebo::transport::TopicManager, 781
  - AddNode, 783
  - Advertise, 784
  - ClearBuffers, 784
  - ConnectPubToSub, 784
  - ConnectSubToPub, 784
  - ConnectSubscribers, 784
  - DisconnectPubFromSub, 784
  - DisconnectSubFromPub, 785
  - FindPublication, 785
  - Finis, 785
  - GetAdvertisedTopics, 785
  - GetTopicNamespaces, 785
  - Init, 786
  - IsAdvertised, 786
  - PauseIncoming, 786
  - ProcessNodes, 786
  - Publish, 786
  - RegisterTopicNamespace, 787
  - RemoveNode, 787
  - SubNodeMap, 783
  - Subscribe, 787
  - Unadvertise, 787
  - Unsubscribe, 787
  - UpdatePublications, 788
- gazebo\_core.hh, 943
- Gazebo\_parser, 68
  - ConstUrdfLinkPtr, 68
  - UrdfCollisionPtr, 68
  - UrdfLinkPtr, 68
  - UrdfVisualPtr, 68
- GazeboGenerator
  - google::protobuf::compiler::cpp::GazeboGenerator, 316
- GazeboGenerator.hh, 943
- gazeboPathsFromEnv
  - gazebo::common::SystemPaths, 757
- GenSphericalTexCoord
  - gazebo::common::Mesh, 457
  - gazebo::common::MeshManager, 467
  - gazebo::common::SubMesh, 738
- Generate
  - google::protobuf::compiler::cpp::GazeboGenerator, 316
- generate
  - gazebo::rendering::GzTerrainMatGen::SM2Profile, 721
- generateForCompositeMap
  - gazebo::rendering::GzTerrainMatGen::SM2Profile, 721
- generateFpDynamicShadows
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 691
- generateFpDynamicShadowsHelpers
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 691
- generateFpDynamicShadowsParams
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 691
- generateFpFooter
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 691
- generateFpHeader
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 691
- generateFpLayer
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 691
- generateFragmentProgram
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 689
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 692
- generateFragmentProgramSource
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 692
- GenerateShaders
  - gazebo::rendering::RTShaderSystem, 649
- generateVertexProgram
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 689
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 692
- generateVertexProgramSource
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 689
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 692
- generateVpDynamicShadows
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 689
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 692
- generateVpDynamicShadowsParams
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 689
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 692
- generateVpFooter
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 689
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 692
- generateVpHeader
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::

- ShaderHelperCg, 689
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::
    - ShaderHelperGLSL, 692
- GeneratorType
  - gazebo::math, 87
- Get
  - gazebo::common::NodeTransform, 527
  - sdf::Param, 539, 540
- get\_master\_uri
  - Transport, 75
- get\_scene
  - Rendering, 66
- get\_sensor
  - Sensors, 71
- get\_topic\_namespaces
  - Transport, 75
- get\_world
  - Classes for physics and dynamics, 44
- GetAABB
  - gazebo::common::Mesh, 457
  - gazebo::rendering::MovableText, 497
- GetAbs
  - gazebo::math::Vector3, 825
- GetAcceleration
  - gazebo::physics::LinkState, 423
- GetAdvertisedTopics
  - gazebo::transport::TopicManager, 785
- getAdvertisedTopics
  - Transport, 75
- GetAllPublishers
  - gazebo::transport::ConnectionManager, 225
- GetAmbient
  - gazebo::common::Material, 438
- GetAmbientColor
  - gazebo::rendering::Scene, 655
- GetAnchor
  - gazebo::physics::Joint, 376
  - gazebo::physics::ScrewJoint, 666
  - gazebo::physics::SliderJoint, 718
- GetAngle
  - gazebo::physics::Joint, 376
  - gazebo::physics::JointState, 389
- GetAngleCount
  - gazebo::physics::BallJoint, 133
  - gazebo::physics::Hinge2Joint, 350
  - gazebo::physics::HingeJoint, 351
  - gazebo::physics::Joint, 377
  - gazebo::physics::JointState, 389
  - gazebo::physics::ScrewJoint, 666
  - gazebo::physics::SliderJoint, 719
  - gazebo::physics::UniversalJoint, 793
- GetAngleImpl
  - gazebo::physics::Joint, 377
- GetAngleMax
  - gazebo::sensors::GpuRaySensor, 324
  - gazebo::sensors::RaySensor, 617
- GetAngleMin
  - gazebo::sensors::GpuRaySensor, 324
  - gazebo::sensors::RaySensor, 617
- GetAngleResolution
  - gazebo::sensors::GpuRaySensor, 324
  - gazebo::sensors::RaySensor, 617
- GetAngles
  - gazebo::physics::JointState, 389
- GetAngularDamping
  - gazebo::physics::Link, 411
- GetAngularVelocity
  - gazebo::sensors::ImuSensor, 359
- GetAnimation
  - gazebo::common::Skeleton, 700
- GetAsABGR
  - gazebo::common::Color, 206
- GetAsARGB
  - gazebo::common::Color, 206
- GetAsAxis
  - gazebo::math::Quaternion, 602
- GetAsBGRA
  - gazebo::common::Color, 206
- GetAsEuler
  - gazebo::math::Quaternion, 602
- GetAsHSV
  - gazebo::common::Color, 207
- GetAsMatrix3
  - gazebo::math::Quaternion, 602
- GetAsMatrix4
  - gazebo::math::Quaternion, 603
- GetAsPose
  - gazebo::math::Matrix4, 450
- GetAsRGBA
  - gazebo::common::Color, 207
- GetAsString
  - sdf::Param, 540
  - sdf::ParamT, 545
- GetAsYUV
  - gazebo::common::Color, 207
- GetAspectRatio
  - gazebo::rendering::Camera, 166
- GetAttachedObjectCount
  - gazebo::rendering::Visual, 857
- GetAttribute
  - sdf::Element, 270
- GetAttributeCount
  - sdf::Element, 270
- GetAttributeSet
  - sdf::Element, 270
- GetAutoDisable
  - gazebo::physics::Model, 473
- GetAutoDisableFlag

- gazebo::physics::PhysicsEngine, 550
- GetAvgColor
  - gazebo::common::Image, 354
- GetAvgFPS
  - gazebo::rendering::Camera, 166
  - gazebo::rendering::UserCamera, 798
  - gazebo::rendering::WindowManager, 872
- GetBPP
  - gazebo::common::Image, 355
- GetBackgroundColor
  - gazebo::rendering::Scene, 655
- GetBaseline
  - gazebo::rendering::MovableText, 497
- GetBindShapeTransform
  - gazebo::common::Skeleton, 700
- GetBlendFactors
  - gazebo::common::Material, 438
- GetBlendMode
  - gazebo::common::Material, 438
- GetBoundingBox
  - gazebo::physics::Collision, 193
  - gazebo::physics::Entity, 277
  - gazebo::physics::Link, 411
  - gazebo::physics::Model, 473
  - gazebo::rendering::Visual, 857
- getBoundingRadius
  - gazebo::rendering::DynamicRenderable, 264
  - gazebo::rendering::MovableText, 497
- GetById
  - gazebo::physics::Base, 138
- GetByName
  - gazebo::physics::Base, 138
  - gazebo::physics::World, 878
- GetCallbackCount
  - gazebo::transport::Publication, 589
- GetCallbackId
  - gazebo::transport::Subscriber, 746
- GetCamera
  - gazebo::rendering::Scene, 656
  - gazebo::sensors::CameraSensor, 184
  - gazebo::sensors::MultiCameraSensor, 503
- GetCameraCount
  - gazebo::rendering::Scene, 656
  - gazebo::sensors::GpuRaySensor, 324
  - gazebo::sensors::MultiCameraSensor, 504
- GetCameraToViewportRay
  - gazebo::rendering::Camera, 166
- GetCellCount
  - gazebo::rendering::Grid, 334
- GetCellLength
  - gazebo::rendering::Grid, 334
- GetCenter
  - gazebo::math::Box, 146
- GetCharHeight
  - gazebo::rendering::MovableText, 497
- GetChild
  - gazebo::common::SkeletonNode, 711
  - gazebo::physics::Base, 138
  - gazebo::physics::Joint, 377
  - gazebo::rendering::Visual, 857
- GetChildById
  - gazebo::common::SkeletonNode, 711
- GetChildByName
  - gazebo::common::SkeletonNode, 712
- GetChildCollision
  - gazebo::physics::Entity, 277
- GetChildCount
  - gazebo::common::SkeletonNode, 712
  - gazebo::physics::Base, 139
  - gazebo::rendering::Visual, 858
- GetChildJointsLinks
  - gazebo::physics::Link, 411
- GetChildLink
  - gazebo::physics::Entity, 278
- GetChunk
  - gazebo::common::LogPlay, 428
- GetChunkCount
  - gazebo::common::LogPlay, 428
- GetCmd
  - gazebo::common::PID, 560
- GetCoG
  - gazebo::physics::Inertial, 363
- GetCollision
  - gazebo::physics::Link, 411
- GetCollisionBoundingBox
  - gazebo::physics::Entity, 278
- GetCollisionById
  - gazebo::physics::Link, 412
- GetCollisionContactCount
  - gazebo::sensors::ContactSensor, 236
- GetCollisionCount
  - gazebo::sensors::ContactSensor, 236
- GetCollisionName
  - gazebo::sensors::ContactSensor, 237
- GetCollisionState
  - gazebo::physics::LinkState, 423, 424
- GetCollisionStateCount
  - gazebo::physics::LinkState, 424
- GetCollisionStates
  - gazebo::physics::LinkState, 424
- GetCollisions
  - gazebo::physics::Link, 412
- GetColor
  - gazebo::rendering::Grid, 334
  - gazebo::rendering::MovableText, 497
- GetConnection
  - gazebo::transport::PublicationTransport, 593
  - gazebo::transport::SubscriptionTransport, 748

- GetContact
  - gazebo::physics::ContactManager, 233
- GetContactCount
  - gazebo::physics::ContactManager, 233
- GetContactManager
  - gazebo::physics::PhysicsEngine, 550
- GetContactMaxCorrectingVel
  - gazebo::physics::PhysicsEngine, 551
- GetContactSurfaceLayer
  - gazebo::physics::PhysicsEngine, 551
- GetContacts
  - gazebo::physics::ContactManager, 233
  - gazebo::sensors::ContactSensor, 237
- GetContactsEnabled
  - gazebo::physics::Collision, 193
- GetCopyChildren
  - sdf::Element, 270
- GetCosHorzFOV
  - gazebo::sensors::GpuRaySensor, 325
- GetCosVertFOV
  - gazebo::sensors::GpuRaySensor, 325
- GetCount
  - gazebo::transport::IOManager, 370
- GetCurrentDir
  - SystemPaths.hh, 1068
- GetDBConfig
  - Common, 34
- GetData
  - gazebo::common::Image, 355
- GetDbNormal
  - gazebo::math::Rand, 612
- GetDbUniform
  - gazebo::math::Rand, 612
- GetDefaultAsString
  - sdf::Param, 540
  - sdf::ParamT, 545
- GetDefaultValue
  - sdf::ParamT, 545
- GetDepthCamera
  - gazebo::sensors::DepthCameraSensor, 252
- GetDepthData
  - gazebo::rendering::DepthCamera, 249
- GetDepthWrite
  - gazebo::common::Material, 438
- GetDescription
  - sdf::Element, 270
  - sdf::Param, 540
- GetDiffuse
  - gazebo::common::Material, 438
- GetDiffuseColor
  - gazebo::rendering::Light, 399
- GetDirection
  - gazebo::rendering::Camera, 166
  - gazebo::rendering::Light, 399
- GetDirtyPose
  - gazebo::physics::Entity, 278
- GetDistToLine
  - gazebo::math::Vector3, 825
- GetElapsed
  - gazebo::common::Timer, 781
- GetElement
  - sdf::Element, 270
- GetElementDescription
  - sdf::Element, 270
- GetElementDescriptionCount
  - sdf::Element, 270
- GetElementImpl
  - sdf::Element, 271
- GetEmissive
  - gazebo::common::Material, 438
- GetEnablePhysicsEngine
  - gazebo::physics::World, 879
- GetEnabled
  - gazebo::common::DiagnosticManager, 255
  - gazebo::physics::Link, 412
- GetEncoding
  - gazebo::common::LogPlay, 428
  - gazebo::common::LogRecord, 431
- GetEntity
  - gazebo::physics::World, 879
- GetEntityBelowPoint
  - gazebo::physics::World, 879
- GetErrorFile
  - gazebo::common::Exception, 311
- GetErrorStr
  - gazebo::common::Exception, 312
- GetErrors
  - gazebo::common::PID, 560
- GetEulerRotation
  - gazebo::math::Matrix4, 450
- GetExp
  - gazebo::math::Quaternion, 603
- GetFarClip
  - gazebo::rendering::Camera, 167
- GetFiducial
  - gazebo::physics::MultiRayShape, 510
  - gazebo::physics::RayShape, 624
  - gazebo::sensors::GpuRaySensor, 325
  - gazebo::sensors::RaySensor, 617
- GetFilename
  - gazebo::common::Image, 355
  - gazebo::physics::TrimeshShape, 791
  - gazebo::PluginT, 571
- GetFirstContact
  - gazebo::rendering::Scene, 656
- GetFirstElement
  - sdf::Element, 271
- GetFocalPoint

- gazebo::rendering::OrbitViewController, 535
- GetFont
  - gazebo::rendering::MovableText, 498
- GetForce
  - gazebo::physics::Joint, 377
- GetForceTorque
  - gazebo::physics::Joint, 378
- GetFrameAt
  - gazebo::common::NodeAnimation, 522
- GetFrameCount
  - gazebo::common::NodeAnimation, 522
- GetFrameFilename
  - gazebo::rendering::Camera, 167
- GetGUIOverlay
  - gazebo::rendering::UserCamera, 798
- GetGazeboPaths
  - gazebo::common::SystemPaths, 756
- GetGazeboVersion
  - gazebo::common::LogPlay, 428
- GetGlobalAxis
  - gazebo::physics::Joint, 378
- GetGlobalPoints
  - gazebo::physics::RayShape, 624
- GetGravity
  - gazebo::physics::PhysicsEngine, 551
- GetGravityMode
  - gazebo::physics::Link, 412
- GetGrid
  - gazebo::rendering::Scene, 657
- GetGridCount
  - gazebo::rendering::Scene, 657
- GetHFOV
  - gazebo::rendering::Camera, 167
- GetHandle
  - gazebo::common::SkeletonNode, 712
  - gazebo::PluginT, 571
- GetHeader
  - Messages, 59
- GetHeight
  - gazebo::common::Image, 355
  - gazebo::common::Video, 844
  - gazebo::physics::HeightmapShape, 346
  - gazebo::rendering::Grid, 334
  - gazebo::rendering::Heightmap, 343
- GetHeightBelowPoint
  - gazebo::rendering::Scene, 657
- GetHeightmap
  - gazebo::rendering::Scene, 657
- GetHighStop
  - gazebo::physics::BallJoint, 133
  - gazebo::physics::Joint, 378
- GetHorzFOV
  - gazebo::sensors::GpuRaySensor, 325
- GetHorzHalfAngle
  - gazebo::sensors::GpuRaySensor, 325
- GetIO
  - gazebo::transport::IOManager, 370
- GetIXX
  - gazebo::physics::Inertial, 363
- GetIXY
  - gazebo::physics::Inertial, 363
- GetIXZ
  - gazebo::physics::Inertial, 363
- GetIYY
  - gazebo::physics::Inertial, 363
- GetIYZ
  - gazebo::physics::Inertial, 363
- GetIZZ
  - gazebo::physics::Inertial, 364
- GetId
  - gazebo::common::SkeletonNode, 712
  - gazebo::event::Connection, 217
  - gazebo::physics::Base, 139
  - gazebo::rendering::Scene, 658
  - gazebo::transport::CallbackHelper, 154
  - gazebo::transport::Connection, 220
  - gazebo::transport::Node, 517
- GetIdString
  - gazebo::rendering::Scene, 658
- GetImageByteSize
  - gazebo::rendering::Camera, 167
- GetImageData
  - gazebo::rendering::Camera, 168
  - gazebo::sensors::CameraSensor, 185
  - gazebo::sensors::MultiCameraSensor, 504
- GetImageDepth
  - gazebo::rendering::Camera, 168
- GetImageFormat
  - gazebo::rendering::Camera, 168
- GetImageHeight
  - gazebo::rendering::Camera, 168
  - gazebo::sensors::CameraSensor, 185
  - gazebo::sensors::MultiCameraSensor, 504
- GetImageWidth
  - gazebo::rendering::Camera, 168
  - gazebo::sensors::CameraSensor, 185
  - gazebo::sensors::MultiCameraSensor, 504
- GetInclude
  - sdf::Element, 271
- GetIndex
  - gazebo::common::SubMesh, 738
- GetIndexCount
  - gazebo::common::Mesh, 457
  - gazebo::common::SubMesh, 739
- GetInertial
  - gazebo::physics::Link, 412
- GetInitialRelativePose
  - gazebo::physics::Entity, 278



- GetInitialized
  - gazebo::rendering::Camera, 168
  - gazebo::Server, 687
- GetIntNormal
  - gazebo::math::Rand, 612
- GetIntUniform
  - gazebo::math::Rand, 612
- GetInterpolatedKeyFrame
  - gazebo::common::NumericAnimation, 531
  - gazebo::common::PoseAnimation, 582
- GetIntersection
  - gazebo::physics::RayShape, 624
- GetInverse
  - gazebo::math::Pose, 576
  - gazebo::math::Quaternion, 603
- GetInverseBindTransform
  - gazebo::common::SkeletonNode, 712
- GetJoint
  - gazebo::physics::Model, 473
- GetJointController
  - gazebo::physics::Model, 474
- GetJointCount
  - gazebo::physics::Model, 474
- GetJointLink
  - gazebo::physics::Joint, 378
- GetJointState
  - gazebo::physics::ModelState, 487
- GetJointStateCount
  - gazebo::physics::ModelState, 488
- GetJointStates
  - gazebo::physics::ModelState, 488
- GetJoints
  - gazebo::physics::Model, 474
- GetKey
  - sdf::Param, 540
- GetKeyFrame
  - gazebo::common::Animation, 123
  - gazebo::common::NodeAnimation, 522
- GetKeyFrameCount
  - gazebo::common::Animation, 123
- GetKeyFramesAtTime
  - gazebo::common::Animation, 123
- GetKinematic
  - gazebo::physics::Link, 412
- GetLabel
  - gazebo::common::DiagnosticManager, 255
- GetLaserCamera
  - gazebo::sensors::GpuRaySensor, 326
- GetLaserData
  - gazebo::rendering::GpuLaser, 319
- GetLaserRetro
  - gazebo::physics::Collision, 194
- GetLaserShape
  - gazebo::sensors::RaySensor, 618
- GetLastMeasurementTime
  - gazebo::sensors::Sensor, 675
- GetLastRenderWallTime
  - gazebo::rendering::Camera, 169
- GetLastUpdateTime
  - gazebo::sensors::Sensor, 675
- GetLatching
  - gazebo::transport::CallbackHelper, 154
  - gazebo::transport::Publisher, 595
  - gazebo::transport::SubscribeOptions, 744
- GetLength
  - gazebo::common::Animation, 124
  - gazebo::common::NodeAnimation, 523
  - gazebo::common::SkeletonAnimation, 705
  - gazebo::math::Vector3, 825
  - gazebo::math::Vector4, 836
  - gazebo::physics::CylinderShape, 244
  - gazebo::physics::RayShape, 625
- GetLight
  - gazebo::rendering::Scene, 658
- GetLightCount
  - gazebo::rendering::Scene, 658
- GetLighting
  - gazebo::common::Material, 439
- getLights
  - gazebo::rendering::MovableText, 498
- GetLineWidth
  - gazebo::rendering::Grid, 334
- GetLinearAcceleration
  - gazebo::sensors::ImuSensor, 359
- GetLinearDamping
  - gazebo::physics::Link, 413
- GetLink
  - gazebo::physics::Collision, 194
  - gazebo::physics::Model, 474
- GetLinkById
  - gazebo::physics::Model, 474
- GetLinkForce
  - gazebo::physics::Joint, 379
- GetLinkState
  - gazebo::physics::ModelState, 488
- GetLinkStateCount
  - gazebo::physics::ModelState, 489
- GetLinkStates
  - gazebo::physics::ModelState, 489
- GetLinkTorque
  - gazebo::physics::Joint, 379
- GetLinks
  - gazebo::physics::Model, 474
- GetLocalAddress
  - gazebo::transport::Connection, 221
- GetLocalAxis
  - gazebo::physics::Joint, 379
- GetLocalHostname

- gazebo::transport::Connection, 221
- GetLocalPort
  - gazebo::transport::Connection, 221
- GetLocalURI
  - gazebo::transport::Connection, 221
- GetLocallyAdvertised
  - gazebo::transport::Publication, 589
- GetLog
  - gazebo::math::Quaternion, 603
- GetLogPath
  - gazebo::common::SystemPaths, 756
- GetLogVersion
  - gazebo::common::LogPlay, 428
- GetLowStop
  - gazebo::physics::BallJoint, 133
  - gazebo::physics::Joint, 379
- GetManager
  - gazebo::rendering::Scene, 659
- GetManifest
  - Common, 34
- GetMass
  - gazebo::physics::Inertial, 364
- GetMaterial
  - gazebo::common::Mesh, 457
- getMaterial
  - gazebo::rendering::MovableText, 498
- GetMaterialCount
  - gazebo::common::Mesh, 457
- GetMaterialIndex
  - gazebo::common::SubMesh, 739
- GetMaterialName
  - gazebo::rendering::Visual, 858
- GetMax
  - gazebo::common::Mesh, 458
  - gazebo::common::SubMesh, 739
  - gazebo::math::Vector3, 825
- GetMaxAngle
  - gazebo::physics::MultiRayShape, 510
- GetMaxColor
  - gazebo::common::Image, 355
- GetMaxContacts
  - gazebo::physics::PhysicsEngine, 551
- GetMaxForce
  - gazebo::physics::Joint, 380
- GetMaxHeight
  - gazebo::physics::HeightmapShape, 346
- GetMaxIndex
  - gazebo::common::SubMesh, 739
- GetMaxRange
  - gazebo::physics::MultiRayShape, 510
- GetMesh
  - gazebo::common::MeshManager, 467
- GetMeshAABB
  - gazebo::common::MeshManager, 467
- GetMeshName
  - gazebo::rendering::Visual, 858
- GetMin
  - gazebo::common::Mesh, 458
  - gazebo::common::SubMesh, 739
  - gazebo::math::Vector3, 825
- GetMinAngle
  - gazebo::physics::MultiRayShape, 511
- GetMinHeight
  - gazebo::physics::HeightmapShape, 346
- GetMinRange
  - gazebo::physics::MultiRayShape, 511
- GetModel
  - gazebo::physics::Collision, 194
  - gazebo::physics::Link, 413
  - gazebo::physics::World, 879, 880
- GetModelBelowPoint
  - gazebo::physics::World, 880
- GetModelConfig
  - Common, 34
- GetModelCount
  - gazebo::physics::World, 880
- GetModelFile
  - Common, 34
- GetModelName
  - Common, 35
- GetModelPath
  - Common, 35
- GetModelPaths
  - gazebo::common::SystemPaths, 756
- GetModelState
  - gazebo::physics::WorldState, 890
- GetModelStateCount
  - gazebo::physics::WorldState, 890
- GetModelStates
  - gazebo::physics::WorldState, 890
- GetModelTransform
  - gazebo::common::SkeletonNode, 712
- GetModelVisualAt
  - gazebo::rendering::Scene, 659
- GetModels
  - Common, 35
  - gazebo::physics::World, 880
- GetMovableType
  - gazebo::rendering::DynamicLines, 261
- getMovableType
  - gazebo::rendering::DynamicLines, 261
- GetMsgType
  - gazebo::transport::CallbackHelper, 154
  - gazebo::transport::CallbackHelperT, 157
  - gazebo::transport::Node, 517
  - gazebo::transport::Publication, 590
  - gazebo::transport::PublicationTransport, 593
  - gazebo::transport::Publisher, 595

- gazebo::transport::RawCallbackHelper, 614
- gazebo::transport::SubscribeOptions, 744
- GetMsgTypes
  - gazebo::msgs::MsgFactory, 501
- GetName
  - gazebo::common::DiagnosticTimer, 258
  - gazebo::common::Material, 439
  - gazebo::common::Mesh, 458
  - gazebo::common::NodeAnimation, 523
  - gazebo::common::SkeletonAnimation, 706
  - gazebo::common::SkeletonNode, 713
  - gazebo::physics::Base, 139
  - gazebo::physics::State, 730
  - gazebo::physics::World, 881
  - gazebo::rendering::Camera, 169
  - gazebo::rendering::Light, 399
  - gazebo::rendering::Scene, 659
  - gazebo::rendering::Visual, 858
  - gazebo::sensors::Sensor, 675
  - sdf::Element, 271
- GetNearClip
  - gazebo::rendering::Camera, 169
- GetNearestEntityBelow
  - gazebo::physics::Entity, 278
- GetNextElement
  - sdf::Element, 271
- GetNextFrame
  - gazebo::common::Video, 844
- GetNode
  - gazebo::transport::SubscribeOptions, 744
- GetNodeAssignment
  - gazebo::common::SubMesh, 739
- GetNodeAssignmentsCount
  - gazebo::common::SubMesh, 739
- GetNodeByHandle
  - gazebo::common::Skeleton, 700
- GetNodeById
  - gazebo::common::Skeleton, 700
- GetNodeByName
  - gazebo::common::Skeleton, 701
- GetNodeCount
  - gazebo::common::SkeletonAnimation, 706
  - gazebo::transport::Publication, 590
- GetNodePoseAt
  - gazebo::common::SkeletonAnimation, 706
- GetNodes
  - gazebo::common::Skeleton, 701
- GetNormal
  - gazebo::common::SubMesh, 739
  - gazebo::math::Vector3, 826
  - gazebo::physics::PlaneShape, 567
- GetNormalCount
  - gazebo::common::Mesh, 458
  - gazebo::common::SubMesh, 740
- GetNormalMap
  - gazebo::rendering::Visual, 858
- GetNumAnimations
  - gazebo::common::Skeleton, 701
- GetNumJoints
  - gazebo::common::Skeleton, 701
- GetNumNodes
  - gazebo::common::Skeleton, 701
- GetNumPoints
  - gazebo::math::RotationSpline, 643
- GetNumRawTrans
  - gazebo::common::SkeletonNode, 713
- GetNumVertNodeWeights
  - gazebo::common::Skeleton, 701
- GetOgreCamera
  - gazebo::rendering::Camera, 169
- GetOgrePaths
  - gazebo::common::SystemPaths, 756
- GetOgreTerrain
  - gazebo::rendering::Heightmap, 343
- GetOperationType
  - gazebo::rendering::DynamicRenderable, 265
- GetOutgoingCount
  - gazebo::transport::Publisher, 596
- GetPSSMShadowCameraSetup
  - gazebo::rendering::RTShaderSystem, 649
- GetParent
  - gazebo::common::SkeletonNode, 713
  - gazebo::physics::Base, 139
  - gazebo::physics::Joint, 380
  - gazebo::rendering::Projector, 586
  - gazebo::rendering::Visual, 858
  - sdf::Element, 271
- GetParentId
  - gazebo::physics::Base, 139
- GetParentJointsLinks
  - gazebo::physics::Link, 413
- GetParentModel
  - gazebo::physics::Entity, 279
- GetParentName
  - gazebo::sensors::Sensor, 675
- GetPath
  - gazebo::common::Mesh, 458
- GetPauseTime
  - gazebo::physics::World, 881
- GetPerpendicular
  - gazebo::math::Vector3, 826
- GetPhysicsEngine
  - gazebo::physics::World, 881
- GetPhysicsUpdateMutex
  - gazebo::physics::PhysicsEngine, 551
- GetPitch
  - gazebo::common::Image, 355
  - gazebo::math::Quaternion, 603

- GetPitchNode
  - gazebo::rendering::Camera, 169
- GetPixel
  - gazebo::common::Image, 355
- GetPixelFormat
  - gazebo::common::Image, 356
- GetPluginCount
  - gazebo::physics::Model, 475
- GetPluginPaths
  - gazebo::common::SystemPaths, 757
- GetPoint
  - gazebo::math::RotationSpline, 643
  - gazebo::math::Spline, 725
  - gazebo::rendering::DynamicLines, 261
- GetPointCount
  - gazebo::math::Spline, 726
  - gazebo::rendering::DynamicLines, 261
- GetPointSize
  - gazebo::common::Material, 439
- GetPos
  - gazebo::physics::HeightmapShape, 346
- GetPose
  - gazebo::physics::CollisionState, 201
  - gazebo::physics::Inertial, 364
  - gazebo::physics::LinkState, 425
  - gazebo::physics::ModelState, 489
  - gazebo::rendering::Visual, 859
  - gazebo::sensors::Sensor, 675
- GetPoseAt
  - gazebo::common::SkeletonAnimation, 706
- GetPoseAtX
  - gazebo::common::SkeletonAnimation, 707
- GetPosition
  - gazebo::rendering::Light, 399
  - gazebo::rendering::Visual, 859
- GetPrevMsg
  - gazebo::transport::Publisher, 596
- GetPrimitiveType
  - gazebo::common::SubMesh, 740
- GetPrincipalMoments
  - gazebo::physics::Inertial, 364
- GetProductsofInertia
  - gazebo::physics::Inertial, 364
- GetRGBData
  - gazebo::common::Image, 356
- GetRadius
  - gazebo::physics::CylinderShape, 244
  - gazebo::physics::SphereShape, 723
- GetRandSeed
  - gazebo::common::LogPlay, 428
- GetRange
  - gazebo::physics::MultiRayShape, 511
  - gazebo::sensors::GpuRaySensor, 326
  - gazebo::sensors::RaySensor, 618
- GetRangeCount
  - gazebo::sensors::GpuRaySensor, 326
  - gazebo::sensors::RaySensor, 618
- GetRangeCountRatio
  - gazebo::sensors::GpuRaySensor, 326
- GetRangeMax
  - gazebo::sensors::GpuRaySensor, 326
  - gazebo::sensors::RaySensor, 618
- GetRangeMin
  - gazebo::sensors::GpuRaySensor, 327
  - gazebo::sensors::RaySensor, 618
- GetRangeResolution
  - gazebo::sensors::GpuRaySensor, 327
  - gazebo::sensors::RaySensor, 619
- GetRanges
  - gazebo::sensors::GpuRaySensor, 327
  - gazebo::sensors::RaySensor, 619
- GetRawTransform
  - gazebo::common::SkeletonNode, 713
- GetRawTransforms
  - gazebo::common::SkeletonNode, 713
- GetRayCount
  - gazebo::sensors::GpuRaySensor, 327
  - gazebo::sensors::RaySensor, 619
- GetRayCountRatio
  - gazebo::sensors::GpuRaySensor, 327
- GetRealTime
  - gazebo::physics::State, 730
  - gazebo::physics::World, 881
- GetRelativeAngularAccel
  - gazebo::physics::Collision, 194
  - gazebo::physics::Entity, 279
  - gazebo::physics::Link, 413
  - gazebo::physics::Model, 475
- GetRelativeAngularVel
  - gazebo::physics::Collision, 194
  - gazebo::physics::Entity, 279
  - gazebo::physics::Link, 413
  - gazebo::physics::Model, 475
- GetRelativeForce
  - gazebo::physics::Link, 413
- GetRelativeLinearAccel
  - gazebo::physics::Collision, 194
  - gazebo::physics::Entity, 279
  - gazebo::physics::Link, 414
  - gazebo::physics::Model, 475
- GetRelativeLinearVel
  - gazebo::physics::Collision, 195
  - gazebo::physics::Entity, 279
  - gazebo::physics::Link, 414
  - gazebo::physics::Model, 475
- GetRelativePoints
  - gazebo::physics::RayShape, 625
- GetRelativePose

- gazebo::physics::Entity, 280
- GetRelativeTorque
  - gazebo::physics::Link, 414
- GetRemoteAddress
  - gazebo::transport::Connection, 221
- GetRemoteHostname
  - gazebo::transport::Connection, 221
- GetRemotePort
  - gazebo::transport::Connection, 222
- GetRemoteSubscriptionCount
  - gazebo::transport::Publication, 590
- GetRemoteURI
  - gazebo::transport::Connection, 222
- getRenderOperation
  - gazebo::rendering::MovableText, 498
- GetRenderPathType
  - gazebo::rendering::RenderEngine, 630
- GetRenderRate
  - gazebo::rendering::Camera, 169
- GetRenderTexture
  - gazebo::rendering::Camera, 170
- GetRequired
  - sdf::Element, 271
  - sdf::Param, 540
- GetResRange
  - gazebo::physics::MultiRayShape, 511
- GetRetro
  - gazebo::physics::MultiRayShape, 511
  - gazebo::physics::RayShape, 625
  - gazebo::sensors::GpuRaySensor, 327
  - gazebo::sensors::RaySensor, 619
- GetRight
  - gazebo::rendering::Camera, 170
- GetRoll
  - gazebo::math::Quaternion, 603
- GetRootNode
  - gazebo::common::Skeleton, 702
- GetRootVisual
  - gazebo::rendering::Visual, 859
- GetRotation
  - gazebo::common::PoseKeyFrame, 584
  - gazebo::math::Matrix4, 450
  - gazebo::rendering::Visual, 859
- GetRounded
  - gazebo::math::Vector3, 826
- GetRunning
  - gazebo::common::LogRecord, 432
- GetSDF
  - gazebo::physics::Actor, 110
  - gazebo::physics::Base, 140
  - gazebo::physics::Model, 476
- GetSID
  - gazebo::common::NodeTransform, 527
- GetSORPGSIters
  - gazebo::physics::PhysicsEngine, 551
- GetSORPGSPreconIters
  - gazebo::physics::PhysicsEngine, 552
- GetSORPGSW
  - gazebo::physics::PhysicsEngine, 552
- GetSampleCount
  - gazebo::physics::MultiRayShape, 512
- GetSaveable
  - gazebo::physics::Base, 139
- GetScale
  - gazebo::rendering::Visual, 859
- GetScanResolution
  - gazebo::physics::MultiRayShape, 512
- GetScene
  - gazebo::rendering::Camera, 170
  - gazebo::rendering::RenderEngine, 630
  - gazebo::rendering::Visual, 859
- GetSceneCount
  - gazebo::rendering::RenderEngine, 630
- GetSceneNode
  - gazebo::rendering::Camera, 170
  - gazebo::rendering::Grid, 334
  - gazebo::rendering::Visual, 860
- GetScopedName
  - gazebo::physics::Base, 140
  - gazebo::sensors::Sensor, 676
- GetSeed
  - gazebo::math::Rand, 612
- GetSelectedEntity
  - gazebo::physics::World, 881
- GetSelectedVisual
  - gazebo::rendering::Scene, 659
- GetSelfCollide
  - gazebo::physics::Link, 414
- GetSensor
  - gazebo::sensors::SensorManager, 683
- GetSensorCount
  - gazebo::physics::Link, 414
  - gazebo::physics::Model, 476
- GetSensorName
  - gazebo::physics::Link, 414
- GetSensorTypes
  - gazebo::sensors::SensorFactory, 681
  - gazebo::sensors::SensorManager, 684
- GetSensors
  - gazebo::sensors::SensorManager, 684
- GetSet
  - sdf::Param, 540
- GetSetWorldPoseMutex
  - gazebo::physics::World, 881
- GetShadeMode
  - gazebo::common::Material, 439
- GetShaderType
  - gazebo::rendering::Visual, 860

- GetShadowsEnabled
  - gazebo::rendering::Scene, 659
- GetShape
  - gazebo::physics::Collision, 195
- GetShapeType
  - gazebo::physics::Collision, 195
- GetShininess
  - gazebo::common::Material, 439
- GetShowOnTop
  - gazebo::rendering::MovableText, 498
- GetSimTime
  - gazebo::physics::State, 731
  - gazebo::physics::World, 882
- GetSize
  - gazebo::math::Box, 146
  - gazebo::physics::BoxShape, 151
  - gazebo::physics::HeightmapShape, 347
  - gazebo::physics::PlaneShape, 567
  - gazebo::physics::TrimeshShape, 791
- GetSkeleton
  - gazebo::common::Mesh, 458
- GetSpaceWidth
  - gazebo::rendering::MovableText, 498
- GetSpecular
  - gazebo::common::Material, 439
- GetSpecularColor
  - gazebo::rendering::Light, 400
- GetSquaredLength
  - gazebo::math::Vector3, 826
  - gazebo::math::Vector4, 836
- getSquaredViewDepth
  - gazebo::rendering::DynamicRenderable, 265
  - gazebo::rendering::MovableText, 498
- GetStartTime
  - gazebo::physics::World, 882
- GetState
  - gazebo::physics::Collision, 195
- GetStepTime
  - gazebo::physics::PhysicsEngine, 552
- GetSubMesh
  - gazebo::common::Mesh, 459
- GetSubMeshCount
  - gazebo::common::Mesh, 459
- GetSubSampling
  - gazebo::physics::HeightmapShape, 347
- GetSum
  - gazebo::math::Vector3, 826
- GetSurface
  - gazebo::physics::Collision, 195
- GetTagPose
  - gazebo::sensors::RFIDTag, 635
- GetTangent
  - gazebo::math::Spline, 726
- GetTension
  - gazebo::math::Spline, 726
- GetTexCoord
  - gazebo::common::SubMesh, 740
- GetTexCoordCount
  - gazebo::common::Mesh, 459
  - gazebo::common::SubMesh, 740
- GetText
  - gazebo::rendering::MovableText, 498
- GetTextureHeight
  - gazebo::rendering::Camera, 170
- GetTextureImage
  - gazebo::common::Material, 440
- GetTextureWidth
  - gazebo::rendering::Camera, 170
- GetTime
  - gazebo::common::Animation, 124
  - gazebo::common::DiagnosticManager, 255, 256
  - gazebo::common::KeyFrame, 395
- GetTimeAtX
  - gazebo::common::NodeAnimation, 523
- GetTimerCount
  - gazebo::common::DiagnosticManager, 256
- GetTopic
  - gazebo::sensors::CameraSensor, 185
  - gazebo::sensors::MultiCameraSensor, 505
  - gazebo::sensors::RaySensor, 620
  - gazebo::sensors::Sensor, 676
  - gazebo::transport::PublicationTransport, 593
  - gazebo::transport::Publisher, 596
  - gazebo::transport::SubscribeOptions, 744
  - gazebo::transport::Subscriber, 746
- getTopicMsgType
  - Transport, 76
- GetTopicNamespace
  - gazebo::transport::Node, 517
- GetTopicNamespaces
  - gazebo::transport::ConnectionManager, 226
  - gazebo::transport::TopicManager, 785
- GetTransform
  - gazebo::common::SkeletonNode, 714
- GetTransforms
  - gazebo::common::SkeletonNode, 714
- GetTranslation
  - gazebo::common::PoseKeyFrame, 584
  - gazebo::math::Matrix4, 450
- GetTransparency
  - gazebo::common::Material, 440
  - gazebo::rendering::Visual, 860
- GetTransportCount
  - gazebo::transport::Publication, 590
- GetTriangleCount
  - gazebo::rendering::Camera, 171
  - gazebo::rendering::UserCamera, 799
  - gazebo::rendering::WindowManager, 872

- GetType
  - gazebo::common::NodeTransform, 527
  - gazebo::physics::Base, 140
  - gazebo::PluginT, 571
  - gazebo::rendering::Light, 400
  - gazebo::sensors::Sensor, 676
- GetTypeName
  - sdf::Param, 540
- GetTypeString
  - gazebo::rendering::FPSViewController, 314
  - gazebo::rendering::OrbitViewController, 535
  - gazebo::rendering::ViewController, 847
- GetURI
  - Common, 36
  - gazebo::physics::HeightmapShape, 347
- GetUp
  - gazebo::rendering::Camera, 171
- GetUpdatePeriod
  - gazebo::physics::PhysicsEngine, 552
- GetUpdateRate
  - gazebo::physics::PhysicsEngine, 552
  - gazebo::sensors::Sensor, 676
- GetUserCamera
  - gazebo::rendering::Scene, 659
- GetUserCameraCount
  - gazebo::rendering::Scene, 660
- GetVFOV
  - gazebo::rendering::Camera, 171
- GetValue
  - gazebo::common::NumericKeyFrame, 533
  - sdf::Element, 271
  - sdf::ParamT, 545
- GetValueBool
  - sdf::Element, 271
- GetValueChar
  - sdf::Element, 271
- GetValueColor
  - sdf::Element, 271
- GetValueDouble
  - sdf::Element, 271
- GetValueFloat
  - sdf::Element, 271
- GetValueInt
  - sdf::Element, 271
- GetValuePose
  - sdf::Element, 271
- GetValueQuaternion
  - sdf::Element, 271
- GetValueString
  - sdf::Element, 271
- GetValueTime
  - sdf::Element, 271
- GetValueUInt
  - sdf::Element, 271
- GetValueVector2d
  - sdf::Element, 271
- GetValueVector3
  - sdf::Element, 271
- GetVelocity
  - gazebo::physics::Joint, 380
  - gazebo::physics::LinkState, 425
- GetVertFOV
  - gazebo::sensors::GpuRaySensor, 328
- GetVertHalfAngle
  - gazebo::sensors::GpuRaySensor, 328
- GetVertNodeWeight
  - gazebo::common::Skeleton, 702
- GetVertex
  - gazebo::common::SubMesh, 740
- GetVertexCount
  - gazebo::common::Mesh, 459
  - gazebo::common::SubMesh, 740
  - gazebo::physics::HeightmapShape, 347
- GetVertexIndex
  - gazebo::common::SubMesh, 741
- GetVerticalAngleMax
  - gazebo::sensors::GpuRaySensor, 328
  - gazebo::sensors::RaySensor, 620
- GetVerticalAngleMin
  - gazebo::sensors::GpuRaySensor, 328
  - gazebo::sensors::RaySensor, 620
- GetVerticalMaxAngle
  - gazebo::physics::MultiRayShape, 512
- GetVerticalMinAngle
  - gazebo::physics::MultiRayShape, 512
- GetVerticalRangeCount
  - gazebo::sensors::GpuRaySensor, 328
  - gazebo::sensors::RaySensor, 620
- GetVerticalRayCount
  - gazebo::sensors::GpuRaySensor, 328
  - gazebo::sensors::RaySensor, 620
- GetVerticalSampleCount
  - gazebo::physics::MultiRayShape, 512
- GetVerticalScanResolution
  - gazebo::physics::MultiRayShape, 512
- GetViewControllerTypeString
  - gazebo::rendering::UserCamera, 799
- GetViewport
  - gazebo::rendering::Camera, 171
- GetViewportHeight
  - gazebo::rendering::Camera, 171
- GetViewportWidth
  - gazebo::rendering::Camera, 171
- GetVisibilityFlags
  - gazebo::rendering::Visual, 860
- GetVisible
  - gazebo::rendering::Visual, 860
- GetVisual

- gazebo::rendering::Scene, 660
- gazebo::rendering::UserCamera, 799
- GetVisualAt
  - gazebo::rendering::Scene, 660
- GetVisualBelow
  - gazebo::rendering::Scene, 661
- GetVisualName
  - gazebo::rendering::SelectionObj, 670
- GetVisualize
  - gazebo::sensors::Sensor, 676
- GetVisualsBelowPoint
  - gazebo::rendering::Scene, 661
- GetWallTime
  - gazebo::common::Time, 764
  - gazebo::physics::State, 731
- GetWidth
  - gazebo::common::Image, 356
  - gazebo::common::Video, 844
- GetWindow
  - gazebo::rendering::WindowManager, 872
- GetWindowId
  - gazebo::rendering::Camera, 172
- GetWorld
  - gazebo::physics::Base, 140
- GetWorldAngularAccel
  - gazebo::physics::Collision, 196
  - gazebo::physics::Entity, 280
  - gazebo::physics::Link, 415
  - gazebo::physics::Model, 476
- GetWorldAngularVel
  - gazebo::physics::Collision, 196
  - gazebo::physics::Entity, 280
  - gazebo::physics::Model, 476
- GetWorldCFM
  - gazebo::physics::PhysicsEngine, 552
- GetWorldERP
  - gazebo::physics::PhysicsEngine, 553
- GetWorldForce
  - gazebo::physics::Link, 415
- GetWorldLinearAccel
  - gazebo::physics::Collision, 196
  - gazebo::physics::Entity, 280
  - gazebo::physics::Link, 415
  - gazebo::physics::Model, 476
- GetWorldLinearVel
  - gazebo::physics::Collision, 196
  - gazebo::physics::Entity, 280
  - gazebo::physics::Model, 477
- GetWorldName
  - gazebo::sensors::Sensor, 676
- GetWorldPathExtension
  - gazebo::common::SystemPaths, 757
- GetWorldPointOnPlane
  - gazebo::rendering::Camera, 172
- GetWorldPose
  - gazebo::physics::Entity, 281
  - gazebo::rendering::Camera, 172
  - gazebo::rendering::Visual, 861
- GetWorldPosition
  - gazebo::rendering::Camera, 172
- GetWorldRotation
  - gazebo::rendering::Camera, 172
- GetWorldTorque
  - gazebo::physics::Link, 415
- getWorldTransforms
  - gazebo::rendering::MovableText, 498
- GetWorldVisual
  - gazebo::rendering::Scene, 661
- GetWrench
  - gazebo::physics::LinkState, 425
- GetXAxis
  - gazebo::math::Quaternion, 604
- GetXLength
  - gazebo::math::Box, 146
- GetYAxis
  - gazebo::math::Quaternion, 604
- GetYLength
  - gazebo::math::Box, 147
- GetYaw
  - gazebo::math::Quaternion, 604
- GetZAxis
  - gazebo::math::Quaternion, 604
- GetZLength
  - gazebo::math::Box, 147
- GetZValue
  - gazebo::rendering::Camera, 173
- globalEndPos
  - gazebo::physics::RayShape, 627
- globalStartPos
  - gazebo::physics::RayShape, 627
- google, 102
- google::protobuf, 102
- google::protobuf::compiler, 102
- google::protobuf::compiler::cpp, 102
- google::protobuf::compiler::cpp::GazeboGenerator, 315
  - ~GazeboGenerator, 316
  - GazeboGenerator, 316
  - Generate, 316
- GpuLaser
  - gazebo::rendering::GpuLaser, 318
- GpuLaser.hh, 944
- GpuLaserPtr
  - gazebo::rendering, 97
- GpuRaySensor
  - gazebo::sensors::GpuRaySensor, 323
- GpuRaySensor.hh, 945
- GpuRaySensor\_V
  - gazebo::sensors, 100



- GpuRaySensorPtr
  - gazebo::sensors, 100
- Green
  - gazebo::common::Color, 213
- Grid
  - gazebo::rendering::Grid, 333
- Grid.hh, 946
- Gripper
  - gazebo::physics::Gripper, 336
- Gripper.hh, 946
- GtsSurface
  - MeshCSG.hh, 981
- GzTerrainMatGen
  - gazebo::rendering::GzTerrainMatGen, 342
- gzclr\_end
  - Common, 31
- gzclr\_start
  - Common, 31
- gzdbg
  - Common, 31
- gzerr
  - Common, 31
- gzlog
  - Common, 31
- gzmsg
  - Common, 31
- gzthrow
  - Common, 32
- gzwarn
  - Common, 32
  
- H\_CENTER
  - gazebo::rendering::MovableText, 496
- H\_LEFT
  - gazebo::rendering::MovableText, 496
- HEADER\_LENGTH
  - Connection.hh, 922
- HEIGHTMAP\_SHAPE
  - gazebo::physics::Base, 137
- HI\_STOP
  - gazebo::physics::Joint, 374
- HINGE2\_JOINT
  - gazebo::physics::Base, 137
- HINGE\_JOINT
  - gazebo::physics::Base, 137
- handle
  - gazebo::common::SkeletonNode, 716
  - gazebo::PluginT, 571
- HandleData
  - gazebo::transport::CallbackHelper, 155
  - gazebo::transport::CallbackHelperT, 157
  - gazebo::transport::Node, 517
  - gazebo::transport::RawCallbackHelper, 614
  - gazebo::transport::SubscriptionTransport, 748
- HandleKeyPressEvent
  - gazebo::rendering::FPSViewController, 314
  - gazebo::rendering::GUIOverlay, 339
  - gazebo::rendering::OrbitViewController, 535
  - gazebo::rendering::UserCamera, 799
  - gazebo::rendering::ViewController, 847
- HandleKeyReleaseEvent
  - gazebo::rendering::FPSViewController, 314
  - gazebo::rendering::GUIOverlay, 340
  - gazebo::rendering::OrbitViewController, 535
  - gazebo::rendering::UserCamera, 800
  - gazebo::rendering::ViewController, 848
- HandleMouseEvent
  - gazebo::rendering::FPSViewController, 314
  - gazebo::rendering::GUIOverlay, 340
  - gazebo::rendering::OrbitViewController, 536
  - gazebo::rendering::UserCamera, 800
  - gazebo::rendering::ViewController, 848
- HasAttachedObject
  - gazebo::rendering::Visual, 861
- HasAttribute
  - sdf::Element, 271
- HasConnections
  - gazebo::transport::Publisher, 596
- HasElement
  - sdf::Element, 272
- HasElementDescription
  - sdf::Element, 272
- HasJointState
  - gazebo::physics::ModelState, 489
- HasLatchedSubscriber
  - gazebo::transport::Node, 518
- HasLinkState
  - gazebo::physics::ModelState, 490
- HasMesh
  - gazebo::common::MeshManager, 467
- HasModel
  - Common, 36
- HasModelState
  - gazebo::physics::WorldState, 891
- HasNode
  - gazebo::common::SkeletonAnimation, 707
- HasSkeleton
  - gazebo::common::Mesh, 459
- HasTransport
  - gazebo::transport::Publication, 590
- HasType
  - gazebo::physics::Base, 140
- HasVertex
  - gazebo::common::SubMesh, 741
- Heightmap
  - gazebo::rendering::Heightmap, 343
- Heightmap.hh, 948
- HeightmapShape

- gazebo::physics::HeightmapShape, 345
- HeightmapShape.hh, 949
- HeightmapShapePtr
  - gazebo::physics, 93
- heights
  - gazebo::physics::HeightmapShape, 348
- Helpers.hh, 950
  - GZ\_DBL\_MAX, 952
  - GZ\_DBL\_MIN, 952
  - GZ\_FLT\_MAX, 952
  - GZ\_FLT\_MIN, 952
- hfov
  - gazebo::sensors::GpuRaySensor, 331
- Hide
  - gazebo::rendering::GUIOverlay, 340
- Hinge2Joint
  - gazebo::physics::Hinge2Joint, 349
- Hinge2Joint.hh, 952
- HingeJoint
  - gazebo::physics::HingeJoint, 351
- HingeJoint.hh, 954
- HorizAlign
  - gazebo::rendering::MovableText, 496
- horzElem
  - gazebo::physics::MultiRayShape, 513
  - gazebo::sensors::GpuRaySensor, 331
- horzHalfAngle
  - gazebo::sensors::GpuRaySensor, 331
- horzRangeCount
  - gazebo::sensors::GpuRaySensor, 331
- horzRayCount
  - gazebo::sensors::GpuRaySensor, 331
- IDENTITY
  - gazebo::math::Matrix4, 454
- INTERSECTION
  - gazebo::common::MeshCSG, 461
- IOManager
  - gazebo::transport::IOManager, 370
- IOManager.hh, 958
- id
  - gazebo::common::SkeletonNode, 716
  - gazebo::physics::TrajectoryInfo, 788
- Image
  - gazebo::common::Image, 354
- Image.hh, 955
- imageFormat
  - gazebo::rendering::Camera, 181
- imageHeight
  - gazebo::rendering::Camera, 181
- imageWidth
  - gazebo::rendering::Camera, 181
- img
  - gazebo::physics::HeightmapShape, 348
- ImuSensor
  - gazebo::sensors::ImuSensor, 359
- ImuSensor.hh, 956
- IncCount
  - gazebo::transport::IOManager, 370
- indexBufferCapacity
  - gazebo::rendering::DynamicRenderable, 266
- Inertial
  - gazebo::physics::Inertial, 362
- inertial
  - gazebo::physics::Link, 421
- Inertial.hh, 957
- InertialPtr
  - gazebo::physics, 93
- Init
  - Common, 36
  - gazebo::common::LogRecord, 432
  - gazebo::common::PID, 560
  - gazebo::Master, 434
  - gazebo::ModelPlugin, 484
  - gazebo::physics::Actor, 111
  - gazebo::physics::Base, 141
  - gazebo::physics::BoxShape, 151
  - gazebo::physics::Collision, 196
  - gazebo::physics::ContactManager, 233
  - gazebo::physics::CylinderShape, 245
  - gazebo::physics::Gripper, 337
  - gazebo::physics::HeightmapShape, 347
  - gazebo::physics::HingeJoint, 351
  - gazebo::physics::Joint, 380
  - gazebo::physics::Link, 416
  - gazebo::physics::Model, 477
  - gazebo::physics::MultiRayShape, 512
  - gazebo::physics::PhysicsEngine, 553
  - gazebo::physics::PlaneShape, 567
  - gazebo::physics::RayShape, 625
  - gazebo::physics::Road, 641
  - gazebo::physics::Shape, 694
  - gazebo::physics::SphereShape, 723
  - gazebo::physics::TrimeshShape, 791
  - gazebo::physics::World, 882
  - gazebo::rendering::Camera, 173
  - gazebo::rendering::DepthCamera, 249
  - gazebo::rendering::DynamicRenderable, 265
  - gazebo::rendering::FPSViewController, 315
  - gazebo::rendering::GpuLaser, 319
  - gazebo::rendering::Grid, 335
  - gazebo::rendering::GUIOverlay, 340
  - gazebo::rendering::OrbitViewController, 536
  - gazebo::rendering::RenderEngine, 631
  - gazebo::rendering::RTShaderSystem, 649
  - gazebo::rendering::Scene, 661
  - gazebo::rendering::SelectionObj, 670
  - gazebo::rendering::UserCamera, 800

- gazebo::rendering::ViewController, 848
- gazebo::rendering::Visual, 861
- gazebo::rendering::WireBox, 874
- gazebo::SensorPlugin, 686
- gazebo::sensors::CameraSensor, 185
- gazebo::sensors::ContactSensor, 237
- gazebo::sensors::DepthCameraSensor, 252
- gazebo::sensors::GpuRaySensor, 329
- gazebo::sensors::ImuSensor, 359
- gazebo::sensors::MultiCameraSensor, 505
- gazebo::sensors::RaySensor, 620
- gazebo::sensors::RFIDSensor, 633
- gazebo::sensors::RFIDTag, 636
- gazebo::sensors::Sensor, 677
- gazebo::sensors::SensorManager, 684
- gazebo::Server, 687
- gazebo::SystemPlugin, 759
- gazebo::transport::ConnectionManager, 226
- gazebo::transport::Node, 518
- gazebo::transport::PublicationTransport, 594
- gazebo::transport::SubscribeOptions, 745
- gazebo::transport::SubscriptionTransport, 748
- gazebo::transport::TopicManager, 786
- gazebo::VisualPlugin, 870
- gazebo::WorldPlugin, 887
- Messages, 60
- init
  - gazebo, 81
  - Rendering, 66
  - sdf, 104
  - Sensors, 71
  - Transport, 76
- init\_world
  - Classes for physics and dynamics, 45
- init\_worlds
  - Classes for physics and dynamics, 45
- initDoc
  - sdf, 104
- initFile
  - sdf, 104
- InitForThread
  - gazebo::physics::PhysicsEngine, 553
- InitModelDoc
  - urdf2gazebo::URDF2Gazebo, 794
- InitModelFile
  - urdf2gazebo::URDF2Gazebo, 794
- InitModelString
  - urdf2gazebo::URDF2Gazebo, 794
- initString
  - sdf, 104
- initXml
  - sdf, 104
- initialTransform
  - gazebo::common::SkeletonNode, 716
- initialized
  - gazebo::rendering::Camera, 181
- InsertElement
  - sdf::Element, 272
- InsertLatchedMsg
  - gazebo::transport::Node, 518
- InsertMesh
  - gazebo::rendering::Visual, 861
- InsertModelFile
  - gazebo::physics::World, 882
- InsertModelSDF
  - gazebo::physics::World, 882
- InsertModelString
  - gazebo::physics::World, 882
- Instance
  - SingletonT, 697
- InternalError
  - gazebo::common::InternalError, 369
- Interpolate
  - gazebo::math::RotationSpline, 644
  - gazebo::math::Spline, 726
- interpolateX
  - gazebo::physics::Actor, 112
- invBindTransform
  - gazebo::common::SkeletonNode, 716
- Inverse
  - gazebo::math::Matrix4, 450
- Invert
  - gazebo::math::Quaternion, 604
- is\_stopped
  - Transport, 76
- IsActive
  - gazebo::physics::Actor, 111
  - gazebo::rendering::SelectionObj, 670
  - gazebo::sensors::CameraSensor, 185
  - gazebo::sensors::ContactSensor, 237
  - gazebo::sensors::RaySensor, 620
  - gazebo::sensors::Sensor, 677
- IsAdvertised
  - gazebo::transport::TopicManager, 786
- IsAffine
  - gazebo::math::Matrix4, 450
- IsAnimating
  - gazebo::rendering::Camera, 173
- IsBool
  - sdf::Param, 540
- IsCanonicalLink
  - gazebo::physics::Entity, 281
- IsChar
  - sdf::Param, 540
- IsColor
  - sdf::Param, 540
- IsDouble
  - sdf::Param, 541

- IsFinite
  - gazebo::math::Pose, 576
  - gazebo::math::Quaternion, 604
  - gazebo::math::Vector2d, 806
  - gazebo::math::Vector2i, 814
  - gazebo::math::Vector3, 827
  - gazebo::math::Vector4, 836
- IsFloat
  - sdf::Param, 541
- IsHorizontal
  - gazebo::sensors::GpuRaySensor, 329
- isHorizontal
  - gazebo::sensors::GpuRaySensor, 331
- IsInitialized
  - gazebo::rendering::Camera, 173
  - gazebo::rendering::GUIOverlay, 340
- IsInt
  - sdf::Param, 541
- IsJoint
  - gazebo::common::SkeletonNode, 714
- IsLoaded
  - gazebo::physics::World, 883
- IsLocal
  - gazebo::transport::CallbackHelper, 155
  - gazebo::transport::CallbackHelperT, 157
  - gazebo::transport::RawCallbackHelper, 614
  - gazebo::transport::SubscriptionTransport, 748
- IsOpen
  - gazebo::common::LogPlay, 429
  - gazebo::transport::Connection, 222
- IsPaused
  - gazebo::physics::World, 883
- IsPlaceable
  - gazebo::physics::Collision, 196
- IsPlane
  - gazebo::rendering::Visual, 861
- IsPose
  - sdf::Param, 541
- isPowerOfTwo
  - Math, 51
- IsQuaternion
  - sdf::Param, 541
- IsRootNode
  - gazebo::common::SkeletonNode, 714
- IsRunning
  - gazebo::transport::ConnectionManager, 226
- IsSelected
  - gazebo::physics::Base, 141
- IsStatic
  - gazebo::physics::Entity, 281
  - gazebo::rendering::Visual, 862
- IsStr
  - sdf::Param, 541
- IsTime
  - sdf::Param, 541
- IsUInt
  - sdf::Param, 541
- IsValidFilename
  - gazebo::common::MeshManager, 467
- IsVector2d
  - sdf::Param, 541
- IsVector2i
  - sdf::Param, 541
- IsVector3
  - sdf::Param, 541
- IsVisible
  - gazebo::rendering::Camera, 173, 174
- IsZero
  - gazebo::physics::CollisionState, 201
  - gazebo::physics::JointState, 389
  - gazebo::physics::LinkState, 425
  - gazebo::physics::ModelState, 490
  - gazebo::physics::WorldState, 891
- isnan
  - Math, 50
- JOINT
  - gazebo::common::SkeletonNode, 710
  - gazebo::physics::Base, 137
- Joint
  - gazebo::physics::Joint, 375
- Joint.hh, 959
- Joint\_V
  - gazebo::physics, 93
- JointController
  - gazebo::physics::JointController, 386
- JointController.hh, 960
- JointController\_V
  - gazebo::physics, 93
- JointControllerPtr
  - gazebo::physics, 93
- JointPtr
  - gazebo::physics, 93
- JointState
  - gazebo::physics::JointState, 388
- JointState.hh, 961
- JointVisual
  - gazebo::rendering::JointVisual, 392
- JointVisual.hh, 962
- JointVisualPtr
  - gazebo::rendering, 97
- JointWrench.hh, 963
- kd
  - gazebo::physics::SurfaceParams, 751
- key
  - sdf::Param, 542
- KeyFrame
  - gazebo::common::KeyFrame, 395

- KeyFrame.hh, 964
- KeyFrame\_V
  - gazebo::common::Animation, 122
- keyFrames
  - gazebo::common::Animation, 124
  - gazebo::common::NodeAnimation, 524
- kp
  - gazebo::physics::SurfaceParams, 751
- L\_INT16
  - gazebo::common::Image, 353
- L\_INT8
  - gazebo::common::Image, 353
- LEFT
  - gazebo::common::MouseEvent, 493
- LIGHT
  - gazebo::physics::Base, 137
- LINE\_MAX\_LEN
  - STLLoader.hh, 1061
- LINES
  - gazebo::common::SubMesh, 736
- LINESTRIPS
  - gazebo::common::SubMesh, 736
- LINK
  - gazebo::physics::Base, 136
- LINUX
  - SystemPaths.hh, 1068
- LO\_STOP
  - gazebo::physics::Joint, 374
- LaserVisual
  - gazebo::rendering::LaserVisual, 396
- LaserVisual.hh, 965
- LaserVisualPtr
  - gazebo::rendering, 97
- lastMeasurementTime
  - gazebo::sensors::Sensor, 679
- lastPos
  - gazebo::physics::Actor, 112
- lastRenderWallTime
  - gazebo::rendering::Camera, 181
- lastScriptTime
  - gazebo::physics::Actor, 112
- lastTraj
  - gazebo::physics::Actor, 112
- lastUpdateTime
  - gazebo::sensors::Sensor, 679
- latching
  - gazebo::transport::CallbackHelper, 155
- length
  - gazebo::common::Animation, 125
  - gazebo::common::NodeAnimation, 524
  - gazebo::common::SkeletonAnimation, 707
- Light
  - gazebo::rendering::Light, 399
- Light.hh, 966
- LightFromSDF
  - Messages, 60
- LightPtr
  - gazebo::rendering, 97
- LightingModel
  - gazebo::rendering::RTShaderSystem, 647
- linearAccel
  - gazebo::physics::Link, 421
- Link
  - gazebo::physics::Link, 408
- link
  - gazebo::physics::Collision, 198
- Link.hh, 967
- Link\_V
  - gazebo::physics, 93
- LinkPtr
  - gazebo::physics, 93
- LinkState
  - gazebo::physics::LinkState, 423
- LinkState.hh, 968
- Listen
  - gazebo::transport::Connection, 222
- Load
  - gazebo::common::BVHLoader, 152
  - gazebo::common::ColladaLoader, 189
  - gazebo::common::Image, 356
  - gazebo::common::MeshLoader, 463
  - gazebo::common::MeshManager, 468
  - gazebo::common::STLLoader, 733
  - gazebo::common::Video, 844
  - gazebo::ModelPlugin, 484
  - gazebo::physics::Actor, 111
  - gazebo::physics::BallJoint, 133
  - gazebo::physics::Base, 141
  - gazebo::physics::Collision, 197
  - gazebo::physics::CollisionState, 201
  - gazebo::physics::Entity, 281
  - gazebo::physics::Gripper, 337
  - gazebo::physics::HeightmapShape, 347
  - gazebo::physics::Hinge2Joint, 350
  - gazebo::physics::HingeJoint, 351
  - gazebo::physics::Inertial, 364
  - gazebo::physics::Joint, 380, 381
  - gazebo::physics::JointState, 390
  - gazebo::physics::Link, 416
  - gazebo::physics::LinkState, 425
  - gazebo::physics::Model, 477
  - gazebo::physics::ModelState, 490
  - gazebo::physics::PhysicsEngine, 553
  - gazebo::physics::Road, 641
  - gazebo::physics::ScrewJoint, 667
  - gazebo::physics::SliderJoint, 719
  - gazebo::physics::State, 731

- gazebo::physics::SurfaceParams, 750
- gazebo::physics::UniversalJoint, 793
- gazebo::physics::World, 883
- gazebo::physics::WorldState, 891
- gazebo::rendering::ArrowVisual, 127
- gazebo::rendering::AxisVisual, 130
- gazebo::rendering::Camera, 174
- gazebo::rendering::CameraVisual, 188
- gazebo::rendering::COMVisual, 215
- gazebo::rendering::DepthCamera, 249
- gazebo::rendering::GpuLaser, 319
- gazebo::rendering::Heightmap, 343
- gazebo::rendering::JointVisual, 392
- gazebo::rendering::Light, 400
- gazebo::rendering::MovableText, 499
- gazebo::rendering::Projector, 586
- gazebo::rendering::RenderEngine, 631
- gazebo::rendering::Road2d, 642
- gazebo::rendering::Scene, 661
- gazebo::rendering::UserCamera, 800
- gazebo::rendering::Visual, 862
- gazebo::SensorPlugin, 686
- gazebo::sensors::CameraSensor, 186
- gazebo::sensors::ContactSensor, 238
- gazebo::sensors::DepthCameraSensor, 252
- gazebo::sensors::GpuRaySensor, 329
- gazebo::sensors::ImuSensor, 359, 360
- gazebo::sensors::MultiCameraSensor, 505
- gazebo::sensors::RaySensor, 621
- gazebo::sensors::RFIDSensor, 633
- gazebo::sensors::RFIDTag, 636
- gazebo::sensors::Sensor, 677
- gazebo::SystemPlugin, 759
- gazebo::VisualPlugin, 870
- gazebo::WorldPlugin, 887
- load
  - Classes for physics and dynamics, 45
  - gazebo, 81
  - Rendering, 66
  - Sensors, 72
- load\_world
  - Classes for physics and dynamics, 45
- load\_worlds
  - Classes for physics and dynamics, 45
- LoadFile
  - gazebo::Server, 687
- LoadFromMsg
  - gazebo::rendering::Heightmap, 343
  - gazebo::rendering::Light, 400
  - gazebo::rendering::Visual, 862
- LoadJoints
  - gazebo::physics::Model, 477
- LoadLayout
  - gazebo::rendering::GUIOverlay, 340
- LoadPlugin
  - gazebo::physics::World, 883
  - gazebo::rendering::Visual, 862
- LoadPlugins
  - gazebo::physics::Model, 477
- LoadString
  - gazebo::Server, 687
- LocalPublish
  - gazebo::transport::Publication, 591
- Log
  - Common, 36
- LogPlay.hh, 970
- LogRecord.hh, 971
  - GZ\_LOG\_VERSION, 973
- Logplay, 429
- loop
  - gazebo::common::Animation, 125
  - gazebo::physics::Actor, 112
- m
  - gazebo::math::Matrix3, 447
  - gazebo::math::Matrix4, 454
- MAP\_SHAPE
  - gazebo::physics::Base, 137
- MATRIX
  - gazebo::common::NodeTransform, 526
- MAX\_COLLIDE\_RETURNS
  - Contact.hh, 926
- MAX\_CONTACT\_JOINTS
  - Contact.hh, 926
- MIDDLE
  - gazebo::common::MouseEvent, 493
- MODEL
  - gazebo::physics::Base, 136
- MODEL\_PLUGIN
  - Common, 32
- MODULATE
  - gazebo::common::Material, 437
- MOVE
  - gazebo::common::MouseEvent, 493
- MSleep
  - gazebo::common::Time, 765
- MULTIRAY\_SHAPE
  - gazebo::physics::Base, 137
- mainLink
  - gazebo::physics::Actor, 112
- mainpage.html, 973
- MakeStatic
  - gazebo::rendering::Visual, 862
- MapShape.hh, 973
- Master
  - gazebo::Master, 433
- Master.hh, 974
- Material

- gazebo::common::Material, 437
- Material.hh, 975, 977
- 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, 977
- Matrix3
  - gazebo::math::Matrix3, 445
- Matrix3.hh, 978
- Matrix4
  - gazebo::math::Matrix4, 449
- Matrix4.hh, 978
- max
  - gazebo::math::Box, 149
  - Math, 51
- maxVel
  - gazebo::physics::SurfaceParams, 752
- mean
  - Math, 51
- Merge
  - gazebo::math::Box, 147
- Mesh
  - gazebo::common::Mesh, 456
- mesh
  - gazebo::physics::Actor, 112
  - gazebo::physics::TrimeshShape, 792
- Mesh.hh, 979
- MeshCSG
  - gazebo::common::MeshCSG, 461
- MeshCSG.hh, 981
  - GPtrArray, 981
  - GtsSurface, 981
- MeshLoader
  - gazebo::common::MeshLoader, 463
- MeshLoader.hh, 982
- MeshManager.hh, 983
- MeshShapePtr
  - gazebo::physics, 93
- 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, 764
- MilToNano
  - gazebo::common::Time, 765
- min
  - gazebo::math::Box, 149
  - Math, 51
- minDepth
  - gazebo::physics::SurfaceParams, 752
- Model
  - gazebo::physics::Model, 472
- model
  - gazebo::physics::Joint, 385
- Model.hh, 984
- Model\_V
  - gazebo::physics, 93
- ModelDatabase.hh, 986
  - GZ\_MODEL\_DB\_MANIFEST\_FILENAME, 986
  - GZ\_MODEL\_MANIFEST\_FILENAME, 986
- modelPathsFromEnv
  - gazebo::common::SystemPaths, 757
- ModelPlugin
  - gazebo::ModelPlugin, 484
- ModelPluginPtr
  - gazebo, 81
- ModelPtr
  - gazebo::physics, 94
- ModelState
  - gazebo::physics::ModelState, 486
- ModelState.hh, 987
- modelTransform
  - gazebo::common::SkeletonNode, 716
- MouseEvent
  - gazebo::common::MouseEvent, 493
- MouseEvent.hh, 989
- MovableText
  - gazebo::rendering::MovableText, 497
- MovableText.hh, 990
- moveScale
  - gazebo::common::MouseEvent, 494
- MoveToPosition
  - gazebo::rendering::Camera, 174
  - gazebo::rendering::UserCamera, 800
  - gazebo::rendering::Visual, 862
- MoveToPositions

- gazebo::rendering::Camera, 174
- gazebo::rendering::Visual, 863
- MoveToVisual
  - gazebo::rendering::UserCamera, 801
- Moved
  - gazebo::rendering::WindowManager, 872
- MsgFactory.hh, 990
- MsgFactoryFn
  - gazebo::msgs, 89
- msgs.hh, 991
- mu1
  - gazebo::physics::SurfaceParams, 752
- mu2
  - gazebo::physics::SurfaceParams, 752
- MultiCameraSensor
  - gazebo::sensors::MultiCameraSensor, 503
- MultiCameraSensor.hh, 994
- MultiRayShape
  - gazebo::physics::MultiRayShape, 509
- MultiRayShape.hh, 994
- MultiRayShapePtr
  - gazebo::physics, 94
  
- NAN\_D
  - Math, 53
- NAN\_I
  - Math, 53
- NO\_BUTTON
  - gazebo::common::MouseEvent, 493
- NO\_EVENT
  - gazebo::common::MouseEvent, 493
- NODE
  - gazebo::common::SkeletonNode, 710
- NONE
  - gazebo::rendering::RenderEngine, 629
- NRealGen
  - gazebo::math, 87
- NSleep
  - gazebo::common::Time, 765
- NULL
  - CommonTypes.hh, 919
- name
  - gazebo::common::Animation, 125
  - gazebo::common::Material, 443
  - gazebo::common::NodeAnimation, 524
  - gazebo::common::SkeletonAnimation, 708
  - gazebo::common::SkeletonNode, 716
  - gazebo::physics::State, 732
  - gazebo::rendering::Camera, 181
  - sdf::Plugin, 569
- near
  - gazebo::sensors::GpuRaySensor, 331
- NewContact
  - gazebo::physics::ContactManager, 234
  
- newData
  - gazebo::rendering::Camera, 181
- newImageFrame
  - gazebo::rendering::Camera, 181
- newLaserScans
  - gazebo::physics::MultiRayShape, 513
- NewMsg
  - gazebo::msgs::MsgFactory, 501
- NewPhysicsEngine
  - gazebo::physics::PhysicsFactory, 558
- NewSensor
  - gazebo::sensors::SensorFactory, 681
- Node
  - gazebo::transport::Node, 516
- node
  - gazebo::physics::Entity, 285
  - gazebo::physics::PhysicsEngine, 556
  - gazebo::sensors::Sensor, 679
- Node.hh, 996
- NodeAnimation
  - gazebo::common::NodeAnimation, 521
- nodeIndex
  - gazebo::common::NodeAssignment, 525
- NodeMap
  - gazebo::common, 84
- NodeMapIter
  - gazebo::common, 84
- NodePtr
  - gazebo::transport, 102
- NodeTransform
  - gazebo::common::NodeTransform, 527
- nodes
  - gazebo::common::Skeleton, 703
- normal
  - gazebo::math::Plane, 565
- NormalRealDist
  - gazebo::math, 87
- Normalize
  - gazebo::math::Angle, 116
  - gazebo::math::Quaternion, 604
  - gazebo::math::Vector2d, 806
  - gazebo::math::Vector2i, 814
  - gazebo::math::Vector3, 827
  - gazebo::math::Vector4, 837
- normals
  - gazebo::physics::Contact, 231
- notifyRenderSingleObject
  - gazebo::rendering::GpuLaser, 319
- nsec
  - gazebo::common::Time, 779
- NullStream
  - Common, 32
- NumericAnimation
  - gazebo::common::NumericAnimation, 531



- NumericAnimationPtr
  - gazebo::common, 84
- NumericKeyFrame
  - gazebo::common::NumericKeyFrame, 532
- ORDER\_MAX
  - STLLoader.hh, 1061
- offset
  - gazebo::physics::MultiRayShape, 513
- Ogre, 103
- ogre, 103
- ogre\_gazebo.h, 997
- ogrePathsFromEnv
  - gazebo::common::SystemPaths, 757
- oldAction
  - gazebo::physics::Actor, 112
- onAnimationComplete
  - gazebo::rendering::Camera, 181
- OnPhysicsMsg
  - gazebo::physics::PhysicsEngine, 553
- OnPoseChange
  - gazebo::physics::Entity, 281
  - gazebo::physics::Link, 416
  - gazebo::physics::Model, 477
  - gazebo::rendering::Light, 400
- OnRequest
  - gazebo::physics::PhysicsEngine, 553
- Open
  - gazebo::common::LogPlay, 429
- operator<
  - gazebo::common::Time, 772, 773
  - gazebo::math::Angle, 118
- operator<<
  - gazebo::common::Color, 213
  - gazebo::common::Exception, 312
  - gazebo::common::Material, 442
  - gazebo::common::Time, 779
  - gazebo::common::Timer, 781
  - gazebo::math::Angle, 120
  - gazebo::math::Box, 148
  - gazebo::math::Matrix3, 447
  - gazebo::math::Matrix4, 454
  - gazebo::math::Pose, 579
  - gazebo::math::Quaternion, 610
  - gazebo::math::Vector2d, 810
  - gazebo::math::Vector2i, 819
  - gazebo::math::Vector3, 832
  - gazebo::math::Vector4, 842
  - gazebo::physics::CollisionState, 202
  - gazebo::physics::Inertial, 368
  - gazebo::physics::JointState, 391
  - gazebo::physics::LinkState, 426
  - gazebo::physics::ModelState, 491
  - gazebo::physics::WorldState, 892
  - sdf::ParamT, 546
- operator<=
  - gazebo::common::Time, 773, 774
  - gazebo::math::Angle, 119
- operator>
  - gazebo::common::Time, 776, 777
  - gazebo::math::Angle, 119
- operator>>
  - gazebo::common::Color, 213
  - gazebo::common::Time, 779
  - gazebo::math::Angle, 120
  - gazebo::math::Pose, 579
  - gazebo::math::Quaternion, 610
  - gazebo::math::Vector2d, 811
  - gazebo::math::Vector2i, 819
  - gazebo::math::Vector3, 833
  - gazebo::math::Vector4, 842
- operator>=
  - gazebo::common::Time, 777, 778
  - gazebo::math::Angle, 119
- operator\*
  - gazebo::common::Color, 207, 208
  - gazebo::common::NodeTransform, 528
  - gazebo::common::Time, 766, 767
  - gazebo::math::Angle, 116
  - gazebo::math::Matrix4, 451
  - gazebo::math::Pose, 577
  - gazebo::math::Quaternion, 605
  - gazebo::math::Vector2d, 806
  - gazebo::math::Vector2i, 815
  - gazebo::math::Vector3, 827, 832
  - gazebo::math::Vector4, 837, 838
  - sdf::ParamT, 545
- operator\*=  
  - gazebo::common::Color, 208
  - gazebo::common::Time, 767, 768
  - gazebo::math::Angle, 117
  - gazebo::math::Quaternion, 606
  - gazebo::math::Vector2d, 807
  - gazebo::math::Vector2i, 815
  - gazebo::math::Vector3, 828
  - gazebo::math::Vector4, 838
- operator()  
  - gazebo::common::NodeTransform, 527
  - gazebo::event::EventT, 304–306
- operator+
  - gazebo::common::Color, 208
  - gazebo::common::Time, 768
  - gazebo::math::Angle, 117
  - gazebo::math::Box, 147
  - gazebo::math::Pose, 577
  - gazebo::math::Quaternion, 606
  - gazebo::math::Vector2d, 807
  - gazebo::math::Vector2i, 816

- gazebo::math::Vector3, 828
- gazebo::math::Vector4, 838
- gazebo::physics::CollisionState, 201
- gazebo::physics::Inertial, 365
- gazebo::physics::JointState, 390
- gazebo::physics::LinkState, 425
- gazebo::physics::ModelState, 490
- gazebo::physics::WorldState, 891
- operator+=
  - gazebo::common::Color, 209
  - gazebo::common::Time, 769
  - gazebo::math::Angle, 117
  - gazebo::math::Box, 147
  - gazebo::math::Pose, 577
  - gazebo::math::Quaternion, 606
  - gazebo::math::Vector2d, 807
  - gazebo::math::Vector2i, 816
  - gazebo::math::Vector3, 828
  - gazebo::math::Vector4, 839
  - gazebo::physics::Inertial, 365
- operator-
  - gazebo::common::Color, 209
  - gazebo::common::Time, 769, 770
  - gazebo::math::Angle, 117
  - gazebo::math::Box, 148
  - gazebo::math::Pose, 577
  - gazebo::math::Quaternion, 606
  - gazebo::math::Vector2d, 808
  - gazebo::math::Vector2i, 816
  - gazebo::math::Vector3, 829
  - gazebo::math::Vector4, 839
  - gazebo::physics::CollisionState, 202
  - gazebo::physics::JointState, 390
  - gazebo::physics::LinkState, 426
  - gazebo::physics::ModelState, 490
  - gazebo::physics::State, 731
  - gazebo::physics::WorldState, 891
- operator==
  - gazebo::common::Color, 209
  - gazebo::common::Time, 770, 771
  - gazebo::math::Angle, 118
  - gazebo::math::Pose, 578
  - gazebo::math::Quaternion, 607
  - gazebo::math::Vector2d, 808
  - gazebo::math::Vector2i, 816
  - gazebo::math::Vector3, 829
  - gazebo::math::Vector4, 839
- operator/
  - gazebo::common::Color, 210
  - gazebo::common::Time, 771
  - gazebo::math::Angle, 118
  - gazebo::math::Vector2d, 808
  - gazebo::math::Vector2i, 817
  - gazebo::math::Vector3, 829, 830
  - gazebo::math::Vector4, 839, 840
- operator/=
  - gazebo::common::Color, 210
  - gazebo::common::Time, 772
  - gazebo::math::Angle, 118
  - gazebo::math::Vector2d, 809
  - gazebo::math::Vector2i, 817, 818
  - gazebo::math::Vector3, 830
  - gazebo::math::Vector4, 840
- operator=
  - gazebo::common::Color, 210
  - gazebo::common::PID, 560
  - gazebo::common::Time, 774, 775
  - gazebo::math::Box, 148
  - gazebo::math::Matrix4, 451, 452
  - gazebo::math::Plane, 564
  - gazebo::math::Quaternion, 607
  - gazebo::math::Vector2d, 809
  - gazebo::math::Vector2i, 818
  - gazebo::math::Vector3, 830, 831
  - gazebo::math::Vector4, 841
  - gazebo::physics::CollisionState, 202
  - gazebo::physics::Contact, 230, 231
  - gazebo::physics::Inertial, 365
  - gazebo::physics::JointState, 390
  - gazebo::physics::JointWrench, 393
  - gazebo::physics::LinkState, 426
  - gazebo::physics::ModelState, 491
  - gazebo::physics::State, 731
  - gazebo::physics::WorldState, 892
- operator==
  - gazebo::common::Color, 211
  - gazebo::common::Time, 775, 776
  - gazebo::math::Angle, 119
  - gazebo::math::Box, 148
  - gazebo::math::Matrix3, 445
  - gazebo::math::Matrix4, 452
  - gazebo::math::Pose, 578
  - gazebo::math::Quaternion, 607
  - gazebo::math::Vector2d, 810
  - gazebo::math::Vector2i, 819
  - gazebo::math::Vector3, 831
  - gazebo::math::Vector4, 841
  - gazebo::physics::Base, 141
- operator[]
  - gazebo::common::Color, 211
  - gazebo::math::Matrix3, 446
  - gazebo::math::Matrix4, 452
  - gazebo::math::Vector2d, 810
  - gazebo::math::Vector2i, 819
  - gazebo::math::Vector3, 831
  - gazebo::math::Vector4, 841
- OrbitViewController
  - gazebo::rendering::OrbitViewController, 535

- OrbitViewController.hh, 998
- PHONG
  - gazebo::common::Material, 437
- PID
  - gazebo::common::PID, 559
- PID.hh, 1009
- PIXEL\_FORMAT\_COUNT
  - gazebo::common::Image, 354
- PLANE\_SHAPE
  - gazebo::physics::Base, 137
- POINTS
  - gazebo::common::SubMesh, 736
- PRESS
  - gazebo::common::MouseEvent, 493
- Param
  - sdf::Param, 539
- Param.hh, 998
- Param\_V
  - gazebo::common, 84
  - sdf, 104
- ParamPtr
  - sdf, 104
- ParamT
  - sdf::ParamT, 544
- ParamT< T >, 543
- parent
  - gazebo::common::SkeletonNode, 717
  - gazebo::physics::Base, 144
  - gazebo::rendering::Visual, 868
- parentEntity
  - gazebo::physics::Entity, 285
- parentLink
  - gazebo::physics::Joint, 385
- parentName
  - gazebo::sensors::Sensor, 679
- ParseArgs
  - gazebo::Server, 687
- parseFloat
  - Math, 52
- parseInt
  - Math, 52
- parser.hh, 1000
- parser\_urdf.hh, 1001
- pathLength
  - gazebo::physics::Actor, 112
- pause
  - gazebo::event::Events, 300
- pause\_incoming
  - Transport, 76
- pause\_world
  - Classes for physics and dynamics, 45
- pause\_worlds
  - Classes for physics and dynamics, 46
- PauseIncoming
  - gazebo::transport::TopicManager, 786
- Physics.hh, 1002
- PhysicsEngine
  - gazebo::physics::PhysicsEngine, 549
- PhysicsEngine.hh, 1003
- PhysicsEnginePtr
  - gazebo::physics, 94
- PhysicsFactory.hh, 1005
- PhysicsFactoryFn
  - Classes for physics and dynamics, 44
- physicsSub
  - gazebo::physics::PhysicsEngine, 556
- PhysicsTypes.hh, 1006
  - GZ\_ALL\_COLLIDE, 1008
  - GZ\_FIXED\_COLLIDE, 1008
  - GZ\_GHOST\_COLLIDE, 1008
  - GZ\_NONE\_COLLIDE, 1008
  - GZ\_SENSOR\_COLLIDE, 1008
- physicsUpdateMutex
  - gazebo::physics::PhysicsEngine, 557
- pitchNode
  - gazebo::rendering::Camera, 181
- PixelFormat
  - gazebo::common::Image, 353
- PixelFormatNames
  - Common, 37
- PlaceOnEntity
  - gazebo::physics::Entity, 282
- PlaceOnNearestEntityBelow
  - gazebo::physics::Entity, 282
- placeable
  - gazebo::physics::Collision, 199
- Plane
  - gazebo::math::Plane, 563
- Plane.hh, 1010
- PlaneShape
  - gazebo::physics::PlaneShape, 566
- PlaneShape.hh, 1011
- Play
  - gazebo::physics::Actor, 111
- playStartTime
  - gazebo::physics::Actor, 113
- Plugin
  - sdf::Plugin, 569
- Plugin.hh, 1012, 1016
- pluginPathsFromEnv
  - gazebo::common::SystemPaths, 757
- PluginType
  - Common, 32
- plugins
  - gazebo::sensors::Sensor, 679
- pointSize
  - gazebo::common::Material, 443

- points
  - gazebo::math::RotationSpline, 645
  - gazebo::math::Spline, 728
- pos
  - gazebo::common::MouseEvent, 494
  - gazebo::math::Pose, 580
- Pose
  - gazebo::math::Pose, 574
- pose
  - gazebo::sensors::Sensor, 679
- Pose.hh, 1016
- PoseAnimation
  - gazebo::common::PoseAnimation, 581
- PoseAnimationPtr
  - gazebo::common, 84
- PoseKeyFrame
  - gazebo::common::PoseKeyFrame, 584
- poseMsg
  - gazebo::physics::Entity, 285
- poseSub
  - gazebo::sensors::Sensor, 679
- positions
  - gazebo::physics::Contact, 231
- PostRender
  - gazebo::rendering::Camera, 175
  - gazebo::rendering::DepthCamera, 249
  - gazebo::rendering::GpuLaser, 319
  - gazebo::rendering::UserCamera, 801
- postRender
  - gazebo::event::Events, 300
- PreRender
  - gazebo::rendering::Scene, 662
- preRender
  - gazebo::event::Events, 300
- precision
  - Math, 52
- PrepareHardwareBuffers
  - gazebo::rendering::DynamicRenderable, 265
- pressPos
  - gazebo::common::MouseEvent, 494
- prevAnimTime
  - gazebo::rendering::Camera, 181
- prevAnimationTime
  - gazebo::physics::Entity, 285
- prevFrameTime
  - gazebo::physics::Actor, 113
- prevPos
  - gazebo::common::MouseEvent, 494
- PrimitiveType
  - gazebo::common::SubMesh, 736
- Print
  - gazebo::common::Exception, 312
  - gazebo::physics::Base, 142
  - sdf::Plugin, 569
- print\_version
  - gazebo, 81
- PrintDescription
  - sdf::Element, 272
  - sdf::SDF, 668
- PrintDoc
  - sdf::SDF, 668
- PrintDocLeftPane
  - sdf::Element, 272
- PrintDocRightPane
  - sdf::Element, 272
- PrintEntityTree
  - gazebo::physics::World, 883
- PrintSceneGraph
  - gazebo::rendering::Scene, 662
- PrintSource
  - gazebo::common::NodeTransform, 528
- PrintTransforms
  - gazebo::common::Skeleton, 702
- PrintUsage
  - gazebo::Server, 687
- PrintValues
  - sdf::Element, 272
  - sdf::SDF, 668
- PrintWiki
  - sdf::Element, 272
  - sdf::SDF, 668
- ProcessIncoming
  - gazebo::transport::Node, 518
- ProcessMsg
  - gazebo::physics::BoxShape, 151
  - gazebo::physics::Collision, 197
  - gazebo::physics::CylinderShape, 245
  - gazebo::physics::HeightmapShape, 348
  - gazebo::physics::Inertial, 365
  - gazebo::physics::Link, 416
  - gazebo::physics::Model, 478
  - gazebo::physics::MultiRayShape, 513
  - gazebo::physics::PlaneShape, 567
  - gazebo::physics::RayShape, 625
  - gazebo::physics::Shape, 694
  - gazebo::physics::SphereShape, 723
  - gazebo::physics::SurfaceParams, 750
  - gazebo::physics::TrimeshShape, 791
- ProcessNodes
  - gazebo::transport::TopicManager, 786
- ProcessPublishers
  - gazebo::transport::Node, 518
- ProcessWriteQueue
  - gazebo::transport::Connection, 222
- Projector
  - gazebo::rendering::Projector, 586
- Projector.hh, 1017
- Publication

- gazebo::transport::Publication, 588
- Publication.hh, 1018
- PublicationPtr
  - gazebo::transport, 102
- PublicationTransport
  - gazebo::transport::PublicationTransport, 593
- PublicationTransport.hh, 1019
- PublicationTransportPtr
  - gazebo::transport, 102
- Publish
  - gazebo::transport::Publication, 591
  - gazebo::transport::Publisher, 596
  - gazebo::transport::TopicManager, 786
- PublishContacts
  - gazebo::physics::ContactManager, 234
- Publisher
  - gazebo::transport::Publisher, 595
- Publisher.hh, 1021
- PublisherPtr
  - gazebo::transport, 102
- Purple
  - gazebo::common::Color, 213
- Quaternion
  - gazebo::math::Quaternion, 600, 601
- Quaternion.hh, 1023
- r
  - gazebo::common::Color, 213
- R\_FLOAT16
  - gazebo::common::Image, 354
- R\_FLOAT32
  - gazebo::common::Image, 354
- RAY\_SHAPE
  - gazebo::physics::Base, 137
- RELEASE
  - gazebo::common::MouseEvent, 493
- RENDER\_PATH\_COUNT
  - gazebo::rendering::RenderEngine, 629
- RENDERING\_LINE\_LIST
  - gazebo::rendering, 97
- RENDERING\_LINE\_STRIP
  - gazebo::rendering, 97
- RENDERING\_MESH\_RESOURCE
  - gazebo::rendering, 97
- RENDERING\_POINT\_LIST
  - gazebo::rendering, 97
- RENDERING\_TRIANGLE\_FAN
  - gazebo::rendering, 97
- RENDERING\_TRIANGLE\_LIST
  - gazebo::rendering, 97
- RENDERING\_TRIANGLE\_STRIP
  - gazebo::rendering, 97
- REPLACE
  - gazebo::common::Material, 437
- RFIDSensor
  - gazebo::sensors::RFIDSensor, 633
- RFIDSensor.hh, 1032
- RFIDSensor\_V
  - gazebo::sensors, 100
- RFIDSensorPtr
  - gazebo::sensors, 100
- RFIDTag
  - gazebo::sensors::RFIDTag, 635
- RFIDTag.hh, 1032
- RFIDTag\_V
  - gazebo::sensors, 100
- RFIDTagPtr
  - gazebo::sensors, 100
- RFIDTagVisual
  - gazebo::rendering::RFIDTagVisual, 637
- RFIDTagVisual.hh, 1033
- RFIDTagVisualPtr
  - gazebo::rendering, 97
- RFIDVisual
  - gazebo::rendering::RFIDVisual, 639
- RFIDVisual.hh, 1034
- RFIDVisualPtr
  - gazebo::rendering, 97
- RGB\_FLOAT16
  - gazebo::common::Image, 354
- RGB\_FLOAT32
  - gazebo::common::Image, 354
- RGB\_INT16
  - gazebo::common::Image, 353
- RGB\_INT32
  - gazebo::common::Image, 353
- RGB\_INT8
  - gazebo::common::Image, 353
- RGBA
  - gazebo::common::Color, 205
- RGBA\_INT8
  - gazebo::common::Image, 353
- RIGHT
  - gazebo::common::MouseEvent, 493
- ROTATE
  - gazebo::common::NodeTransform, 526
- RTShaderSystem.hh, 1037
- Radian
  - gazebo::math::Angle, 120
- Rand.hh, 1024
- rangeCountRatio
  - gazebo::sensors::GpuRaySensor, 331
- rangeElem
  - gazebo::physics::MultiRayShape, 513
  - gazebo::sensors::GpuRaySensor, 331
- RawCallbackHelper
  - gazebo::transport::RawCallbackHelper, 614
- rawNW

- gazebo::common::Skeleton, 703
- RawNodeAnim
  - gazebo::common, 84
- RawNodeWeights
  - gazebo::common, 84
- RawSkeletonAnim
  - gazebo::common, 84
- rawTransforms
  - gazebo::common::SkeletonNode, 717
- rayCountRatio
  - gazebo::sensors::GpuRaySensor, 331
- rayElem
  - gazebo::physics::MultiRayShape, 513
- RaySensor
  - gazebo::sensors::RaySensor, 616
- RaySensor.hh, 1025
- RaySensor\_V
  - gazebo::sensors, 100
- RaySensorPtr
  - gazebo::sensors, 100
- RayShape
  - gazebo::physics::RayShape, 623, 624
- RayShape.hh, 1026
- RayShapePtr
  - gazebo::physics, 94
- rays
  - gazebo::physics::MultiRayShape, 514
- Read
  - gazebo::transport::Connection, 222
- ReadCallback
  - gazebo::transport::Connection, 219
- readDoc
  - sdf, 104, 105
- readFile
  - sdf, 105
- readString
  - sdf, 105
- readXml
  - sdf, 105
- realTime
  - gazebo::physics::State, 732
- RecalcTangents
  - gazebo::math::RotationSpline, 644
  - gazebo::math::Spline, 726
- RecalculateMatrix
  - gazebo::common::NodeTransform, 528
- RecalculateNormals
  - gazebo::common::Mesh, 459
  - gazebo::common::SubMesh, 741
- Red
  - gazebo::common::Color, 214
- RegisterAll
  - gazebo::physics::PhysicsFactory, 558
  - gazebo::sensors::SensorFactory, 681
- RegisterMsg
  - gazebo::msgs::MsgFactory, 501
- RegisterPhysicsEngine
  - gazebo::physics::PhysicsFactory, 558
- RegisterSensor
  - gazebo::sensors::SensorFactory, 681
- RegisterTopicNamespace
  - gazebo::transport::ConnectionManager, 226
  - gazebo::transport::TopicManager, 787
- relativeEndPos
  - gazebo::physics::RayShape, 627
- relativeStartPos
  - gazebo::physics::RayShape, 627
- Remove
  - gazebo::common::LogRecord, 432
- remove\_scene
  - Rendering, 66
- remove\_sensor
  - Sensors, 72
- remove\_sensors
  - Sensors, 72
- remove\_worlds
  - Classes for physics and dynamics, 46
- RemoveCallback
  - gazebo::transport::Node, 518
- RemoveChild
  - gazebo::physics::Base, 142
  - gazebo::physics::Model, 478
  - sdf::Element, 272
- RemoveChildJoint
  - gazebo::physics::Link, 416
- RemoveChildren
  - gazebo::physics::Base, 142
- RemoveConnection
  - gazebo::transport::ConnectionManager, 226
- RemoveFromParent
  - sdf::Element, 272
- RemoveNode
  - gazebo::transport::TopicManager, 787
- RemoveParentJoint
  - gazebo::physics::Link, 416
- RemovePlugin
  - gazebo::physics::World, 884
  - gazebo::rendering::Visual, 863
- RemoveScene
  - gazebo::rendering::RenderEngine, 631
  - gazebo::rendering::RTShaderSystem, 649
- removeScene
  - gazebo::rendering::Events, 289
- RemoveSensor
  - gazebo::sensors::SensorManager, 684
- RemoveSensors
  - gazebo::sensors::SensorManager, 684
- RemoveShadows

- gazebo::rendering::RTShaderSystem, 649
- RemoveSubscription
  - gazebo::transport::Publication, 591
- RemoveTransport
  - gazebo::transport::Publication, 591
- RemoveVisual
  - gazebo::rendering::Scene, 662
- Render
  - gazebo::rendering::Camera, 175
- render
  - gazebo::event::Events, 301
- RenderEngine.hh, 1027
- RenderEvents.hh, 1028
- RenderImpl
  - gazebo::rendering::Camera, 175
- RenderOpType
  - gazebo::rendering, 97
- RenderPathType
  - gazebo::rendering::RenderEngine, 629
- renderTarget
  - gazebo::rendering::Camera, 182
- renderTexture
  - gazebo::rendering::Camera, 182
- RenderTypes.hh, 1030
  - GZ\_VISIBILITY\_ALL, 1031
  - GZ\_VISIBILITY\_GUI, 1031
  - GZ\_VISIBILITY\_NOT\_SELECTABLE, 1031
  - GZ\_VISIBILITY\_SELECTION, 1031
- Rendering, 64
  - create\_scene, 66
  - fini, 66
  - get\_scene, 66
  - init, 66
  - load, 66
  - remove\_scene, 66
- Rendering.hh, 1029
- request
  - Transport, 76
- requestNoReply
  - Transport, 77
- requestPub
  - gazebo::physics::Entity, 285
- requestSub
  - gazebo::physics::PhysicsEngine, 557
- requests
  - gazebo::rendering::Camera, 182
- required
  - sdf::Param, 542
- Rescale
  - gazebo::common::Image, 356
- Reset
  - gazebo::common::Color, 211
  - gazebo::common::PID, 560
  - gazebo::common::SkeletonNode, 714
  - gazebo::math::Pose, 578
  - gazebo::ModelPlugin, 484
  - gazebo::physics::Base, 142
  - gazebo::physics::Contact, 231
  - gazebo::physics::Entity, 282
  - gazebo::physics::Inertial, 365
  - gazebo::physics::Joint, 381
  - gazebo::physics::JointController, 386
  - gazebo::physics::Link, 417
  - gazebo::physics::Model, 478
  - gazebo::physics::PhysicsEngine, 554
  - gazebo::physics::World, 884
  - gazebo::SensorPlugin, 686
  - gazebo::SystemPlugin, 759
  - gazebo::VisualPlugin, 870
  - gazebo::WorldPlugin, 887
  - sdf::Element, 273
  - sdf::Param, 541
  - sdf::ParamT, 545
- ResetCount
  - gazebo::physics::ContactManager, 234
- ResetEntities
  - gazebo::physics::World, 884
- ResetTime
  - gazebo::physics::World, 884
- Resize
  - gazebo::rendering::GUIOverlay, 341
  - gazebo::rendering::UserCamera, 801
  - gazebo::rendering::WindowManager, 873
- responsePub
  - gazebo::physics::PhysicsEngine, 557
- Road, 639
  - gazebo::physics::Road, 640
- Road.hh, 1034
- Road2d
  - gazebo::rendering::Road2d, 641
- Road2d.hh, 1036
- RoadPtr
  - gazebo::physics, 94
- root
  - gazebo::common::Skeleton, 703
  - gazebo::rendering::RenderEngine, 631
  - sdf::SDF, 669
- rot
  - gazebo::math::Pose, 580
- Rotate
  - gazebo::physics::Inertial, 366
- rotate
  - gazebo::common::PoseKeyFrame, 585
- RotatePitch
  - gazebo::rendering::Camera, 175
- RotatePositionAboutOrigin
  - gazebo::math::Pose, 578
- RotateVector

- gazebo::math::Quaternion, 607
- RotateVectorReverse
  - gazebo::math::Quaternion, 608
- RotateYaw
  - gazebo::rendering::Camera, 175
- RotationSpline
  - gazebo::math::RotationSpline, 643
- RotationSpline.hh, 1036
- Round
  - gazebo::math::Pose, 579
  - gazebo::math::Quaternion, 608
  - gazebo::math::Vector3, 831
- Run
  - gazebo::Master, 434
  - gazebo::physics::World, 884
  - gazebo::sensors::SensorManager, 684
  - gazebo::Server, 687
  - gazebo::transport::ConnectionManager, 226
- run
  - gazebo, 81
  - Sensors, 72
  - Transport, 77
- run\_once
  - Sensors, 72
- run\_world
  - Classes for physics and dynamics, 46
- run\_worlds
  - Classes for physics and dynamics, 46
- RunOnce
  - gazebo::Master, 434
- RunThread
  - gazebo::Master, 434
- RunUpdate
  - gazebo::transport::ConnectionManager, 227
- SCALE
  - gazebo::common::NodeTransform, 526
- SCREW\_JOINT
  - gazebo::physics::Base, 137
- SCROLL
  - gazebo::common::MouseEvent, 493
- SDF
  - sdf::SDF, 668
- SDF.hh, 1041
  - SDF\_VERSION, 1042
- SDF\_VERSION
  - SDF.hh, 1042
- SDFPtr
  - sdf, 104
- SENSOR\_PLUGIN
  - Common, 32
- SHADE\_COUNT
  - gazebo::common::Material, 437
- SHAPE
  - gazebo::physics::Base, 137
- SLIDER\_JOINT
  - gazebo::physics::Base, 137
- SM2Profile
  - gazebo::rendering::GzTerrainMatGen::SM2Profile, 721
- SPHERE\_SHAPE
  - gazebo::physics::Base, 137
- SSLM\_NormalMapLightingObjectSpace
  - gazebo::rendering::RTShaderSystem, 647
- SSLM\_NormalMapLightingTangentSpace
  - gazebo::rendering::RTShaderSystem, 647
- SSLM\_PerPixelLighting
  - gazebo::rendering::RTShaderSystem, 647
- SSLM\_PerVertexLighting
  - gazebo::rendering::RTShaderSystem, 647
- STLLoader
  - gazebo::common::STLLoader, 733
- STLLoader.hh, 1059
  - COR3\_MAX, 1060
  - FACE\_MAX, 1060
  - LINE\_MAX\_LEN, 1061
  - ORDER\_MAX, 1061
- STOP\_CFM
  - gazebo::physics::Joint, 374
- STOP\_ERP
  - gazebo::physics::Joint, 374
- SUSPENSION\_CFM
  - gazebo::physics::Joint, 374
- SUSPENSION\_ERP
  - gazebo::physics::Joint, 374
- SYSTEM\_PLUGIN
  - Common, 32
- Save
  - gazebo::physics::World, 884
- saveCount
  - gazebo::rendering::Camera, 182
- SaveFrame
  - gazebo::rendering::Camera, 175, 176
  - gazebo::sensors::CameraSensor, 186
  - gazebo::sensors::DepthCameraSensor, 253
  - gazebo::sensors::MultiCameraSensor, 505
- saveFrameBuffer
  - gazebo::rendering::Camera, 182
- SavePNG
  - gazebo::common::Image, 357
- Scale
  - gazebo::common::Mesh, 460
  - gazebo::common::NodeAnimation, 523
  - gazebo::common::Skeleton, 702
  - gazebo::common::SkeletonAnimation, 707
  - gazebo::common::SubMesh, 741
  - gazebo::math::Quaternion, 608
- scale



- gazebo::physics::HeightmapShape, 348
- ScaleXAxis
  - gazebo::rendering::AxisVisual, 130
- ScaleYAxis
  - gazebo::rendering::AxisVisual, 130
- ScaleZAxis
  - gazebo::rendering::AxisVisual, 131
- scanElem
  - gazebo::physics::MultiRayShape, 514
  - gazebo::sensors::GpuRaySensor, 332
- Scene
  - gazebo::rendering::Scene, 653
- scene
  - gazebo::rendering::Camera, 182
  - gazebo::rendering::Visual, 868
- Scene.hh, 1038
- SceneFromSDF
  - Messages, 60
- sceneNode
  - gazebo::rendering::Camera, 182
  - gazebo::rendering::Visual, 869
- ScenePtr
  - gazebo::rendering, 97
- ScrewJoint
  - gazebo::physics::ScrewJoint, 666
- ScrewJoint.hh, 1039
- scriptLength
  - gazebo::physics::Actor, 113
- scroll
  - gazebo::common::MouseEvent, 494
- sdf, 103
  - addNestedModel, 104
  - copyChildren, 104
  - ElementPtr, 104
  - ElementPtr\_V, 104
  - gazebo::physics::Base, 144
  - gazebo::physics::PhysicsEngine, 557
  - gazebo::rendering::Camera, 182
  - gazebo::sensors::Sensor, 679
  - init, 104
  - initDoc, 104
  - initFile, 104
  - initString, 104
  - initXml, 104
  - Param\_V, 104
  - ParamPtr, 104
  - readDoc, 104, 105
  - readFile, 105
  - readString, 105
  - readXml, 105
  - SDFPtr, 104
- sdf.hh, 1040
- sdf::Converter, 242
  - Convert, 242
- sdf::Element, 266
  - ~Element, 269
  - AddAttribute, 269
  - AddElement, 269
  - AddElementDescription, 269
  - AddValue, 269
  - ClearElements, 269
  - Clone, 269
  - Copy, 270
  - Element, 269
  - GetAttribute, 270
  - GetAttributeCount, 270
  - GetAttributeSet, 270
  - GetCopyChildren, 270
  - GetDescription, 270
  - GetElement, 270
  - GetElementDescription, 270
  - GetElementDescriptionCount, 270
  - GetElementImpl, 271
  - GetFirstElement, 271
  - GetInclude, 271
  - GetName, 271
  - GetNextElement, 271
  - GetParent, 271
  - GetRequired, 271
  - GetValue, 271
  - GetValueBool, 271
  - GetValueChar, 271
  - GetValueColor, 271
  - GetValueDouble, 271
  - GetValueFloat, 271
  - GetValueInt, 271
  - GetValuePose, 271
  - GetValueQuaternion, 271
  - GetValueString, 271
  - GetValueTime, 271
  - GetValueUInt, 271
  - GetValueVector2d, 271
  - GetValueVector3, 271
  - HasAttribute, 271
  - HasElement, 272
  - HasElementDescription, 272
  - InsertElement, 272
  - PrintDescription, 272
  - PrintDocLeftPane, 272
  - PrintDocRightPane, 272
  - PrintValues, 272
  - PrintWiki, 272
  - RemoveChild, 272
  - RemoveFromParent, 272
  - Reset, 273
  - Set, 273
  - SetCopyChildren, 273
  - SetDescription, 273

- SetInclude, 273
- SetName, 273
- SetParent, 273
- SetRequired, 273
- ToString, 273
- Update, 274
- sdf::Param, 537
  - ~Param, 539
  - Clone, 539
  - description, 542
  - Get, 539, 540
  - GetAsString, 540
  - GetDefaultAsString, 540
  - GetDescription, 540
  - GetKey, 540
  - GetRequired, 540
  - GetSet, 540
  - GetTypeNames, 540
  - IsBool, 540
  - IsChar, 540
  - IsColor, 540
  - IsDouble, 541
  - IsFloat, 541
  - IsInt, 541
  - IsPose, 541
  - IsQuaternion, 541
  - IsStr, 541
  - IsTime, 541
  - IsUInt, 541
  - IsVector2d, 541
  - IsVector2i, 541
  - IsVector3, 541
  - key, 542
  - Param, 539
  - required, 542
  - Reset, 541
  - Set, 541, 542
  - set, 542
  - SetDescription, 542
  - SetFromString, 542
  - SetUpdateFunc, 542
  - typeName, 542
  - Update, 542
  - updateFunc, 543
- sdf::ParamT
  - ~ParamT, 544
  - Clone, 545
  - defaultValue, 546
  - GetAsString, 545
  - GetDefaultAsString, 545
  - GetDefaultValue, 545
  - GetValue, 545
  - operator<<, 546
  - operator\*, 545
  - ParamT, 544
  - Reset, 545
  - Set, 545
  - SetFromString, 545
  - SetValue, 545
  - Update, 546
  - value, 546
- sdf::ParamT< T >, 543
- sdf::Plugin, 568
  - Clear, 569
  - data, 569
  - filename, 569
  - name, 569
  - Plugin, 569
  - Print, 569
- sdf::SDF, 668
  - PrintDescription, 668
  - PrintDoc, 668
  - PrintValues, 668
  - PrintWiki, 668
  - root, 669
  - SDF, 668
  - SetFromString, 668
  - ToString, 668
  - version, 669
  - Write, 669
- sec
  - gazebo::common::Time, 779
- SecToNano
  - gazebo::common::Time, 778
- SelectVisual
  - gazebo::rendering::Scene, 662
- SelectionObj
  - gazebo::rendering::SelectionObj, 670
- SelectionObj.hh, 1042
- SendMessage
  - gazebo::transport::Publisher, 597
- Sensor
  - gazebo::sensors::Sensor, 674
- Sensor.hh, 1043
- Sensor\_V
  - gazebo::sensors, 100
- SensorFactor, 680
- SensorFactory.hh, 1044
- SensorFactoryFn
  - gazebo::sensors, 100
- SensorManager.hh, 1045
- SensorPlugin
  - gazebo::SensorPlugin, 686
- SensorPluginPtr
  - gazebo, 81
- SensorPtr
  - gazebo::sensors, 100
- SensorTypes.hh, 1047

- 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
  - stop, 72
- Sensors.hh, 1046
- SensorsInitialized
  - gazebo::sensors::SensorManager, 684
- Server
  - gazebo::Server, 687
- Server.hh, 1049
- Set
  - gazebo::common::Color, 211
  - gazebo::common::NodeTransform, 528
  - gazebo::common::Time, 778, 779
  - gazebo::math::Matrix4, 453
  - gazebo::math::Plane, 564
  - gazebo::math::Pose, 579
  - gazebo::math::Quaternion, 608
  - gazebo::math::Vector2d, 810
  - gazebo::math::Vector2i, 819
  - gazebo::math::Vector3, 831
  - gazebo::math::Vector4, 841
  - Messages, 61, 62
  - sdf::Element, 273
  - sdf::Param, 541, 542
  - sdf::ParamT, 545
- set
  - sdf::Param, 542
- SetActive
  - gazebo::rendering::SelectionObj, 670
  - gazebo::sensors::DepthCameraSensor, 253
  - gazebo::sensors::Sensor, 678
- SetAltitude
  - gazebo::physics::PlaneShape, 567
- SetAmbient
  - gazebo::common::Material, 440
  - gazebo::rendering::Visual, 863
- SetAmbientColor
  - gazebo::rendering::Scene, 662
- SetAnchor
  - gazebo::physics::Joint, 381
  - gazebo::physics::ScrewJoint, 667
  - gazebo::physics::SliderJoint, 719
- SetAngle
  - gazebo::physics::Joint, 381
- SetAngleMax
  - gazebo::sensors::GpuRaySensor, 329
- SetAngleMin
  - gazebo::sensors::GpuRaySensor, 329
- SetAngularAccel
  - gazebo::physics::Link, 417
  - gazebo::physics::Model, 478
- SetAngularDamping
  - gazebo::physics::Link, 417
- SetAngularVel
  - gazebo::physics::Link, 417
  - gazebo::physics::Model, 478
- SetAnimation
  - gazebo::physics::Entity, 282
- SetAspectRatio
  - gazebo::rendering::Camera, 176
- SetAttenuation
  - gazebo::rendering::Light, 400
- SetAttribute
  - gazebo::physics::Joint, 382
- SetAutoCalculate
  - gazebo::math::RotationSpline, 645
  - gazebo::math::Spline, 727
- SetAutoDisable
  - gazebo::physics::Link, 417
  - gazebo::physics::Model, 478
- SetAutoDisableFlag
  - gazebo::physics::PhysicsEngine, 554
- SetAxis
  - gazebo::physics::BallJoint, 133
  - gazebo::physics::Joint, 382
- SetAxisMaterial
  - gazebo::rendering::AxisVisual, 131
- SetBackgroundColor
  - gazebo::rendering::Scene, 662
- SetBaseline
  - gazebo::rendering::MovableText, 499
- SetBindShapeTransform
  - gazebo::common::Skeleton, 702
- SetBlendFactors
  - gazebo::common::Material, 440
- SetBlendMode
  - gazebo::common::Material, 440
- SetCallbackId
  - gazebo::transport::Subscriber, 746
- SetCamera
  - gazebo::rendering::WindowManager, 873
- SetCanonicalLink
  - gazebo::physics::Entity, 282
- SetCaptureData
  - gazebo::rendering::Camera, 176
- SetCastShadows
  - gazebo::rendering::Light, 401
  - gazebo::rendering::Visual, 863
- SetCategoryBits

- gazebo::physics::Collision, 197
- SetCellCount
  - gazebo::rendering::Grid, 335
- SetCellLength
  - gazebo::rendering::Grid, 335
- SetCharHeight
  - gazebo::rendering::MovableText, 499
- SetClipDist
  - gazebo::rendering::Camera, 176
- SetCmd
  - gazebo::common::PID, 561
- SetCmdMax
  - gazebo::common::PID, 561
- SetCmdMin
  - gazebo::common::PID, 561
- SetCoG
  - gazebo::physics::Inertial, 366
- SetCol
  - gazebo::math::Matrix3, 446
- SetCollideBits
  - gazebo::physics::Collision, 197
- SetCollideMode
  - gazebo::physics::Link, 417
  - gazebo::physics::Model, 478
- SetCollision
  - gazebo::physics::Collision, 197
- SetColor
  - gazebo::rendering::Grid, 335
  - gazebo::rendering::MovableText, 499
- SetComponent
  - gazebo::common::NodeTransform, 528
- SetContactMaxCorrectingVel
  - gazebo::physics::PhysicsEngine, 554
- SetContactSurfaceLayer
  - gazebo::physics::PhysicsEngine, 554
- SetContactsEnabled
  - gazebo::physics::Collision, 198
- SetCopyChildren
  - sdf::Element, 273
- SetDGain
  - gazebo::common::PID, 561
- SetDamping
  - gazebo::physics::Joint, 382
- SetDepthTarget
  - gazebo::rendering::DepthCamera, 249
- SetDepthWrite
  - gazebo::common::Material, 441
- SetDescription
  - sdf::Element, 273
  - sdf::Param, 542
- SetDiffuse
  - gazebo::common::Material, 441
  - gazebo::rendering::Visual, 863
- SetDiffuseColor
  - gazebo::rendering::Light, 401
- SetDirection
  - gazebo::rendering::Light, 401
- SetDistance
  - gazebo::rendering::OrbitViewController, 536
- SetEmissive
  - gazebo::common::Material, 441
  - gazebo::rendering::LaserVisual, 397
  - gazebo::rendering::Visual, 864
- SetEnabled
  - gazebo::common::DiagnosticManager, 256
  - gazebo::physics::Link, 418
  - gazebo::physics::Model, 479
  - gazebo::rendering::ContactVisual, 240
  - gazebo::rendering::Projector, 587
  - gazebo::rendering::ViewController, 848
- SetFiducial
  - gazebo::physics::RayShape, 625
- SetFilename
  - gazebo::physics::TrimeshShape, 791
- SetFocalPoint
  - gazebo::rendering::OrbitViewController, 536
  - gazebo::rendering::UserCamera, 801
- SetFog
  - gazebo::rendering::Scene, 662
- SetFontName
  - gazebo::rendering::MovableText, 499
- SetForce
  - gazebo::physics::Joint, 382
  - gazebo::physics::Link, 418
- SetFromABGR
  - gazebo::common::Color, 211
- SetFromARGB
  - gazebo::common::Color, 212
- SetFromAxes
  - gazebo::math::Matrix3, 446
- SetFromAxis
  - gazebo::math::Matrix3, 446
  - gazebo::math::Quaternion, 608, 609
- SetFromBGRA
  - gazebo::common::Color, 212
- SetFromData
  - gazebo::common::Image, 357
- SetFromDegree
  - gazebo::math::Angle, 120
- SetFromEuler
  - gazebo::math::Quaternion, 609
- SetFromHSV
  - gazebo::common::Color, 212
- SetFromRGBA
  - gazebo::common::Color, 212
- SetFromRadian
  - gazebo::math::Angle, 120
- SetFromString

- sdf::Param, 542
- sdf::ParamT, 545
- sdf::SDF, 668
- SetFromYUV
  - gazebo::common::Color, 212
- SetGravity
  - gazebo::physics::PhysicsEngine, 554
- SetGravityMode
  - gazebo::physics::Link, 418
  - gazebo::physics::Model, 479
- SetGrid
  - gazebo::rendering::Scene, 663
- SetHFOV
  - gazebo::rendering::Camera, 177
- SetHandle
  - gazebo::common::SkeletonNode, 714
- SetHeight
  - gazebo::rendering::Grid, 335
- SetHighStop
  - gazebo::physics::BallJoint, 133
  - gazebo::physics::Joint, 383
- SetHighlight
  - gazebo::rendering::SelectionObj, 671
- SetHighlighted
  - gazebo::rendering::Visual, 864
- SetIGain
  - gazebo::common::PID, 561
- SetIMax
  - gazebo::common::PID, 561
- SetIMin
  - gazebo::common::PID, 562
- SetIXX
  - gazebo::physics::Inertial, 366
- SetIXY
  - gazebo::physics::Inertial, 367
- SetIXZ
  - gazebo::physics::Inertial, 367
- SetIYY
  - gazebo::physics::Inertial, 367
- SetIYZ
  - gazebo::physics::Inertial, 367
- SetIZZ
  - gazebo::physics::Inertial, 367
- SetId
  - gazebo::common::SkeletonNode, 714
- SetImageHeight
  - gazebo::rendering::Camera, 177
- SetImageSize
  - gazebo::rendering::Camera, 177
- SetImageWidth
  - gazebo::rendering::Camera, 177
- SetInclude
  - sdf::Element, 273
- SetIndexCount
  - gazebo::common::SubMesh, 741
- SetInertiaMatrix
  - gazebo::physics::Inertial, 366
- SetInertial
  - gazebo::physics::Link, 418
- SetInitialRelativePose
  - gazebo::physics::Entity, 283
- SetInitialTransform
  - gazebo::common::SkeletonNode, 715
- SetInverseBindTransform
  - gazebo::common::SkeletonNode, 715
- SetJointAnimation
  - gazebo::physics::Model, 479
- SetJointPosition
  - gazebo::physics::JointController, 386
  - gazebo::physics::Model, 479
- SetJointPositions
  - gazebo::physics::JointController, 386
  - gazebo::physics::Model, 479
- SetKinematic
  - gazebo::physics::Link, 418
- SetLaserRetro
  - gazebo::physics::Collision, 198
  - gazebo::physics::Link, 418
  - gazebo::physics::Model, 480
- SetLength
  - gazebo::common::Animation, 124
  - gazebo::physics::CylinderShape, 245
  - gazebo::physics::RayShape, 626
- SetLightType
  - gazebo::rendering::Light, 401
- SetLighting
  - gazebo::common::Material, 441
- SetLineWidth
  - gazebo::rendering::Grid, 335
- SetLinearAccel
  - gazebo::physics::Link, 419
  - gazebo::physics::Model, 480
- SetLinearDamping
  - gazebo::physics::Link, 419
- SetLinearVel
  - gazebo::physics::Link, 419
  - gazebo::physics::Model, 480
- SetLinkWorldPose
  - gazebo::physics::Model, 480
- SetLocallyAdvertised
  - gazebo::transport::Publication, 591
- SetLowStop
  - gazebo::physics::BallJoint, 133
  - gazebo::physics::Joint, 383
- SetMass
  - gazebo::physics::Inertial, 367
- SetMaterial
  - gazebo::rendering::Visual, 864

- SetMaterialIndex
  - gazebo::common::SubMesh, 741
- SetMaxContacts
  - gazebo::physics::PhysicsEngine, 554
- SetMaxForce
  - gazebo::physics::Joint, 383
- SetModel
  - gazebo::physics::Joint, 383
- SetModelTransform
  - gazebo::common::SkeletonNode, 715
- SetName
  - gazebo::common::Mesh, 460
  - gazebo::common::NodeAnimation, 523
  - gazebo::common::SkeletonAnimation, 707
  - gazebo::common::SkeletonNode, 715
  - gazebo::physics::Base, 142
  - gazebo::physics::Entity, 283
  - gazebo::physics::State, 732
  - gazebo::rendering::Camera, 177
  - gazebo::rendering::Light, 401
  - gazebo::rendering::Visual, 864
  - sdf::Element, 273
- SetNormal
  - gazebo::common::SubMesh, 742
  - gazebo::physics::PlaneShape, 568
- SetNormalCount
  - gazebo::common::SubMesh, 742
- SetNormalMap
  - gazebo::rendering::Visual, 864
- SetNumVertAttached
  - gazebo::common::Skeleton, 703
- SetOperationType
  - gazebo::rendering::DynamicRenderable, 266
- SetPGain
  - gazebo::common::PID, 562
- SetParams
  - gazebo::Server, 687
- SetParent
  - gazebo::common::SkeletonNode, 715
  - gazebo::physics::Base, 143
  - gazebo::sensors::CameraSensor, 186
  - gazebo::sensors::DepthCameraSensor, 253
  - gazebo::sensors::Sensor, 678
  - sdf::Element, 273
- SetParentSensor
  - gazebo::rendering::GpuLaser, 319
- SetPath
  - gazebo::common::Mesh, 460
- SetPaused
  - gazebo::physics::World, 884
- SetPerPixelLighting
  - gazebo::rendering::RTShaderSystem, 649
- SetPoint
  - gazebo::rendering::DynamicLines, 262
- SetPointSize
  - gazebo::common::Material, 441
- SetPoints
  - gazebo::physics::RayShape, 626
- SetPose
  - gazebo::rendering::Visual, 864
- SetPosition
  - gazebo::rendering::Light, 401
  - gazebo::rendering::Visual, 865
- SetPrimitiveType
  - gazebo::common::SubMesh, 742
- SetPublication
  - gazebo::transport::Publisher, 597
- SetQuiet
  - Common, 36
- SetRadius
  - gazebo::physics::CylinderShape, 245
  - gazebo::physics::SphereShape, 723
- SetRange
  - gazebo::rendering::Light, 402
- SetRangeCount
  - gazebo::rendering::GpuLaser, 320
- SetRelativePose
  - gazebo::physics::Entity, 283
- SetRenderRate
  - gazebo::rendering::Camera, 177
- SetRenderTarget
  - gazebo::rendering::Camera, 178
  - gazebo::rendering::UserCamera, 801
- SetRequired
  - sdf::Element, 273
- SetRetro
  - gazebo::physics::RayShape, 626
- SetRibbonTrail
  - gazebo::rendering::Visual, 865
- SetRootNode
  - gazebo::common::Skeleton, 703
- SetRotation
  - gazebo::common::PoseKeyFrame, 584
  - gazebo::rendering::Visual, 865
- SetSID
  - gazebo::common::NodeTransform, 529
- SetSORPGSIters
  - gazebo::physics::PhysicsEngine, 555
- SetSORPGSPreconIters
  - gazebo::physics::PhysicsEngine, 555
- SetSORPGSW
  - gazebo::physics::PhysicsEngine, 555
- SetSaveFramePathname
  - gazebo::rendering::Camera, 178
- SetSaveable
  - gazebo::physics::Base, 143
- SetScale
  - gazebo::common::Mesh, 460

- gazebo::common::SubMesh, 742
- gazebo::math::Matrix4, 453
- gazebo::physics::TrimeshShape, 791
- gazebo::rendering::Visual, 865
- SetScene
  - gazebo::rendering::Camera, 178
  - gazebo::rendering::Visual, 865
- SetSceneNode
  - gazebo::rendering::Camera, 178
- SetSeed
  - gazebo::math::Rand, 612
  - gazebo::physics::PhysicsEngine, 555
- SetSelected
  - gazebo::physics::Base, 143
  - gazebo::physics::Link, 419
  - gazebo::rendering::Light, 402
- setSelectedEntity
  - gazebo::event::Events, 301
- SetSelfCollide
  - gazebo::physics::Link, 419
- SetShadeMode
  - gazebo::common::Material, 441
- SetShaderType
  - gazebo::rendering::Visual, 866
- SetShadowsEnabled
  - gazebo::rendering::Scene, 663
- SetShape
  - gazebo::physics::Collision, 198
- SetShininess
  - gazebo::common::Material, 441
- SetShowOnTop
  - gazebo::rendering::MovableText, 499
- SetSimTime
  - gazebo::physics::World, 885
- SetSize
  - gazebo::physics::BoxShape, 151
  - gazebo::physics::CylinderShape, 245
  - gazebo::physics::PlaneShape, 568
- SetSkeleton
  - gazebo::common::Mesh, 460
- SetSkeletonPose
  - gazebo::rendering::Visual, 866
- SetSourceValues
  - gazebo::common::NodeTransform, 529
- SetSpaceWidth
  - gazebo::rendering::MovableText, 500
- SetSpecular
  - gazebo::common::Material, 442
  - gazebo::rendering::Visual, 866
- SetSpecularColor
  - gazebo::rendering::Light, 402
- SetSpotFalloff
  - gazebo::rendering::Light, 402
- SetSpotInnerAngle
  - gazebo::rendering::Light, 402
- SetSpotOuterAngle
  - gazebo::rendering::Light, 402
- SetState
  - gazebo::physics::Collision, 198
  - gazebo::physics::Joint, 383
  - gazebo::physics::Link, 419
  - gazebo::physics::Model, 481
  - gazebo::physics::World, 885
- SetStatic
  - gazebo::physics::Entity, 283
- SetStepTime
  - gazebo::physics::PhysicsEngine, 555
- SetSubMeshCenter
  - gazebo::common::SubMesh, 742
- SetTension
  - gazebo::math::Spline, 727
- SetTexCoord
  - gazebo::common::SubMesh, 742
- SetTexCoordCount
  - gazebo::common::SubMesh, 743
- SetText
  - gazebo::rendering::MovableText, 500
- SetTextAlignment
  - gazebo::rendering::MovableText, 500
- SetTexture
  - gazebo::rendering::Projector, 587
- SetTextureImage
  - gazebo::common::Material, 442
- SetThreadPitch
  - gazebo::physics::ScrewJoint, 667
- SetTime
  - gazebo::common::Animation, 124
- SetToldentity
  - gazebo::math::Quaternion, 609
- SetToMax
  - gazebo::math::Vector3, 832
- SetToMin
  - gazebo::math::Vector3, 832
- SetToWallTime
  - gazebo::common::Time, 779
- SetTorque
  - gazebo::physics::Link, 420
- SetTransform
  - gazebo::common::SkeletonNode, 715
- SetTranslate
  - gazebo::math::Matrix4, 453
- SetTranslation
  - gazebo::common::PoseKeyFrame, 584
- SetTransparency
  - gazebo::common::Material, 442
  - gazebo::rendering::Visual, 866
- SetTransparent
  - gazebo::rendering::Scene, 663

- SetType
  - gazebo::common::NodeTransform, 529
  - gazebo::common::SkeletonNode, 716
- SetUpdateFunc
  - sdf::Param, 542
- SetUpdateRate
  - gazebo::physics::PhysicsEngine, 556
  - gazebo::sensors::Sensor, 678
- SetUserData
  - gazebo::rendering::Grid, 336
- SetValue
  - gazebo::common::NumericKeyFrame, 533
  - sdf::ParamT, 545
- SetVelocity
  - gazebo::physics::Joint, 384
- SetVertex
  - gazebo::common::SubMesh, 743
- SetVertexCount
  - gazebo::common::SubMesh, 743
- SetVerticalAngleMax
  - gazebo::sensors::GpuRaySensor, 330
- SetVerticalAngleMin
  - gazebo::sensors::GpuRaySensor, 330
- SetViewController
  - gazebo::rendering::UserCamera, 802
- SetViewportDimensions
  - gazebo::rendering::UserCamera, 802
- SetVisibilityFlags
  - gazebo::rendering::Visual, 866
- SetVisible
  - gazebo::rendering::Scene, 663
  - gazebo::rendering::Visual, 867
  - gazebo::rendering::WireBox, 874
- SetWindowId
  - gazebo::rendering::Camera, 178
- SetWorld
  - gazebo::physics::Base, 143
- SetWorldCFM
  - gazebo::physics::PhysicsEngine, 556
- SetWorldERP
  - gazebo::physics::PhysicsEngine, 556
- SetWorldPose
  - gazebo::physics::Entity, 283
  - gazebo::rendering::Camera, 178
  - gazebo::rendering::UserCamera, 802
  - gazebo::rendering::Visual, 867
- SetWorldPosition
  - gazebo::rendering::Camera, 179
  - gazebo::rendering::Visual, 867
- SetWorldRotation
  - gazebo::rendering::Camera, 179
  - gazebo::rendering::Visual, 867
- SetWorldTwist
  - gazebo::physics::Entity, 284
- ShadeMode
  - gazebo::common::Material, 437
- shadeMode
  - gazebo::common::Material, 443
- ShadeModeStr
  - gazebo::common::Material, 443
- Shape
  - gazebo::physics::Shape, 693
- shape
  - gazebo::physics::Collision, 199
- Shape.hh, 1050
- ShapePtr
  - gazebo::physics, 94
- shift
  - gazebo::common::MouseEvent, 494
- shininess
  - gazebo::common::Material, 443
- Show
  - gazebo::rendering::GUIOverlay, 341
- ShowBoundingBox
  - gazebo::rendering::Visual, 867
- ShowCOM
  - gazebo::rendering::Visual, 867
- ShowCOMs
  - gazebo::rendering::Scene, 664
- ShowCollision
  - gazebo::rendering::Visual, 867
- ShowCollisions
  - gazebo::rendering::Scene, 663
- ShowContacts
  - gazebo::rendering::Scene, 664
- ShowJoints
  - gazebo::rendering::Scene, 664
  - gazebo::rendering::Visual, 868
- ShowRotation
  - gazebo::rendering::ArrowVisual, 127
  - gazebo::rendering::AxisVisual, 131
- ShowSkeleton
  - gazebo::rendering::Visual, 868
- ShowVisual
  - gazebo::rendering::Light, 403
- ShowWireframe
  - gazebo::rendering::Camera, 179
- Shutdown
  - gazebo::transport::Connection, 223
- sid
  - gazebo::common::NodeTransform, 529
- Signal
  - gazebo::event::EventT, 307–309
- simTime
  - gazebo::physics::State, 732
- SingletonT
  - ~SingletonT, 697
  - Instance, 697



- SingletonT, 697
- SingletonT, 697
- SingletonT< T >, 695
- SingletonT.hh, 1051
- size
  - gazebo::math::Plane, 565
- skelAnimation
  - gazebo::physics::Actor, 113
- skelNodesMap
  - gazebo::physics::Actor, 113
- Skeleton
  - gazebo::common::Skeleton, 699
- skeleton
  - gazebo::physics::Actor, 113
- Skeleton.hh, 1051
- SkeletonAnimation
  - gazebo::common::SkeletonAnimation, 705
- SkeletonAnimation.hh, 1053
- SkeletonNode
  - gazebo::common::SkeletonNode, 710, 711
- SkeletonNodeType
  - gazebo::common::SkeletonNode, 710
- skinFile
  - gazebo::physics::Actor, 113
- skinScale
  - gazebo::physics::Actor, 113
- SkyX, 105
- skyx
  - gazebo::rendering::Scene, 665
- Slerp
  - gazebo::math::Quaternion, 609
- SliderJoint
  - gazebo::physics::SliderJoint, 718
- SliderJoint.hh, 1054
- slip1
  - gazebo::physics::SurfaceParams, 752
- slip2
  - gazebo::physics::SurfaceParams, 753
- SnapVisualToNearestBelow
  - gazebo::rendering::Scene, 664
- source
  - gazebo::common::NodeTransform, 529
- specular
  - gazebo::common::Material, 443
- SphereShape
  - gazebo::physics::SphereShape, 723
- SphereShape.hh, 1056
- SphereShapePtr
  - gazebo::physics, 94
- Spline
  - gazebo::math::Spline, 725
- Spline.hh, 1057
- Squad
  - gazebo::math::Quaternion, 610
- Stamp
  - Messages, 62
- Start
  - gazebo::common::LogRecord, 432
  - gazebo::common::Timer, 781
- startDelay
  - gazebo::physics::Actor, 113
- StartRead
  - gazebo::transport::Connection, 223
- startTime
  - gazebo::physics::TrajectoryInfo, 788
- State
  - gazebo::physics::State, 730
- State.hh, 1058
- Step
  - gazebo::common::LogPlay, 429
- step
  - gazebo::event::Events, 301
- StepWorld
  - gazebo::physics::World, 885
- Stop
  - gazebo::common::LogRecord, 432
  - gazebo::Master, 434
  - gazebo::physics::Actor, 111
  - gazebo::physics::World, 885
  - gazebo::sensors::SensorManager, 684
  - gazebo::Server, 687
  - gazebo::transport::ConnectionManager, 227
  - gazebo::transport::IOManager, 371
- stop
  - gazebo, 81
  - gazebo::event::Events, 301
  - Sensors, 72
  - Transport, 77
- stop\_world
  - Classes for physics and dynamics, 46
- stop\_worlds
  - Classes for physics and dynamics, 46
- StopAnimation
  - gazebo::physics::Entity, 284
  - gazebo::physics::Model, 481
- StopRead
  - gazebo::transport::Connection, 223
- StrStr\_M
  - gazebo::common, 84
- StripSceneName
  - gazebo::rendering::Scene, 664
- StripWorldName
  - gazebo::physics::World, 885
- SubMesh
  - gazebo::common::SubMesh, 736
- SubNodeMap
  - gazebo::transport::TopicManager, 783
- subSampling

- gazebo::physics::HeightmapShape, 348
- Subscribe
  - gazebo::transport::ConnectionManager, 227
  - gazebo::transport::Node, 518–520
  - gazebo::transport::TopicManager, 787
- SubscribeOptions
  - gazebo::transport::SubscribeOptions, 744
- SubscribeOptions.hh, 1061
- Subscriber
  - gazebo::transport::Subscriber, 746
- Subscriber.hh, 1062
- SubscriberPtr
  - gazebo::transport, 102
- SubscriptionTransport
  - gazebo::transport::SubscriptionTransport, 748
- SubscriptionTransport.hh, 1064
- SubscriptionTransportPtr
  - gazebo::transport, 102
- SurfaceParams
  - gazebo::physics::SurfaceParams, 750
- SurfaceParams.hh, 1065
- SurfaceParamsPtr
  - gazebo::physics, 94
- SystemPaths.hh, 1067
  - GetCurrentDir, 1068
  - LINUX, 1068
- SystemPlugin
  - gazebo::SystemPlugin, 758
- SystemPluginPtr
  - gazebo, 81
- systemPluginsArgc
  - gazebo::Server, 688
- systemPluginsArgv
  - gazebo::Server, 688
- TPtr
  - gazebo::PluginT, 570
- TRANSLATE
  - gazebo::common::NodeTransform, 526
- TRIANGLES
  - gazebo::common::SubMesh, 736
- TRIFANS
  - gazebo::common::SubMesh, 736
- TRIMESH\_SHAPE
  - gazebo::physics::Base, 137
- TRISTRIPS
  - gazebo::common::SubMesh, 736
- tangents
  - gazebo::math::RotationSpline, 645
  - gazebo::math::Spline, 728
- tension
  - gazebo::math::Spline, 728
- texImage
  - gazebo::common::Material, 443
- textureHeight
  - gazebo::rendering::Camera, 182
- textureWidth
  - gazebo::rendering::Camera, 182
- threadPitch
  - gazebo::physics::ScrewJoint, 667
- Time
  - gazebo::common::Time, 763, 764
- time
  - gazebo::common::KeyFrame, 395
  - gazebo::physics::Contact, 232
- Time.hh, 1068
- timePos
  - gazebo::common::Animation, 125
- Timer
  - gazebo::common::Timer, 781
- Timer.hh, 1069
- TimerStart
  - gazebo::common::DiagnosticManager, 256
- TimerStop
  - gazebo::common::DiagnosticManager, 256
- ToStdString
  - sdf::Element, 273
  - sdf::SDF, 668
- Toggle
  - gazebo::rendering::Projector, 587
- ToggleShowVisual
  - gazebo::rendering::Light, 403
- ToggleShowWireframe
  - gazebo::rendering::Camera, 179
- ToggleVisible
  - gazebo::rendering::Visual, 868
- TopicManager.hh, 1070
- TrackVisual
  - gazebo::rendering::Camera, 179
- TrackVisualFromSDF
  - Messages, 63
- TrackVisualImpl
  - gazebo::rendering::Camera, 179, 180
  - gazebo::rendering::UserCamera, 802
- trajInfo
  - gazebo::physics::Actor, 113
- trajectories
  - gazebo::physics::Actor, 113
- transform
  - gazebo::common::NodeTransform, 530
  - gazebo::common::SkeletonNode, 717
- TransformAffine
  - gazebo::math::Matrix4, 453
- TransformType
  - gazebo::common::NodeTransform, 526
- Translate
  - gazebo::rendering::Camera, 180
- translate

- gazebo::common::PoseKeyFrame, 585
- translated
  - gazebo::physics::TrajectoryInfo, 788
- transparency
  - gazebo::common::Material, 443
- Transport, 73
  - CallbackHelperPtr, 74
  - clear\_buffers, 75
  - fini, 75
  - get\_master\_uri, 75
  - get\_topic\_namespaces, 75
  - getAdvertisedTopics, 75
  - getTopicMsgType, 76
  - init, 76
  - is\_stopped, 76
  - pause\_incoming, 76
  - request, 76
  - requestNoReply, 77
  - run, 77
  - stop, 77
- Transport.hh, 1072
- TransportTypes.hh, 1074
- TrimeshShape
  - gazebo::physics::TrimeshShape, 790
- TrimeshShape.hh, 1076
- type
  - gazebo::common::MouseEvent, 494
  - gazebo::common::NodeTransform, 530
  - gazebo::common::SkeletonNode, 717
  - gazebo::physics::TrajectoryInfo, 788
  - gazebo::PluginT, 571
- typeName
  - sdf::Param, 542
- typeString
  - gazebo::rendering::ViewController, 849
- UIntGen
  - gazebo::math, 87
- UNION
  - gazebo::common::MeshCSG, 461
- UNIVERSAL\_JOINT
  - gazebo::physics::Base, 137
- UNKNOWN\_PIXEL\_FORMAT
  - gazebo::common::Image, 353
- URDF2Gazebo
  - urdf2gazebo::URDF2Gazebo, 794
- URRealGen
  - gazebo::math, 87
- Unadvertise
  - gazebo::transport::ConnectionManager, 227
  - gazebo::transport::TopicManager, 787
- UniformIntDist
  - gazebo::math, 87
- UniformRealDist
  - gazebo::math, 87
- UniversalJoint
  - gazebo::physics::UniversalJoint, 793
- UniversalJoint.hh, 1077
- Unsubscribe
  - gazebo::transport::ConnectionManager, 227
  - gazebo::transport::Subscriber, 746
  - gazebo::transport::TopicManager, 787
- Update
  - gazebo::common::PID, 562
  - gazebo::physics::Actor, 111
  - gazebo::physics::Base, 143
  - gazebo::physics::Joint, 384
  - gazebo::physics::JointController, 387
  - gazebo::physics::Link, 420
  - gazebo::physics::Model, 481
  - gazebo::physics::MultiRayShape, 513
  - gazebo::physics::RayShape, 626
  - gazebo::physics::TrimeshShape, 791
  - gazebo::rendering::Camera, 180
  - gazebo::rendering::DynamicLines, 262
  - gazebo::rendering::FPSViewController, 315
  - gazebo::rendering::GUIOverlay, 341
  - gazebo::rendering::MovableText, 500
  - gazebo::rendering::OrbitViewController, 537
  - gazebo::rendering::UserCamera, 803
  - gazebo::rendering::ViewController, 849
  - gazebo::rendering::Visual, 868
  - gazebo::sensors::Sensor, 678
  - gazebo::sensors::SensorManager, 685
  - sdf::Element, 274
  - sdf::Param, 542
  - sdf::ParamT, 546
- UpdateChildrenTransforms
  - gazebo::common::SkeletonNode, 716
- UpdateCollision
  - gazebo::physics::PhysicsEngine, 556
- UpdateFromMsg
  - gazebo::rendering::Light, 403
  - gazebo::rendering::Visual, 868
- updateFunc
  - sdf::Param, 543
- UpdateImpl
  - gazebo::sensors::CameraSensor, 187
  - gazebo::sensors::ContactSensor, 238
  - gazebo::sensors::DepthCameraSensor, 253
  - gazebo::sensors::GpuRaySensor, 330
  - gazebo::sensors::ImuSensor, 360
  - gazebo::sensors::MultiCameraSensor, 506
  - gazebo::sensors::RaySensor, 621
  - gazebo::sensors::RFIDSensor, 634
  - gazebo::sensors::RFIDTag, 636
  - gazebo::sensors::Sensor, 678
- UpdateMass

- gazebo::physics::Link, 420
- UpdateParameters
  - gazebo::physics::Actor, 111
  - gazebo::physics::Base, 144
  - gazebo::physics::Collision, 198
  - gazebo::physics::Entity, 284
  - gazebo::physics::Inertial, 367
  - gazebo::physics::Joint, 384
  - gazebo::physics::Link, 420
  - gazebo::physics::Model, 481
- UpdateParams
  - gazebo::rendering::GzTerrainMatGen::SM2Profile, 721
- updateParams
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 692
- UpdateParamsForCompositeMap
  - gazebo::rendering::GzTerrainMatGen::SM2Profile, 721
- updatePeriod
  - gazebo::sensors::Sensor, 680
- UpdatePhysics
  - gazebo::physics::PhysicsEngine, 556
- UpdatePoint
  - gazebo::math::RotationSpline, 645
  - gazebo::math::Spline, 727
- UpdatePublications
  - gazebo::transport::TopicManager, 788
- UpdateRays
  - gazebo::physics::MultiRayShape, 513
- UpdateShaders
  - gazebo::rendering::RTShaderSystem, 650
- UpdateStateSDF
  - gazebo::physics::World, 885
- UpdateSurface
  - gazebo::physics::Link, 420
- updateVpParams
  - gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 692
- urdf2gazebo, 105
- urdf2gazebo::GazeboExtension, 315
- urdf2gazebo::URDF2Gazebo, 793
  - ~URDF2Gazebo, 794
  - InitModelDoc, 794
  - InitModelFile, 794
  - InitModelString, 794
  - URDF2Gazebo, 794
- UrdfCollisionPtr
  - Gazebo\_parser, 68
- UrdfLinkPtr
  - Gazebo\_parser, 68
- UrdfVisualPtr
  - Gazebo\_parser, 68
- UserCamera
  - gazebo::rendering::UserCamera, 797
  - UserCamera.hh, 1078
  - UserCameraPtr
    - gazebo::rendering, 97
- V\_ABOVE
  - gazebo::rendering::MovableText, 497
- V\_BELOW
  - gazebo::rendering::MovableText, 497
- VEL
  - gazebo::physics::Joint, 374
- VERTEX
  - gazebo::rendering::RenderEngine, 629
- VISUAL
  - gazebo::physics::Base, 137
- VISUAL\_PLUGIN
  - Common, 32
- Valid
  - gazebo::common::Image, 357
- ValidateIP
  - gazebo::transport::Connection, 223
- value
  - gazebo::common::NumericKeyFrame, 533
  - sdf::ParamT, 546
- variance
  - Math, 52
- Vector2d
  - gazebo::math::Vector2d, 805
- Vector2d.hh, 1078
- Vector2i
  - gazebo::math::Vector2i, 813
- Vector2i.hh, 1079
- Vector3
  - gazebo::math::Vector3, 823
- Vector3.hh, 1080
- Vector4
  - gazebo::math::Vector4, 835, 836
- Vector4.hh, 1081
- version
  - sdf::SDF, 669
- VertAlign
  - gazebo::rendering::MovableText, 496
- vertElem
  - gazebo::physics::MultiRayShape, 514
  - gazebo::sensors::GpuRaySensor, 332
- vertHalfAngle
  - gazebo::sensors::GpuRaySensor, 332
- vertRangeCount
  - gazebo::sensors::GpuRaySensor, 332
- vertRayCount
  - gazebo::sensors::GpuRaySensor, 332
- vertSize
  - gazebo::physics::HeightmapShape, 348
- vertexBufferCapacity

- gazebo::rendering::DynamicRenderable, 266
- vertexIndex
  - gazebo::common::NodeAssignment, 525
- vfov
  - gazebo::sensors::GpuRaySensor, 332
- Video
  - gazebo::common::Video, 843
- Video.hh, 1083
- VideoVisual
  - gazebo::rendering::VideoVisual, 845
- VideoVisual.hh, 1084
- ViewController
  - gazebo::rendering::ViewController, 847
- ViewController.hh, 1085
- viewport
  - gazebo::rendering::Camera, 182
- visPub
  - gazebo::physics::Entity, 285
- visitRenderables
  - gazebo::rendering::MovableText, 500
- Visual
  - gazebo::rendering::Visual, 854
- Visual.hh, 1086
- VisualFromSDF
  - Messages, 63
- visualMsg
  - gazebo::physics::Entity, 285
- visualName
  - gazebo::physics::Actor, 114
- VisualPlugin
  - gazebo::VisualPlugin, 870
- VisualPluginPtr
  - gazebo, 81
- VisualPtr
  - gazebo::rendering, 97
- visuals
  - gazebo::physics::Link, 421
- w
  - gazebo::math::Quaternion, 610
  - gazebo::math::Vector4, 842
- WORLD\_PLUGIN
  - Common, 32
- WaitForConnection
  - gazebo::transport::Publisher, 597
- wallTime
  - gazebo::physics::State, 732
- weight
  - gazebo::common::NodeAssignment, 525
- White
  - gazebo::common::Color, 214
- windowId
  - gazebo::rendering::Camera, 182
- WindowManager.hh, 1087
- WireBox
  - gazebo::rendering::WireBox, 874
- WireBox.hh, 1087
- World
  - gazebo::physics::World, 877
- world
  - gazebo::physics::Base, 144
  - gazebo::physics::PhysicsEngine, 557
  - gazebo::sensors::Sensor, 680
- World.hh, 1088
- worldCreated
  - gazebo::event::Events, 301
- WorldPlugin
  - gazebo::WorldPlugin, 887
- WorldPluginPtr
  - gazebo, 81
- WorldPtr
  - gazebo::physics, 94
- WorldState
  - gazebo::physics::WorldState, 889
- WorldState.hh, 1090
- worldUpdateEnd
  - gazebo::event::Events, 301
- worldUpdateStart
  - gazebo::event::Events, 301
- wrench
  - gazebo::physics::Contact, 232
- Write
  - sdf::SDF, 669
- x
  - gazebo::math::Quaternion, 610
  - gazebo::math::Vector2d, 811
  - gazebo::math::Vector2i, 820
  - gazebo::math::Vector3, 833
  - gazebo::math::Vector4, 842
- X\_POSITION
  - BVHLoader.hh, 906
- X\_ROTATION
  - BVHLoader.hh, 906
- y
  - gazebo::math::Quaternion, 611
  - gazebo::math::Vector2d, 811
  - gazebo::math::Vector2i, 820
  - gazebo::math::Vector3, 833
  - gazebo::math::Vector4, 842
- Y\_POSITION
  - BVHLoader.hh, 906
- Y\_ROTATION
  - BVHLoader.hh, 906
- Yellow
  - gazebo::common::Color, 214
- z

---

- gazebo::math::Quaternion, 611
- gazebo::math::Vector3, 833
- gazebo::math::Vector4, 843
- Z\_POSITION
  - BVHLoader.hh, 906
- Z\_ROTATION
  - BVHLoader.hh, 906
- ZERO
  - gazebo::math::Matrix4, 454
- Zero
  - gazebo::math::Pose, 580
  - gazebo::math::Vector3, 833