

Gazebo
1.3.0

Generated by Doxygen 1.8.2

Fri Nov 30 2012 15:02:24

Contents

1	Gazebo API Reference	1
2	Todo List	3
3	Module Index	5
3.1	Modules	5
4	Namespace Index	7
4.1	Namespace List	7
5	Hierarchical Index	9
5.1	Class Hierarchy	9
6	Class Index	15
6.1	Class List	15
7	File Index	23
7.1	File List	23
8	Module Documentation	27
8.1	Common	27
8.1.1	Detailed Description	30
8.1.2	Macro Definition Documentation	30
8.1.2.1	DIAG_TIMER	30
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	gzmsg	31
8.1.2.7	gzthrow	31
8.1.2.8	gzwarn	31

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	32
8.1.5.3	ColorErr	32
8.1.5.4	ColorMsg	32
8.1.5.5	DownloadDependencies	33
8.1.5.6	find_file	33
8.1.5.7	find_file_path	33
8.1.5.8	GetManifest	33
8.1.5.9	GetModelFile	34
8.1.5.10	GetModelName	34
8.1.5.11	GetModelPath	34
8.1.5.12	GetModels	34
8.1.5.13	GetModels	35
8.1.5.14	GetURI	35
8.1.5.15	HasModel	35
8.1.5.16	Instance	35
8.1.5.17	Load	35
8.1.5.18	SetQuiet	35
8.2	Events	36
8.2.1	Detailed Description	36
8.2.2	Function Documentation	36
8.2.2.1	~EventT	36
8.2.2.2	Connect	36
8.2.2.3	ConnectionCount	37
8.2.2.4	Disconnect	37
8.2.2.5	Disconnect	38
8.3	Classes for physics and dynamics	39
8.3.1	Detailed Description	42
8.3.2	Macro Definition Documentation	42
8.3.2.1	GZ_REGISTER_PHYSICS_ENGINE	42
8.3.3	Typedef Documentation	42
8.3.3.1	PhysicsFactoryFn	42

8.3.4	Function Documentation	42
8.3.4.1	create_world	42
8.3.4.2	fini	42
8.3.4.3	get_world	42
8.3.4.4	init_world	43
8.3.4.5	init_worlds	43
8.3.4.6	load	43
8.3.4.7	load_world	43
8.3.4.8	load_worlds	43
8.3.4.9	pause_world	43
8.3.4.10	pause_worlds	44
8.3.4.11	remove_worlds	44
8.3.4.12	run_world	44
8.3.4.13	run_worlds	44
8.3.4.14	stop_world	44
8.3.4.15	stop_worlds	44
8.3.5	Variable Documentation	44
8.3.5.1	EntityTypename	44
8.4	Math	46
8.4.1	Detailed Description	47
8.4.2	Function Documentation	48
8.4.2.1	clamp	48
8.4.2.2	equal	48
8.4.2.3	isnan	48
8.4.2.4	isnan	48
8.4.2.5	isPowerOfTwo	49
8.4.2.6	max	49
8.4.2.7	mean	49
8.4.2.8	min	49
8.4.2.9	parseFloat	50
8.4.2.10	parseInt	50
8.4.2.11	precision	50
8.4.2.12	variance	51
8.4.3	Variable Documentation	51
8.4.3.1	NAN_D	51
8.4.3.2	NAN_I	51
8.5	Messages	52

8.5.1	Detailed Description	53
8.5.2	Function Documentation	53
8.5.2.1	Convert	53
8.5.2.2	Convert	54
8.5.2.3	Convert	54
8.5.2.4	Convert	54
8.5.2.5	Convert	54
8.5.2.6	Convert	55
8.5.2.7	Convert	55
8.5.2.8	Convert	55
8.5.2.9	Convert	55
8.5.2.10	Convert	56
8.5.2.11	Convert	56
8.5.2.12	Convert	56
8.5.2.13	CreateRequest	56
8.5.2.14	FogFromSDF	57
8.5.2.15	GetHeader	57
8.5.2.16	GUIFromSDF	57
8.5.2.17	Init	57
8.5.2.18	LightFromSDF	58
8.5.2.19	SceneFromSDF	58
8.5.2.20	Set	58
8.5.2.21	Set	58
8.5.2.22	Set	58
8.5.2.23	Set	59
8.5.2.24	Set	59
8.5.2.25	Set	59
8.5.2.26	Set	59
8.5.2.27	Set	59
8.5.2.28	Set	60
8.5.2.29	Stamp	60
8.5.2.30	Stamp	60
8.5.2.31	TrackVisualFromSDF	60
8.5.2.32	VisualFromSDF	60
8.6	Rendering	61
8.6.1	Detailed Description	63
8.6.2	Function Documentation	63

8.6.2.1	create_scene	63
8.6.2.2	fini	63
8.6.2.3	get_scene	63
8.6.2.4	init	63
8.6.2.5	load	63
8.6.2.6	remove_scene	63
8.7	Gazebo_parser	65
8.7.1	Detailed Description	65
8.7.2	Typedef Documentation	65
8.7.2.1	CollisionPtr	65
8.7.2.2	VisualPtr	65
8.8	Sensors	66
8.8.1	Detailed Description	67
8.8.2	Macro Definition Documentation	67
8.8.2.1	GZ_REGISTER_STATIC_SENSOR	67
8.8.3	Function Documentation	68
8.8.3.1	create_sensor	68
8.8.3.2	fini	68
8.8.3.3	get_sensor	68
8.8.3.4	init	68
8.8.3.5	load	69
8.8.3.6	remove_sensor	69
8.8.3.7	remove_sensors	69
8.8.3.8	run	69
8.8.3.9	run_once	69
8.8.3.10	stop	69
8.9	Transport	70
8.9.1	Detailed Description	71
8.9.2	Typedef Documentation	71
8.9.2.1	CallbackHelperPtr	71
8.9.3	Function Documentation	71
8.9.3.1	clear_buffers	71
8.9.3.2	fini	71
8.9.3.3	get_master_uri	72
8.9.3.4	get_topic_namespaces	72
8.9.3.5	init	72
8.9.3.6	is_stopped	72

8.9.3.7	pause_incoming	72
8.9.3.8	request	73
8.9.3.9	run	73
8.9.3.10	stop	73
9	Namespace Documentation	75
9.1	boost Namespace Reference	75
9.2	gazebo Namespace Reference	75
9.2.1	Detailed Description	76
9.2.2	Typedef Documentation	77
9.2.2.1	GUIPluginPtr	77
9.2.2.2	ModelPluginPtr	77
9.2.2.3	SensorPluginPtr	77
9.2.2.4	SystemPluginPtr	77
9.2.2.5	VisualPluginPtr	77
9.2.2.6	WorldPluginPtr	77
9.2.3	Function Documentation	77
9.2.3.1	add_plugin	77
9.2.3.2	find_file	77
9.2.3.3	fini	77
9.2.3.4	init	77
9.2.3.5	load	77
9.2.3.6	print_version	77
9.2.3.7	run	77
9.2.3.8	stop	77
9.3	gazebo::common Namespace Reference	77
9.3.1	Detailed Description	79
9.3.2	Typedef Documentation	80
9.3.2.1	AnimationPtr	80
9.3.2.2	NodeMap	80
9.3.2.3	NodeMapIter	80
9.3.2.4	NumericAnimationPtr	80
9.3.2.5	Param_V	80
9.3.2.6	PoseAnimationPtr	80
9.3.2.7	RawNodeAnim	80
9.3.2.8	RawNodeWeights	80
9.3.2.9	RawSkeletonAnim	80

9.3.2.10	StrStr_M	80
9.4	gazebo::event Namespace Reference	80
9.4.1	Detailed Description	80
9.4.2	Typedef Documentation	81
9.4.2.1	Connection_V	81
9.4.2.2	ConnectionPtr	81
9.5	gazebo::math Namespace Reference	81
9.5.1	Detailed Description	83
9.5.2	Typedef Documentation	83
9.5.2.1	GeneratorType	83
9.5.2.2	NormalRealDist	83
9.5.2.3	NRealGen	83
9.5.2.4	UIntGen	83
9.5.2.5	UniformIntDist	83
9.5.2.6	UniformRealDist	83
9.5.2.7	URealGen	83
9.6	gazebo::msgs Namespace Reference	83
9.6.1	Detailed Description	85
9.7	gazebo::physics Namespace Reference	85
9.7.1	Detailed Description	88
9.7.2	Typedef Documentation	88
9.7.2.1	Actor_V	88
9.7.2.2	ActorPtr	88
9.7.2.3	Base_V	88
9.7.2.4	BasePtr	88
9.7.2.5	BoxShapePtr	88
9.7.2.6	Collision_V	88
9.7.2.7	CollisionPtr	88
9.7.2.8	ContactPtr	88
9.7.2.9	CylinderShapePtr	88
9.7.2.10	EntityPtr	88
9.7.2.11	HeightmapShapePtr	89
9.7.2.12	InertialPtr	89
9.7.2.13	Joint_V	89
9.7.2.14	JointPtr	89
9.7.2.15	Link_V	89
9.7.2.16	LinkPtr	89

9.7.2.17	MeshShapePtr	89
9.7.2.18	Model_V	89
9.7.2.19	ModelPtr	89
9.7.2.20	MultiRayShapePtr	89
9.7.2.21	PhysicsEnginePtr	89
9.7.2.22	RayShapePtr	89
9.7.2.23	RoadPtr	89
9.7.2.24	ShapePtr	89
9.7.2.25	SphereShapePtr	89
9.7.2.26	SurfaceParamsPtr	89
9.7.2.27	WorldPtr	89
9.8	gazebo::rendering Namespace Reference	89
9.8.1	Detailed Description	92
9.8.2	Typedef Documentation	92
9.8.2.1	ArrowVisualPtr	92
9.8.2.2	AxisVisualPtr	92
9.8.2.3	CameraPtr	92
9.8.2.4	CameraVisualPtr	92
9.8.2.5	COMVisualPtr	92
9.8.2.6	ContactVisualPtr	92
9.8.2.7	DepthCameraPtr	92
9.8.2.8	DynamicLinesPtr	92
9.8.2.9	GpuLaserPtr	92
9.8.2.10	JointVisualPtr	92
9.8.2.11	LaserVisualPtr	92
9.8.2.12	LightPtr	92
9.8.2.13	RFIDTagVisualPtr	92
9.8.2.14	RFIDVisualPtr	92
9.8.2.15	ScenePtr	92
9.8.2.16	UserCameraPtr	92
9.8.2.17	VisualPtr	92
9.8.3	Enumeration Type Documentation	93
9.8.3.1	RenderOpType	93
9.9	gazebo::sensors Namespace Reference	93
9.9.1	Detailed Description	95
9.9.2	Typedef Documentation	95
9.9.2.1	CameraSensor_V	95

9.9.2.2	CameraSensorPtr	95
9.9.2.3	ContactSensor_V	95
9.9.2.4	ContactSensorPtr	95
9.9.2.5	DepthCameraSensor_V	95
9.9.2.6	DepthCameraSensorPtr	95
9.9.2.7	GpuRaySensor_V	95
9.9.2.8	GpuRaySensorPtr	95
9.9.2.9	RaySensor_V	95
9.9.2.10	RaySensorPtr	95
9.9.2.11	RFIDSensor_V	95
9.9.2.12	RFIDSensorPtr	95
9.9.2.13	RFIDTag_V	95
9.9.2.14	RFIDTagPtr	95
9.9.2.15	Sensor_V	95
9.9.2.16	SensorFactoryFn	95
9.9.2.17	SensorPtr	95
9.10	gazebo::transport Namespace Reference	95
9.10.1	Typedef Documentation	97
9.10.1.1	ConnectionPtr	97
9.10.1.2	NodePtr	97
9.10.1.3	PublicationPtr	97
9.10.1.4	PublicationTransportPtr	97
9.10.1.5	PublisherPtr	97
9.10.1.6	SubscriberPtr	97
9.10.1.7	SubscriptionTransportPtr	97
9.11	google Namespace Reference	97
9.12	google::protobuf Namespace Reference	97
9.13	google::protobuf::compiler Namespace Reference	97
9.14	google::protobuf::compiler::cpp Namespace Reference	98
9.15	Ogre Namespace Reference	98
9.16	ogre Namespace Reference	98
9.17	sdf Namespace Reference	98
9.17.1	Detailed Description	99
9.17.2	Typedef Documentation	99
9.17.2.1	ElementPtr	99
9.17.2.2	ElementPtr_V	99
9.17.2.3	Param_V	99

9.17.2.4	ParamPtr	99
9.17.2.5	SDFPtr	99
9.17.3	Function Documentation	99
9.17.3.1	addNestedModel	99
9.17.3.2	copyChildren	99
9.17.3.3	init	99
9.17.3.4	initDoc	99
9.17.3.5	initDoc	99
9.17.3.6	initFile	99
9.17.3.7	initFile	99
9.17.3.8	initString	99
9.17.3.9	initXml	100
9.17.3.10	readDoc	100
9.17.3.11	readDoc	100
9.17.3.12	readFile	100
9.17.3.13	readString	100
9.17.3.14	readString	100
9.17.3.15	readXml	100
9.18	SkyX Namespace Reference	100
9.19	urdf2gazebo Namespace Reference	100
9.19.1	Detailed Description	100
10	Class Documentation	101
10.1	gazebo::physics::Actor Class Reference	101
10.1.1	Detailed Description	103
10.1.2	Constructor & Destructor Documentation	103
10.1.2.1	Actor	103
10.1.2.2	~Actor	104
10.1.3	Member Function Documentation	104
10.1.3.1	Fini	104
10.1.3.2	GetSDF	104
10.1.3.3	Init	104
10.1.3.4	IsActive	104
10.1.3.5	Load	104
10.1.3.6	Play	104
10.1.3.7	Stop	104
10.1.3.8	Update	105

10.1.3.9	UpdateParameters	105
10.1.4	Member Data Documentation	105
10.1.4.1	active	105
10.1.4.2	autoStart	105
10.1.4.3	bonePosePub	105
10.1.4.4	interpolateX	105
10.1.4.5	lastPos	105
10.1.4.6	lastScriptTime	105
10.1.4.7	lastTraj	105
10.1.4.8	loop	106
10.1.4.9	mainLink	106
10.1.4.10	mesh	106
10.1.4.11	oldAction	106
10.1.4.12	pathLength	106
10.1.4.13	playStartTime	106
10.1.4.14	prevFrameTime	106
10.1.4.15	scriptLength	106
10.1.4.16	skelAnimation	106
10.1.4.17	skeleton	106
10.1.4.18	skelNodesMap	106
10.1.4.19	skinFile	107
10.1.4.20	skinScale	107
10.1.4.21	startDelay	107
10.1.4.22	trajectories	107
10.1.4.23	trajInfo	107
10.1.4.24	visualName	107
10.2	gazebo::math::Angle Class Reference	107
10.2.1	Detailed Description	109
10.2.2	Constructor & Destructor Documentation	109
10.2.2.1	Angle	109
10.2.2.2	Angle	109
10.2.2.3	Angle	109
10.2.2.4	~Angle	109
10.2.3	Member Function Documentation	109
10.2.3.1	Degree	109
10.2.3.2	GetAsDegree	109
10.2.3.3	GetAsRadian	110

10.2.3.4	Normalize	110
10.2.3.5	operator!=	110
10.2.3.6	operator*	110
10.2.3.7	operator*	110
10.2.3.8	operator*==	110
10.2.3.9	operator+	111
10.2.3.10	operator+=	111
10.2.3.11	operator-	111
10.2.3.12	operator-=	111
10.2.3.13	operator/	112
10.2.3.14	operator/=	112
10.2.3.15	operator<	112
10.2.3.16	operator<=	112
10.2.3.17	operator==	113
10.2.3.18	operator>	113
10.2.3.19	operator>=	113
10.2.3.20	Radian	113
10.2.3.21	SetFromDegree	114
10.2.3.22	SetFromRadian	114
10.2.4	Friends And Related Function Documentation	114
10.2.4.1	operator<<	114
10.2.4.2	operator>>	114
10.3	gazebo::common::Animation Class Reference	115
10.3.1	Detailed Description	116
10.3.2	Member Typedef Documentation	116
10.3.2.1	KeyFrame_V	116
10.3.3	Constructor & Destructor Documentation	116
10.3.3.1	Animation	116
10.3.3.2	~Animation	117
10.3.4	Member Function Documentation	117
10.3.4.1	AddTime	117
10.3.4.2	GetKeyFrame	117
10.3.4.3	GetKeyFrameCount	117
10.3.4.4	GetKeyFramesAtTime	117
10.3.4.5	GetLength	118
10.3.4.6	GetTime	118
10.3.4.7	SetLength	118

10.3.4.8	SetTime	118
10.3.5	Member Data Documentation	118
10.3.5.1	build	118
10.3.5.2	keyFrames	118
10.3.5.3	length	118
10.3.5.4	loop	119
10.3.5.5	name	119
10.3.5.6	timePos	119
10.4	gazebo::rendering::ArrowVisual Class Reference	119
10.4.1	Detailed Description	120
10.4.2	Constructor & Destructor Documentation	120
10.4.2.1	ArrowVisual	120
10.4.2.2	~ArrowVisual	120
10.4.3	Member Function Documentation	120
10.4.3.1	Load	120
10.4.3.2	ShowRotation	120
10.5	gazebo::rendering::AxisVisual Class Reference	120
10.5.1	Detailed Description	121
10.5.2	Constructor & Destructor Documentation	122
10.5.2.1	AxisVisual	122
10.5.2.2	~AxisVisual	122
10.5.3	Member Function Documentation	122
10.5.3.1	Load	122
10.5.3.2	ScaleXAxis	122
10.5.3.3	ScaleYAxis	122
10.5.3.4	ScaleZAxis	122
10.5.3.5	SetAxisMaterial	123
10.5.3.6	ShowRotation	123
10.6	gazebo::physics::BallJoint< T > Class Template Reference	123
10.6.1	Detailed Description	124
10.6.2	Constructor & Destructor Documentation	124
10.6.2.1	BallJoint	124
10.6.2.2	~BallJoint	124
10.6.3	Member Function Documentation	124
10.6.3.1	GetAngleCount	124
10.6.3.2	GetHighStop	124
10.6.3.3	GetLowStop	124

10.6.3.4	Load	124
10.6.3.5	SetAxis	125
10.6.3.6	SetHighStop	125
10.6.3.7	SetLowStop	125
10.7	gazebo::physics::Base Class Reference	125
10.7.1	Detailed Description	128
10.7.2	Member Enumeration Documentation	128
10.7.2.1	EntityType	128
10.7.3	Constructor & Destructor Documentation	128
10.7.3.1	Base	128
10.7.3.2	~Base	129
10.7.4	Member Function Documentation	129
10.7.4.1	AddChild	129
10.7.4.2	AddType	129
10.7.4.3	Fini	129
10.7.4.4	GetById	129
10.7.4.5	GetByName	129
10.7.4.6	GetChild	130
10.7.4.7	GetChild	130
10.7.4.8	GetChildCount	130
10.7.4.9	GetId	130
10.7.4.10	GetName	131
10.7.4.11	GetParent	131
10.7.4.12	GetParentId	131
10.7.4.13	GetSaveable	131
10.7.4.14	GetScopedName	131
10.7.4.15	GetSDF	131
10.7.4.16	GetType	132
10.7.4.17	GetWorld	132
10.7.4.18	HasType	132
10.7.4.19	Init	132
10.7.4.20	IsSelected	132
10.7.4.21	Load	132
10.7.4.22	operator==	133
10.7.4.23	Print	133
10.7.4.24	RemoveChild	133
10.7.4.25	RemoveChild	133

10.7.4.26	RemoveChildren	133
10.7.4.27	Reset	134
10.7.4.28	Reset	134
10.7.4.29	SetName	134
10.7.4.30	SetParent	134
10.7.4.31	SetSaveable	134
10.7.4.32	SetSelected	134
10.7.4.33	SetWorld	135
10.7.4.34	Update	135
10.7.4.35	UpdateParameters	135
10.7.5	Member Data Documentation	135
10.7.5.1	children	135
10.7.5.2	childrenEnd	135
10.7.5.3	parent	135
10.7.5.4	sdf	135
10.7.5.5	world	136
10.8	gazebo::math::Box Class Reference	136
10.8.1	Detailed Description	137
10.8.2	Constructor & Destructor Documentation	137
10.8.2.1	Box	137
10.8.2.2	Box	137
10.8.2.3	Box	137
10.8.2.4	~Box	137
10.8.3	Member Function Documentation	137
10.8.3.1	GetCenter	138
10.8.3.2	GetSize	138
10.8.3.3	GetXLength	138
10.8.3.4	GetYLength	138
10.8.3.5	GetZLength	138
10.8.3.6	Merge	138
10.8.3.7	operator+	139
10.8.3.8	operator+=	139
10.8.3.9	operator-	139
10.8.3.10	operator=	139
10.8.3.11	operator==	140
10.8.4	Friends And Related Function Documentation	140
10.8.4.1	operator<<	140

10.8.5	Member Data Documentation	140
10.8.5.1	max	140
10.8.5.2	min	140
10.9	gazebo::physics::BoxShape Class Reference	140
10.9.1	Detailed Description	142
10.9.2	Constructor & Destructor Documentation	142
10.9.2.1	BoxShape	142
10.9.2.2	~BoxShape	142
10.9.3	Member Function Documentation	142
10.9.3.1	FillMsg	142
10.9.3.2	FillShapeMsg	142
10.9.3.3	GetInertial	142
10.9.3.4	GetMass	142
10.9.3.5	GetSize	143
10.9.3.6	Init	143
10.9.3.7	ProcessMsg	143
10.9.3.8	SetSize	143
10.10	gazebo::common::BVHLoader Class Reference	143
10.10.1	Detailed Description	144
10.10.2	Constructor & Destructor Documentation	144
10.10.2.1	BVHLoader	144
10.10.2.2	~BVHLoader	144
10.10.3	Member Function Documentation	144
10.10.3.1	Load	144
10.11	gazebo::transport::CallbackHelper Class Reference	144
10.11.1	Detailed Description	145
10.11.2	Constructor & Destructor Documentation	145
10.11.2.1	CallbackHelper	145
10.11.2.2	~CallbackHelper	146
10.11.3	Member Function Documentation	146
10.11.3.1	GetLatching	146
10.11.3.2	GetMsgType	146
10.11.3.3	HandleData	146
10.11.3.4	IsLocal	146
10.11.4	Member Data Documentation	147
10.11.4.1	latching	147
10.12	gazebo::transport::CallbackHelperT< M > Class Template Reference	147

10.12.1 Detailed Description	147
10.12.2 Constructor & Destructor Documentation	148
10.12.2.1 CallbackHelperT	148
10.12.3 Member Function Documentation	148
10.12.3.1 GetMsgType	148
10.12.3.2 HandleData	148
10.12.3.3 IsLocal	148
10.13 gazebo::rendering::Camera Class Reference	149
10.13.1 Detailed Description	154
10.13.2 Constructor & Destructor Documentation	155
10.13.2.1 Camera	155
10.13.2.2 ~Camera	155
10.13.3 Member Function Documentation	155
10.13.3.1 AnimationComplete	155
10.13.3.2 AttachToVisual	155
10.13.3.3 AttachToVisualImpl	155
10.13.3.4 AttachToVisualImpl	156
10.13.3.5 ConnectNewImageFrame	156
10.13.3.6 CreateRenderTexture	156
10.13.3.7 DisconnectNewImageFrame	156
10.13.3.8 EnableSaveFrame	157
10.13.3.9 Fini	157
10.13.3.10 GetAspectRatio	157
10.13.3.11 GetAvgFPS	157
10.13.3.12 GetCameraToViewportRay	157
10.13.3.13 GetDirection	158
10.13.3.14 GetFarClip	158
10.13.3.15 GetFrameFilename	158
10.13.3.16 GetHFOV	158
10.13.3.17 GetImageByteSize	158
10.13.3.18 GetImageByteSize	158
10.13.3.19 GetImageData	159
10.13.3.20 GetImageDepth	159
10.13.3.21 GetImageFormat	159
10.13.3.22 GetImageHeight	159
10.13.3.23 GetImageWidth	159
10.13.3.24 GetInitialized	160

10.13.3.25	GetLastRenderWallTime	160
10.13.3.26	GetName	160
10.13.3.27	GetNearClip	160
10.13.3.28	GetOgreCamera	160
10.13.3.29	GetRenderRate	160
10.13.3.30	GetRenderTexture	161
10.13.3.31	GetRight	161
10.13.3.32	GetScene	161
10.13.3.33	GetSceneNode	161
10.13.3.34	GetTextureHeight	161
10.13.3.35	GetTextureWidth	161
10.13.3.36	GetTriangleCount	162
10.13.3.37	GetUp	162
10.13.3.38	GetVFOV	162
10.13.3.39	GetViewport	162
10.13.3.40	GetViewportHeight	162
10.13.3.41	GetViewportWidth	162
10.13.3.42	GetWindowId	163
10.13.3.43	GetWorldPointOnPlane	163
10.13.3.44	GetWorldPose	163
10.13.3.45	GetWorldPosition	163
10.13.3.46	GetWorldRotation	163
10.13.3.47	GetZValue	164
10.13.3.48	Init	164
10.13.3.49	IsInitialized	164
10.13.3.50	IsVisible	164
10.13.3.51	IsVisible	164
10.13.3.52	Load	165
10.13.3.53	Load	165
10.13.3.54	MoveToPosition	165
10.13.3.55	MoveToPositions	165
10.13.3.56	PostRender	166
10.13.3.57	Render	166
10.13.3.58	RenderImpl	166
10.13.3.59	RotatePitch	166
10.13.3.60	RotateYaw	166
10.13.3.61	SaveFrame	166

10.13.3.62	SaveFrame	166
10.13.3.63	SetAspectRatio	167
10.13.3.64	SetCaptureData	167
10.13.3.65	SetClipDist	167
10.13.3.66	SetHFOV	167
10.13.3.67	SetImageHeight	168
10.13.3.68	SetImageSize	168
10.13.3.69	SetImageWidth	168
10.13.3.70	SetName	168
10.13.3.71	SetRenderRate	168
10.13.3.72	SetRenderTarget	168
10.13.3.73	SetSaveFramePathname	169
10.13.3.74	SetScene	169
10.13.3.75	SetSceneNode	169
10.13.3.76	SetWindowId	169
10.13.3.77	SetWorldPose	169
10.13.3.78	SetWorldPosition	169
10.13.3.79	SetWorldRotation	170
10.13.3.80	ShowWireframe	170
10.13.3.81	ToggleShowWireframe	170
10.13.3.82	TrackVisual	170
10.13.3.83	TrackVisualImpl	170
10.13.3.84	TrackVisualImpl	170
10.13.3.85	Translate	171
10.13.3.86	Update	171
10.13.4	Member Data Documentation	171
10.13.4.1	animState	171
10.13.4.2	bayerFrameBuffer	171
10.13.4.3	camera	171
10.13.4.4	captureData	171
10.13.4.5	connections	171
10.13.4.6	imageFormat	171
10.13.4.7	imageHeight	171
10.13.4.8	imageWidth	172
10.13.4.9	initialized	172
10.13.4.10	lastRenderWallTime	172
10.13.4.11	name	172

10.13.4.12	newData	172
10.13.4.13	newImageFrame	172
10.13.4.14	onAnimationComplete	172
10.13.4.15	pitchNode	172
10.13.4.16	prevAnimTime	172
10.13.4.17	renderTarget	172
10.13.4.18	renderTexture	172
10.13.4.19	requests	173
10.13.4.20	saveCount	173
10.13.4.21	saveFrameBuffer	173
10.13.4.22	scene	173
10.13.4.23	sceneNode	173
10.13.4.24	sdf	173
10.13.4.25	textureHeight	173
10.13.4.26	textureWidth	173
10.13.4.27	viewport	173
10.13.4.28	windowId	173
10.14	gazebo::sensors::CameraSensor Class Reference	173
10.14.1	Detailed Description	175
10.14.2	Constructor & Destructor Documentation	175
10.14.2.1	CameraSensor	175
10.14.2.2	~CameraSensor	175
10.14.3	Member Function Documentation	175
10.14.3.1	Fini	175
10.14.3.2	GetCamera	175
10.14.3.3	GetImageData	175
10.14.3.4	GetImageHeight	176
10.14.3.5	GetImageWidth	176
10.14.3.6	GetTopic	176
10.14.3.7	Init	176
10.14.3.8	Load	176
10.14.3.9	Load	176
10.14.3.10	SaveFrame	177
10.14.3.11	SetParent	177
10.14.3.12	UpdateImpl	177
10.15	gazebo::rendering::CameraVisual Class Reference	177
10.15.1	Detailed Description	178

10.15.2 Constructor & Destructor Documentation	178
10.15.2.1 CameraVisual	178
10.15.2.2 ~CameraVisual	179
10.15.3 Member Function Documentation	179
10.15.3.1 Load	179
10.16 gazebo::common::ColladaLoader Class Reference	179
10.16.1 Detailed Description	180
10.16.2 Constructor & Destructor Documentation	180
10.16.2.1 ColladaLoader	180
10.16.2.2 ~ColladaLoader	180
10.16.3 Member Function Documentation	180
10.16.3.1 Load	180
10.17 gazebo::physics::Collision Class Reference	180
10.17.1 Detailed Description	183
10.17.2 Constructor & Destructor Documentation	183
10.17.2.1 Collision	183
10.17.2.2 ~Collision	183
10.17.3 Member Function Documentation	183
10.17.3.1 AddContact	183
10.17.3.2 ConnectContact	184
10.17.3.3 DisconnectContact	184
10.17.3.4 FillCollisionMsg	184
10.17.3.5 FillMsg	184
10.17.3.6 Fini	184
10.17.3.7 GetBoundingBox	184
10.17.3.8 GetContactsEnabled	184
10.17.3.9 GetLaserRetro	185
10.17.3.10 GetLink	185
10.17.3.11 GetModel	185
10.17.3.12 GetRelativeAngularAccel	185
10.17.3.13 GetRelativeAngularVel	185
10.17.3.14 GetRelativeLinearAccel	185
10.17.3.15 GetRelativeLinearVel	186
10.17.3.16 GetShape	186
10.17.3.17 GetShapeType	186
10.17.3.18 GetState	186
10.17.3.19 GetSurface	186

10.17.3.20	GetWorldAngularAccel	187
10.17.3.21	GetWorldAngularVel	187
10.17.3.22	GetWorldLinearAccel	187
10.17.3.23	GetWorldLinearVel	187
10.17.3.24	Init	187
10.17.3.25	IsPlaceable	187
10.17.3.26	Load	188
10.17.3.27	ProcessMsg	188
10.17.3.28	SetCategoryBits	188
10.17.3.29	SetCollideBits	188
10.17.3.30	SetCollision	188
10.17.3.31	SetContactsEnabled	188
10.17.3.32	SetLaserRetro	189
10.17.3.33	SetShape	189
10.17.3.34	SetState	189
10.17.3.35	UpdateParameters	189
10.17.4	Member Data Documentation	189
10.17.4.1	link	189
10.17.4.2	placeable	189
10.17.4.3	shape	189
10.18	gazebo::physics::CollisionState Class Reference	190
10.18.1	Detailed Description	191
10.18.2	Constructor & Destructor Documentation	191
10.18.2.1	CollisionState	191
10.18.2.2	CollisionState	191
10.18.2.3	CollisionState	191
10.18.2.4	~CollisionState	191
10.18.3	Member Function Documentation	191
10.18.3.1	GetPose	191
10.18.3.2	IsZero	192
10.18.3.3	Load	192
10.18.3.4	operator+	192
10.18.3.5	operator-	192
10.18.3.6	operator=	193
10.18.4	Friends And Related Function Documentation	193
10.18.4.1	operator<<	193
10.19	gazebo::common::Color Class Reference	193

10.19.1 Detailed Description	196
10.19.2 Member Typedef Documentation	196
10.19.2.1 ABGR	196
10.19.2.2 ARGB	196
10.19.2.3 BGRA	196
10.19.2.4 RGBA	196
10.19.3 Constructor & Destructor Documentation	196
10.19.3.1 Color	196
10.19.3.2 Color	196
10.19.3.3 Color	196
10.19.3.4 ~Color	196
10.19.4 Member Function Documentation	196
10.19.4.1 GetAsABGR	196
10.19.4.2 GetAsARGB	197
10.19.4.3 GetAsBGRA	197
10.19.4.4 GetAsHSV	197
10.19.4.5 GetAsRGBA	197
10.19.4.6 GetAsYUV	197
10.19.4.7 operator!=	197
10.19.4.8 operator*	198
10.19.4.9 operator*	198
10.19.4.10operator*=	198
10.19.4.11operator+	198
10.19.4.12operator+	199
10.19.4.13operator+=	199
10.19.4.14operator-	199
10.19.4.15operator-	199
10.19.4.16operator-=	200
10.19.4.17operator/	200
10.19.4.18operator/	200
10.19.4.19operator/=	200
10.19.4.20operator=	201
10.19.4.21operator==	201
10.19.4.22operator[]	201
10.19.4.23Reset	201
10.19.4.24Set	202
10.19.4.25SetFromABGR	202

10.19.4.26	SetFromARGB	202
10.19.4.27	SetFromBGRA	202
10.19.4.28	SetFromHSV	202
10.19.4.29	SetFromRGBA	202
10.19.4.30	SetFromYUV	203
10.19.5	Friends And Related Function Documentation	203
10.19.5.1	operator<<	203
10.19.5.2	operator>>	203
10.19.6	Member Data Documentation	203
10.19.6.1	a	203
10.19.6.2	b	203
10.19.6.3	Black	203
10.19.6.4	Blue	204
10.19.6.5	g	204
10.19.6.6	Green	204
10.19.6.7	Purple	204
10.19.6.8	r	204
10.19.6.9	Red	204
10.19.6.10	White	204
10.19.6.11	Yellow	204
10.20	gazebo::rendering::COMVisual Class Reference	204
10.20.1	Detailed Description	205
10.20.2	Constructor & Destructor Documentation	205
10.20.2.1	COMVisual	205
10.20.2.2	~COMVisual	206
10.20.3	Member Function Documentation	206
10.20.3.1	Load	206
10.20.3.2	Load	206
10.21	gazebo::event::Connection Class Reference	206
10.21.1	Detailed Description	207
10.21.2	Constructor & Destructor Documentation	207
10.21.2.1	Connection	207
10.21.2.2	Connection	207
10.21.2.3	~Connection	207
10.21.3	Member Function Documentation	207
10.21.3.1	GetId	207
10.22	gazebo::transport::Connection Class Reference	207

10.22.1 Detailed Description	209
10.22.2 Member Typedef Documentation	209
10.22.2.1 AcceptCallback	209
10.22.2.2 ReadCallback	209
10.22.3 Constructor & Destructor Documentation	210
10.22.3.1 Connection	210
10.22.3.2 ~Connection	210
10.22.4 Member Function Documentation	210
10.22.4.1 AsyncRead	210
10.22.4.2 Cancel	210
10.22.4.3 Connect	210
10.22.4.4 ConnectToShutdown	210
10.22.4.5 DisconnectShutdown	211
10.22.4.6 EnqueueMsg	211
10.22.4.7 GetId	211
10.22.4.8 GetLocalAddress	211
10.22.4.9 GetLocalHostname	211
10.22.4.10GetLocalPort	211
10.22.4.11GetLocalURI	212
10.22.4.12GetRemoteAddress	212
10.22.4.13GetRemoteHostname	212
10.22.4.14GetRemotePort	212
10.22.4.15GetRemoteURI	212
10.22.4.16IsOpen	212
10.22.4.17Listen	213
10.22.4.18ProcessWriteQueue	213
10.22.4.19Read	213
10.22.4.20Shutdown	213
10.22.4.21StartRead	213
10.22.4.22StopRead	213
10.23gazebo::transport::ConnectionManager Class Reference	213
10.23.1 Detailed Description	215
10.23.2 Member Function Documentation	215
10.23.2.1 Advertise	215
10.23.2.2 ConnectToRemoteHost	215
10.23.2.3 Fini	215
10.23.2.4 GetAllPublishers	215

10.23.2.5	GetTopicNamespaces	216
10.23.2.6	Init	216
10.23.2.7	IsRunning	216
10.23.2.8	RegisterTopicNamespace	216
10.23.2.9	RemoveConnection	216
10.23.2.10	Run	217
10.23.2.11	RunUpdate	217
10.23.2.12	Stop	217
10.23.2.13	Subscribe	217
10.23.2.14	Unadvertise	217
10.23.2.15	Unsubscribe	217
10.23.2.16	Unsubscribe	217
10.23.3	Member Data Documentation	218
10.23.3.1	eventConnections	218
10.24	gazebo::common::Console Class Reference	218
10.24.1	Detailed Description	218
10.25	gazebo::physics::Contact Class Reference	218
10.25.1	Detailed Description	219
10.25.2	Constructor & Destructor Documentation	220
10.25.2.1	Contact	220
10.25.2.2	Contact	220
10.25.2.3	~Contact	220
10.25.3	Member Function Documentation	220
10.25.3.1	Clone	220
10.25.3.2	DebugString	220
10.25.3.3	FillMsg	220
10.25.3.4	operator=	220
10.25.3.5	operator=	221
10.25.3.6	Reset	221
10.25.4	Member Data Documentation	221
10.25.4.1	collision1	221
10.25.4.2	collision2	221
10.25.4.3	count	221
10.25.4.4	depths	221
10.25.4.5	normals	221
10.25.4.6	positions	221
10.25.4.7	time	222

10.25.4.8 wrench	222
10.26gazebo::physics::ContactManager Class Reference	222
10.26.1 Detailed Description	222
10.26.2 Constructor & Destructor Documentation	223
10.26.2.1 ContactManager	223
10.26.2.2 ~ContactManager	223
10.26.3 Member Function Documentation	223
10.26.3.1 Clear	223
10.26.3.2 GetContact	223
10.26.3.3 GetContactCount	223
10.26.3.4 GetContacts	223
10.26.3.5 Init	223
10.26.3.6 NewContact	224
10.26.3.7 PublishContacts	224
10.26.3.8 ResetCount	224
10.27gazebo::sensors::ContactSensor Class Reference	224
10.27.1 Detailed Description	226
10.27.2 Constructor & Destructor Documentation	226
10.27.2.1 ContactSensor	226
10.27.2.2 ~ContactSensor	226
10.27.3 Member Function Documentation	226
10.27.3.1 Fini	226
10.27.3.2 GetCollisionContact	226
10.27.3.3 GetCollisionContactCount	226
10.27.3.4 GetCollisionCount	227
10.27.3.5 GetCollisionName	227
10.27.3.6 GetContacts	227
10.27.3.7 Init	227
10.27.3.8 IsActive	227
10.27.3.9 Load	228
10.27.3.10Load	228
10.27.3.11UpdateImpl	228
10.28gazebo::rendering::ContactVisual Class Reference	228
10.28.1 Detailed Description	229
10.28.2 Constructor & Destructor Documentation	229
10.28.2.1 ContactVisual	229
10.28.2.2 ~ContactVisual	230

10.28.3 Member Function Documentation	230
10.28.3.1 SetEnabled	230
10.29 gazebo::rendering::Conversions Class Reference	230
10.29.1 Detailed Description	230
10.29.2 Member Function Documentation	231
10.29.2.1 Convert	231
10.29.2.2 Convert	231
10.29.2.3 Convert	231
10.29.2.4 Convert	231
10.29.2.5 Convert	232
10.29.2.6 Convert	232
10.30 sdf::Converter Class Reference	232
10.30.1 Detailed Description	232
10.30.2 Member Function Documentation	232
10.30.2.1 Convert	232
10.31 gazebo::physics::CylinderShape Class Reference	233
10.31.1 Detailed Description	234
10.31.2 Constructor & Destructor Documentation	234
10.31.2.1 CylinderShape	234
10.31.2.2 ~CylinderShape	234
10.31.3 Member Function Documentation	234
10.31.3.1 FillMsg	234
10.31.3.2 GetInertial	234
10.31.3.3 GetLength	235
10.31.3.4 GetMass	235
10.31.3.5 GetRadius	235
10.31.3.6 Init	235
10.31.3.7 ProcessMsg	235
10.31.3.8 SetLength	235
10.31.3.9 SetRadius	235
10.31.3.10 SetSize	236
10.32 gazebo::transport::DebugCallbackHelper Class Reference	236
10.32.1 Detailed Description	237
10.32.2 Constructor & Destructor Documentation	237
10.32.2.1 DebugCallbackHelper	237
10.32.3 Member Function Documentation	237
10.32.3.1 GetMsgType	237

10.32.3.2 HandleData	237
10.32.3.3 IsLocal	237
10.33gazebo::rendering::DepthCamera Class Reference	238
10.33.1 Detailed Description	239
10.33.2 Constructor & Destructor Documentation	239
10.33.2.1 DepthCamera	239
10.33.2.2 ~DepthCamera	240
10.33.3 Member Function Documentation	240
10.33.3.1 ConnectNewDepthFrame	240
10.33.3.2 ConnectNewRGBPointCloud	240
10.33.3.3 CreateDepthTexture	240
10.33.3.4 DisconnectNewDepthFrame	240
10.33.3.5 DisconnectNewRGBPointCloud	241
10.33.3.6 Fini	241
10.33.3.7 GetDepthData	241
10.33.3.8 Init	241
10.33.3.9 Load	241
10.33.3.10Load	241
10.33.3.11PostRender	242
10.33.3.12SetDepthTarget	242
10.33.4 Member Data Documentation	242
10.33.4.1 depthTarget	242
10.33.4.2 depthTexture	242
10.33.4.3 depthViewport	242
10.34gazebo::sensors::DepthCameraSensor Class Reference	242
10.34.1 Constructor & Destructor Documentation	244
10.34.1.1 DepthCameraSensor	244
10.34.1.2 ~DepthCameraSensor	244
10.34.2 Member Function Documentation	244
10.34.2.1 Fini	244
10.34.2.2 GetDepthCamera	244
10.34.2.3 Init	244
10.34.2.4 Load	244
10.34.2.5 Load	245
10.34.2.6 SaveFrame	245
10.34.2.7 SetActive	245
10.34.2.8 SetParent	245

10.34.2.9 UpdateImpl	245
10.35gazebo::common::DiagnosticManager Class Reference	246
10.35.1 Detailed Description	247
10.35.2 Member Function Documentation	247
10.35.2.1 CreateTimer	247
10.35.2.2 GetEnabled	247
10.35.2.3 GetLabel	247
10.35.2.4 GetTime	247
10.35.2.5 GetTime	248
10.35.2.6 GetTimerCount	248
10.35.2.7 SetEnabled	248
10.35.2.8 TimerStart	248
10.35.2.9 TimerStop	248
10.36gazebo::common::DiagnosticTimer Class Reference	249
10.36.1 Detailed Description	249
10.36.2 Constructor & Destructor Documentation	249
10.36.2.1 DiagnosticTimer	249
10.36.2.2 ~DiagnosticTimer	250
10.36.3 Member Function Documentation	250
10.36.3.1 GetName	250
10.37gazebo::rendering::DynamicLines Class Reference	250
10.37.1 Detailed Description	252
10.37.2 Constructor & Destructor Documentation	252
10.37.2.1 DynamicLines	252
10.37.2.2 ~DynamicLines	252
10.37.3 Member Function Documentation	252
10.37.3.1 AddPoint	252
10.37.3.2 AddPoint	252
10.37.3.3 Clear	253
10.37.3.4 CreateVertexDeclaration	253
10.37.3.5 FillHardwareBuffers	253
10.37.3.6 GetMovableType	253
10.37.3.7 getMovableType	253
10.37.3.8 GetPoint	253
10.37.3.9 GetPointCount	254
10.37.3.10SetPoint	254
10.37.3.11Update	254

10.38gazebo::rendering::DynamicRenderable Class Reference	254
10.38.1 Detailed Description	256
10.38.2 Constructor & Destructor Documentation	256
10.38.2.1 DynamicRenderable	256
10.38.2.2 ~DynamicRenderable	256
10.38.3 Member Function Documentation	256
10.38.3.1 CreateVertexDeclaration	256
10.38.3.2 FillHardwareBuffers	256
10.38.3.3 getBoundingRadius	257
10.38.3.4 GetOperationType	257
10.38.3.5 getSquaredViewDepth	257
10.38.3.6 Init	257
10.38.3.7 PrepareHardwareBuffers	257
10.38.3.8 SetOperationType	258
10.38.4 Member Data Documentation	258
10.38.4.1 indexBufferCapacity	258
10.38.4.2 vertexBufferCapacity	258
10.39sdf::Element Class Reference	258
10.39.1 Detailed Description	261
10.39.2 Constructor & Destructor Documentation	261
10.39.2.1 Element	261
10.39.2.2 ~Element	261
10.39.3 Member Function Documentation	261
10.39.3.1 AddAttribute	261
10.39.3.2 AddElement	261
10.39.3.3 AddElementDescription	261
10.39.3.4 AddValue	261
10.39.3.5 ClearElements	261
10.39.3.6 Clone	261
10.39.3.7 Copy	261
10.39.3.8 GetAttribute	261
10.39.3.9 GetAttribute	262
10.39.3.10GetAttributeCount	262
10.39.3.11GetAttributeSet	262
10.39.3.12GetCopyChildren	262
10.39.3.13GetDescription	262
10.39.3.14GetElement	262

10.39.3.15	GetElement	262
10.39.3.16	GetElementDescription	262
10.39.3.17	GetElementDescription	262
10.39.3.18	GetElementDescriptionCount	262
10.39.3.19	GetElementImpl	262
10.39.3.20	GetFirstElement	262
10.39.3.21	GetInclude	262
10.39.3.22	GetName	263
10.39.3.23	GetNextElement	263
10.39.3.24	GetParent	263
10.39.3.25	GetRequired	263
10.39.3.26	GetValue	263
10.39.3.27	GetValueBool	263
10.39.3.28	GetValueChar	263
10.39.3.29	GetValueColor	263
10.39.3.30	GetValueDouble	263
10.39.3.31	GetValueFloat	263
10.39.3.32	GetValueInt	263
10.39.3.33	GetValuePose	263
10.39.3.34	GetValueQuaternion	263
10.39.3.35	GetValueString	263
10.39.3.36	GetValueTime	263
10.39.3.37	GetValueUInt	263
10.39.3.38	GetValueVector2d	263
10.39.3.39	GetValueVector3	263
10.39.3.40	HasAttribute	263
10.39.3.41	HasElement	263
10.39.3.42	HasElementDescription	264
10.39.3.43	InsertElement	264
10.39.3.44	PrintDescription	264
10.39.3.45	PrintDocLeftPane	264
10.39.3.46	PrintDocRightPane	264
10.39.3.47	PrintValues	264
10.39.3.48	PrintWiki	264
10.39.3.49	Reset	264
10.39.3.50	Set	264
10.39.3.51	Set	264

10.39.3.52Set	264
10.39.3.53Set	264
10.39.3.54Set	264
10.39.3.55Set	264
10.39.3.56Set	264
10.39.3.57Set	265
10.39.3.58Set	265
10.39.3.59Set	265
10.39.3.60Set	265
10.39.3.61Set	265
10.39.3.62Set	265
10.39.3.63Set	265
10.39.3.64Set	265
10.39.3.65SetCopyChildren	265
10.39.3.66SetDescription	265
10.39.3.67SetInclude	265
10.39.3.68SetName	265
10.39.3.69SetParent	265
10.39.3.70SetRequired	265
10.39.3.71ToString	265
10.39.3.72Update	265
10.40gazebo::physics::Entity Class Reference	265
10.40.1 Detailed Description	268
10.40.2 Constructor & Destructor Documentation	269
10.40.2.1 Entity	269
10.40.2.2 ~Entity	269
10.40.3 Member Function Documentation	269
10.40.3.1 Fini	269
10.40.3.2 GetBoundingBox	269
10.40.3.3 GetChildCollision	269
10.40.3.4 GetChildLink	269
10.40.3.5 GetCollisionBoundingBox	270
10.40.3.6 GetDirtyPose	270
10.40.3.7 GetNearestEntityBelow	270
10.40.3.8 GetParentModel	270
10.40.3.9 GetRelativeAngularAccel	270
10.40.3.10GetRelativeAngularVel	271

10.40.3.11	GetRelativeLinearAccel	271
10.40.3.12	GetRelativeLinearVel	271
10.40.3.13	GetRelativePose	271
10.40.3.14	GetWorldAngularAccel	272
10.40.3.15	GetWorldAngularVel	272
10.40.3.16	GetWorldLinearAccel	272
10.40.3.17	GetWorldLinearVel	272
10.40.3.18	GetWorldPose	272
10.40.3.19	IsCanonicalLink	273
10.40.3.20	IsStatic	273
10.40.3.21	Load	273
10.40.3.22	OnPoseChange	273
10.40.3.23	PlaceOnEntity	273
10.40.3.24	PlaceOnNearestEntityBelow	273
10.40.3.25	Reset	273
10.40.3.26	SetAnimation	274
10.40.3.27	SetAnimation	274
10.40.3.28	SetCanonicalLink	274
10.40.3.29	SetInitialRelativePose	274
10.40.3.30	SetName	274
10.40.3.31	SetRelativePose	275
10.40.3.32	SetStatic	275
10.40.3.33	SetWorldPose	275
10.40.3.34	SetWorldTwist	275
10.40.3.35	StopAnimation	275
10.40.3.36	UpdateParameters	276
10.40.4	Member Data Documentation	276
10.40.4.1	animation	276
10.40.4.2	animationConnection	276
10.40.4.3	animationStartPose	276
10.40.4.4	connections	276
10.40.4.5	dirtyPose	276
10.40.4.6	node	276
10.40.4.7	parentEntity	276
10.40.4.8	poseMsg	276
10.40.4.9	prevAnimationTime	277
10.40.4.10	requestPub	277

10.40.4.11visPub	277
10.40.4.12visualMsg	277
10.41gazebo::event::Event Class Reference	277
10.41.1 Detailed Description	279
10.41.2 Constructor & Destructor Documentation	279
10.41.2.1 ~Event	279
10.41.3 Member Function Documentation	279
10.41.3.1 Disconnect	279
10.41.3.2 Disconnect	279
10.42gazebo::rendering::Events Class Reference	280
10.42.1 Detailed Description	280
10.42.2 Member Function Documentation	280
10.42.2.1 ConnectCreateScene	280
10.42.2.2 ConnectRemoveScene	281
10.42.2.3 ConnectViewContacts	281
10.42.2.4 DisconnectCreateScene	281
10.42.2.5 DisconnectRemoveScene	281
10.42.2.6 DisconnectViewContacts	282
10.42.3 Member Data Documentation	282
10.42.3.1 createScene	282
10.42.3.2 removeScene	282
10.42.3.3 viewContacts	282
10.43gazebo::event::Events Class Reference	282
10.43.1 Detailed Description	285
10.43.2 Member Function Documentation	285
10.43.2.1 ConnectAddEntity	285
10.43.2.2 ConnectCreateEntity	285
10.43.2.3 ConnectDeleteEntity	286
10.43.2.4 ConnectDiagTimerStart	286
10.43.2.5 ConnectDiagTimerStop	286
10.43.2.6 ConnectPause	286
10.43.2.7 ConnectPostRender	287
10.43.2.8 ConnectPreRender	287
10.43.2.9 ConnectRender	287
10.43.2.10ConnectSetSelectedEntity	288
10.43.2.11ConnectStep	288
10.43.2.12ConnectStop	288

10.43.2.13	ConnectWorldCreated	288
10.43.2.14	ConnectWorldUpdateEnd	289
10.43.2.15	ConnectWorldUpdateStart	289
10.43.2.16	DisconnectAddEntity	289
10.43.2.17	DisconnectCreateEntity	290
10.43.2.18	DisconnectDeleteEntity	290
10.43.2.19	DisconnectDiagTimerStart	290
10.43.2.20	DisconnectDiagTimerStop	290
10.43.2.21	DisconnectPause	290
10.43.2.22	DisconnectPostRender	291
10.43.2.23	DisconnectPreRender	291
10.43.2.24	DisconnectRender	291
10.43.2.25	DisconnectSetSelectedEntity	291
10.43.2.26	DisconnectStep	291
10.43.2.27	DisconnectStop	292
10.43.2.28	DisconnectWorldCreated	292
10.43.2.29	DisconnectWorldUpdateEnd	292
10.43.2.30	DisconnectWorldUpdateStart	292
10.43.3	Member Data Documentation	292
10.43.3.1	addEntity	292
10.43.3.2	deleteEntity	293
10.43.3.3	diagTimerStart	293
10.43.3.4	diagTimerStop	293
10.43.3.5	entityCreated	293
10.43.3.6	pause	293
10.43.3.7	postRender	293
10.43.3.8	preRender	293
10.43.3.9	render	293
10.43.3.10	setSelectedEntity	293
10.43.3.11	step	294
10.43.3.12	stop	294
10.43.3.13	worldCreated	294
10.43.3.14	worldUpdateEnd	294
10.43.3.15	worldUpdateStart	294
10.44	gazebo::event::EventT< T > Class Template Reference	294
10.44.1	Detailed Description	297
10.44.2	Member Function Documentation	297

10.44.2.1 operator()	297
10.44.2.2 operator()	297
10.44.2.3 operator()	297
10.44.2.4 operator()	297
10.44.2.5 operator()	298
10.44.2.6 operator()	298
10.44.2.7 operator()	298
10.44.2.8 operator()	298
10.44.2.9 operator()	299
10.44.2.10operator()	299
10.44.2.11operator()	300
10.44.2.12Signal	300
10.44.2.13Signal	300
10.44.2.14Signal	300
10.44.2.15Signal	300
10.44.2.16Signal	301
10.44.2.17Signal	301
10.44.2.18Signal	301
10.44.2.19Signal	302
10.44.2.20Signal	302
10.44.2.21Signal	302
10.44.2.22Signal	303
10.45gazebo::common::Exception Class Reference	303
10.45.1 Detailed Description	304
10.45.2 Constructor & Destructor Documentation	304
10.45.2.1 Exception	304
10.45.2.2 Exception	304
10.45.2.3 ~Exception	304
10.45.3 Member Function Documentation	304
10.45.3.1 GetErrorFile	304
10.45.3.2 GetErrorStr	304
10.45.3.3 Print	304
10.45.4 Friends And Related Function Documentation	305
10.45.4.1 operator<<	305
10.46gazebo::rendering::FPSViewController Class Reference	305
10.46.1 Detailed Description	306
10.46.2 Constructor & Destructor Documentation	306

10.46.2.1 FPSViewController	306
10.46.2.2 ~FPSViewController	306
10.46.3 Member Function Documentation	306
10.46.3.1 GetTypeString	306
10.46.3.2 HandleKeyPressEvent	306
10.46.3.3 HandleKeyReleaseEvent	307
10.46.3.4 HandleMouseEvent	307
10.46.3.5 Init	307
10.46.3.6 Update	307
10.47 urdf2gazebo::GazeboExtension Class Reference	307
10.47.1 Constructor & Destructor Documentation	308
10.47.1.1 GazeboExtension	308
10.47.1.2 GazeboExtension	308
10.47.2 Member Data Documentation	309
10.47.2.1 blobs	309
10.47.2.2 damping_factor	309
10.47.2.3 fdir1	309
10.47.2.4 fudge_factor	309
10.47.2.5 gravity	309
10.47.2.6 initial_joint_position	309
10.47.2.7 is_damping_factor	309
10.47.2.8 is_fudge_factor	309
10.47.2.9 is_initial_joint_position	309
10.47.2.10 s_kd	309
10.47.2.11 s_kp	309
10.47.2.12 s_laser_retro	310
10.47.2.13 s_maxVel	310
10.47.2.14 s_minDepth	310
10.47.2.15 s_mu1	310
10.47.2.16 s_mu2	310
10.47.2.17 s_stop_cfm	310
10.47.2.18 s_stop_erp	310
10.47.2.19 kd	310
10.47.2.20 kp	310
10.47.2.21 laser_retro	310
10.47.2.22 material	310
10.47.2.23 maxVel	311

10.47.2.24	minDepth	311
10.47.2.25	mu1	311
10.47.2.26	mu2	311
10.47.2.27	old_link_name	311
10.47.2.28	provideFeedback	311
10.47.2.29	reduction_transform	311
10.47.2.30	self_collide	311
10.47.2.31	setStaticFlag	311
10.47.2.32	stop_cfm	311
10.47.2.33	stop_erp	311
10.48	google::protobuf::compiler::cpp::GazeboGenerator Class Reference	312
10.48.1	Detailed Description	312
10.48.2	Constructor & Destructor Documentation	312
10.48.2.1	GazeboGenerator	312
10.48.2.2	~GazeboGenerator	312
10.48.3	Member Function Documentation	312
10.48.3.1	Generate	312
10.49	gazebo::rendering::GpuLaser Class Reference	313
10.49.1	Detailed Description	314
10.49.2	Constructor & Destructor Documentation	314
10.49.2.1	GpuLaser	314
10.49.2.2	~GpuLaser	314
10.49.3	Member Function Documentation	314
10.49.3.1	ConnectNewLaserFrame	314
10.49.3.2	CreateLaserTexture	315
10.49.3.3	DisconnectNewLaserFrame	315
10.49.3.4	Fini	315
10.49.3.5	GetLaserData	315
10.49.3.6	Init	315
10.49.3.7	Load	315
10.49.3.8	Load	315
10.49.3.9	notifyRenderSingleObject	315
10.49.3.10	PostRender	316
10.49.3.11	SetParentSensor	316
10.49.3.12	SetRangeCount	316
10.50	gazebo::sensors::GpuRaySensor Class Reference	316
10.50.1	Constructor & Destructor Documentation	320

10.50.1.1 GpuRaySensor	320
10.50.1.2 ~GpuRaySensor	320
10.50.2 Member Function Documentation	320
10.50.2.1 ConnectNewLaserFrame	320
10.50.2.2 DisconnectNewLaserFrame	321
10.50.2.3 Fini	321
10.50.2.4 Get1stRatio	321
10.50.2.5 Get2ndRatio	321
10.50.2.6 GetAngleMax	321
10.50.2.7 GetAngleMin	321
10.50.2.8 GetAngleResolution	322
10.50.2.9 GetCameraCount	322
10.50.2.10GetCHFOV	322
10.50.2.11GetCosHorzFOV	322
10.50.2.12GetCosVertFOV	322
10.50.2.13GetCVFOV	322
10.50.2.14GetFiducial	322
10.50.2.15GetHAngle	323
10.50.2.16GetHFOV	323
10.50.2.17GetHorzFOV	323
10.50.2.18GetHorzHalfAngle	323
10.50.2.19GetLaserCamera	323
10.50.2.20GetRange	324
10.50.2.21GetRangeCount	324
10.50.2.22GetRangeCountRatio	324
10.50.2.23GetRangeMax	324
10.50.2.24GetRangeMin	324
10.50.2.25GetRangeResolution	325
10.50.2.26GetRanges	325
10.50.2.27GetRayCount	325
10.50.2.28GetRayCountRatio	325
10.50.2.29GetRetro	325
10.50.2.30GetVAngle	326
10.50.2.31GetVertFOV	326
10.50.2.32GetVertHalfAngle	326
10.50.2.33GetVerticalAngleMax	326
10.50.2.34GetVerticalAngleMin	326

10.50.2.35	GetVerticalRangeCount	326
10.50.2.36	GetVerticalRayCount	327
10.50.2.37	GetVFOV	327
10.50.2.38	init	327
10.50.2.39	isHorizontal	327
10.50.2.40	Load	327
10.50.2.41	Load	327
10.50.2.42	SetAngleMax	328
10.50.2.43	SetAngleMin	328
10.50.2.44	SetVerticalAngleMax	328
10.50.2.45	SetVerticalAngleMin	328
10.50.2.46	UpdateImpl	328
10.50.3	Member Data Documentation	328
10.50.3.1	cameraCount	328
10.50.3.2	cameraElem	329
10.50.3.3	chfov	329
10.50.3.4	cvfov	329
10.50.3.5	far	329
10.50.3.6	hfov	329
10.50.3.7	horzElem	329
10.50.3.8	horzHalfAngle	329
10.50.3.9	horzRangeCount	329
10.50.3.10	horzRayCount	329
10.50.3.11	isHorizontal	329
10.50.3.12	near	329
10.50.3.13	rangeCountRatio	330
10.50.3.14	rangeElem	330
10.50.3.15	rayCountRatio	330
10.50.3.16	scanElem	330
10.50.3.17	vertElem	330
10.50.3.18	vertHalfAngle	330
10.50.3.19	vertRangeCount	330
10.50.3.20	vertRayCount	330
10.50.3.21	vfov	330
10.51	gazebo::rendering::Grid Class Reference	330
10.51.1	Detailed Description	331
10.51.2	Constructor & Destructor Documentation	331

10.51.2.1 Grid	331
10.51.2.2 ~Grid	332
10.51.3 Member Function Documentation	332
10.51.3.1 Enable	332
10.51.3.2 GetCellCount	332
10.51.3.3 GetCellLength	332
10.51.3.4 GetColor	332
10.51.3.5 GetHeight	332
10.51.3.6 GetLineWidth	333
10.51.3.7 GetSceneNode	333
10.51.3.8 Init	333
10.51.3.9 SetCellCount	333
10.51.3.10SetCellLength	333
10.51.3.11SetColor	333
10.51.3.12SetHeight	333
10.51.3.13SetLineWidth	334
10.51.3.14SetUserData	334
10.52gazebo::physics::Gripper Class Reference	334
10.52.1 Detailed Description	334
10.52.2 Constructor & Destructor Documentation	335
10.52.2.1 Gripper	335
10.52.2.2 ~Gripper	335
10.52.3 Member Function Documentation	335
10.52.3.1 Init	335
10.52.3.2 Load	335
10.53gazebo::rendering::GUIOverlay Class Reference	335
10.53.1 Detailed Description	336
10.53.2 Constructor & Destructor Documentation	336
10.53.2.1 GUIOverlay	336
10.53.2.2 ~GUIOverlay	336
10.53.3 Member Function Documentation	336
10.53.3.1 AttachCameraToImage	336
10.53.3.2 AttachCameraToImage	337
10.53.3.3 ButtonCallback	337
10.53.3.4 CreateWindow	337
10.53.3.5 HandleKeyPressEvent	338
10.53.3.6 HandleKeyReleaseEvent	338

10.53.3.7 HandleMouseEvent	338
10.53.3.8 Hide	338
10.53.3.9 Init	338
10.53.3.10IsInitialized	339
10.53.3.11LoadLayout	339
10.53.3.12Resize	339
10.53.3.13Show	339
10.53.3.14Update	339
10.54gazebo::rendering::Heightmap Class Reference	339
10.54.1 Detailed Description	340
10.54.2 Constructor & Destructor Documentation	340
10.54.2.1 Heightmap	340
10.54.2.2 ~Heightmap	340
10.54.3 Member Function Documentation	340
10.54.3.1 GetHeight	340
10.54.3.2 Load	340
10.54.3.3 LoadFromMsg	340
10.55gazebo::physics::HeightmapShape Class Reference	341
10.55.1 Detailed Description	342
10.55.2 Constructor & Destructor Documentation	343
10.55.2.1 HeightmapShape	343
10.55.2.2 ~HeightmapShape	343
10.55.3 Member Function Documentation	343
10.55.3.1 FillMsg	343
10.55.3.2 GetHeight	343
10.55.3.3 GetMaxHeight	343
10.55.3.4 GetMinHeight	343
10.55.3.5 GetPos	344
10.55.3.6 GetSize	344
10.55.3.7 GetSubSampling	344
10.55.3.8 GetURI	344
10.55.3.9 GetVertexCount	344
10.55.3.10Init	344
10.55.3.11Load	345
10.55.3.12ProcessMsg	345
10.55.4 Member Data Documentation	345
10.55.4.1 heights	345

10.55.4.2	img	345
10.55.4.3	scale	345
10.55.4.4	subSampling	345
10.55.4.5	vertSize	345
10.56	gazebo::physics::Hinge2Joint< T > Class Template Reference	345
10.56.1	Detailed Description	346
10.56.2	Constructor & Destructor Documentation	346
10.56.2.1	Hinge2Joint	346
10.56.2.2	~Hinge2Joint	347
10.56.3	Member Function Documentation	347
10.56.3.1	GetAngleCount	347
10.56.3.2	Load	347
10.57	gazebo::physics::HingeJoint< T > Class Template Reference	347
10.57.1	Detailed Description	348
10.57.2	Constructor & Destructor Documentation	348
10.57.2.1	HingeJoint	348
10.57.2.2	~HingeJoint	348
10.57.3	Member Function Documentation	348
10.57.3.1	GetAngleCount	348
10.57.3.2	Init	348
10.57.3.3	Load	349
10.58	gazebo::common::Image Class Reference	349
10.58.1	Detailed Description	350
10.58.2	Member Enumeration Documentation	350
10.58.2.1	PixelFormat	350
10.58.3	Constructor & Destructor Documentation	351
10.58.3.1	Image	351
10.58.3.2	~Image	351
10.58.4	Member Function Documentation	351
10.58.4.1	GetAvgColor	351
10.58.4.2	GetBPP	351
10.58.4.3	GetData	351
10.58.4.4	GetFilename	352
10.58.4.5	GetHeight	352
10.58.4.6	GetMaxColor	352
10.58.4.7	GetPitch	352
10.58.4.8	GetPixel	352

10.58.4.9 GetPixelFormat	352
10.58.4.10 GetRGBData	353
10.58.4.11 GetWidth	353
10.58.4.12 Load	353
10.58.4.13 Rescale	353
10.58.4.14 SavePNG	353
10.58.4.15 SetFromData	353
10.58.4.16 Valid	354
10.59 gazebo::sensors::ImuSensor Class Reference	354
10.59.1 Detailed Description	355
10.59.2 Constructor & Destructor Documentation	355
10.59.2.1 ImuSensor	355
10.59.2.2 ~ImuSensor	355
10.59.3 Member Function Documentation	355
10.59.3.1 Fini	355
10.59.3.2 GetAngularVelocity	356
10.59.3.3 GetLinearAcceleration	356
10.59.3.4 Init	356
10.59.3.5 Load	356
10.59.3.6 Load	356
10.59.3.7 UpdateImpl	356
10.60 gazebo::physics::Inertial Class Reference	357
10.60.1 Detailed Description	358
10.60.2 Constructor & Destructor Documentation	359
10.60.2.1 Inertial	359
10.60.2.2 Inertial	359
10.60.2.3 Inertial	359
10.60.2.4 ~Inertial	359
10.60.3 Member Function Documentation	359
10.60.3.1 GetCoG	359
10.60.3.2 GetIXX	359
10.60.3.3 GetIXY	359
10.60.3.4 GetIXZ	360
10.60.3.5 GetIYY	360
10.60.3.6 GetIYZ	360
10.60.3.7 GetIZZ	360
10.60.3.8 GetMass	360

10.60.3.9	GetPose	360
10.60.3.10	GetPrincipalMoments	361
10.60.3.11	GetProductsofInertia	361
10.60.3.12	Load	361
10.60.3.13	operator+	361
10.60.3.14	operator+=	361
10.60.3.15	operator=	362
10.60.3.16	ProcessMsg	362
10.60.3.17	Reset	362
10.60.3.18	Rotate	362
10.60.3.19	SetCoG	362
10.60.3.20	SetCoG	362
10.60.3.21	SetInertiaMatrix	363
10.60.3.22	SetIXX	363
10.60.3.23	SetIXY	363
10.60.3.24	SetIXZ	363
10.60.3.25	SetIYY	363
10.60.3.26	SetIYZ	364
10.60.3.27	SetIZZ	364
10.60.3.28	SetMass	364
10.60.3.29	UpdateParameters	364
10.60.4	Friends And Related Function Documentation	364
10.60.4.1	operator<<	364
10.61	gazebo::transport::IOManager Class Reference	364
10.61.1	Detailed Description	365
10.61.2	Constructor & Destructor Documentation	365
10.61.2.1	IOManager	365
10.61.2.2	~IOManager	365
10.61.3	Member Function Documentation	365
10.61.3.1	DecCount	365
10.61.3.2	GetCount	365
10.61.3.3	GetIO	366
10.61.3.4	IncCount	366
10.61.3.5	Stop	366
10.62	gazebo::physics::Joint Class Reference	366
10.62.1	Detailed Description	369
10.62.2	Member Enumeration Documentation	369

10.62.2.1 Attribute	369
10.62.3 Constructor & Destructor Documentation	370
10.62.3.1 Joint	370
10.62.3.2 ~Joint	370
10.62.4 Member Function Documentation	370
10.62.4.1 AreConnected	370
10.62.4.2 Attach	370
10.62.4.3 ConnectJointUpdate	370
10.62.4.4 Detach	371
10.62.4.5 DisconnectJointUpdate	371
10.62.4.6 FillJointMsg	371
10.62.4.7 FillMsg	371
10.62.4.8 GetAnchor	371
10.62.4.9 GetAngle	372
10.62.4.10GetAngleCount	372
10.62.4.11GetAngleImpl	372
10.62.4.12GetChild	372
10.62.4.13GetForce	372
10.62.4.14GetGlobalAxis	373
10.62.4.15GetHighStop	373
10.62.4.16GetJointLink	373
10.62.4.17GetLinkForce	373
10.62.4.18GetLinkTorque	374
10.62.4.19GetLocalAxis	374
10.62.4.20GetLowStop	374
10.62.4.21GetMaxForce	375
10.62.4.22GetParent	375
10.62.4.23GetVelocity	375
10.62.4.24Init	375
10.62.4.25Load	375
10.62.4.26Load	376
10.62.4.27Reset	376
10.62.4.28SetAnchor	376
10.62.4.29SetAngle	376
10.62.4.30SetAttribute	376
10.62.4.31SetAttribute	377
10.62.4.32SetAxis	377

10.62.4.33	SetDamping	377
10.62.4.34	SetForce	377
10.62.4.35	SetHighStop	378
10.62.4.36	SetLowStop	378
10.62.4.37	SetMaxForce	378
10.62.4.38	SetModel	378
10.62.4.39	SetState	378
10.62.4.40	SetVelocity	378
10.62.4.41	Update	379
10.62.4.42	UpdateParameters	379
10.62.5	Member Data Documentation	379
10.62.5.1	anchorLink	379
10.62.5.2	anchorPos	379
10.62.5.3	childLink	379
10.62.5.4	damping_coefficient	379
10.62.5.5	model	379
10.62.5.6	parentLink	379
10.63	gazebo::physics::JointController Class Reference	380
10.63.1	Detailed Description	380
10.63.2	Constructor & Destructor Documentation	380
10.63.2.1	JointController	380
10.63.3	Member Function Documentation	380
10.63.3.1	AddJoint	380
10.63.3.2	Reset	381
10.63.3.3	SetJointPosition	381
10.63.3.4	SetJointPosition	381
10.63.3.5	SetJointPositions	381
10.63.3.6	Update	381
10.64	gazebo::physics::JointState Class Reference	381
10.64.1	Detailed Description	383
10.64.2	Constructor & Destructor Documentation	383
10.64.2.1	JointState	383
10.64.2.2	JointState	383
10.64.2.3	JointState	383
10.64.2.4	~JointState	383
10.64.3	Member Function Documentation	383
10.64.3.1	GetAngle	383

10.64.3.2	GetAngleCount	384
10.64.3.3	GetAngles	384
10.64.3.4	IsZero	384
10.64.3.5	Load	384
10.64.3.6	operator+	384
10.64.3.7	operator-	385
10.64.3.8	operator=	385
10.64.4	Friends And Related Function Documentation	385
10.64.4.1	operator<<	385
10.65	gazebo::rendering::JointVisual Class Reference	385
10.65.1	Detailed Description	386
10.65.2	Constructor & Destructor Documentation	386
10.65.2.1	JointVisual	386
10.65.2.2	~JointVisual	387
10.65.3	Member Function Documentation	387
10.65.3.1	Load	387
10.66	gazebo::physics::JointWrench Class Reference	387
10.66.1	Detailed Description	388
10.66.2	Member Function Documentation	388
10.66.2.1	operator=	388
10.66.3	Member Data Documentation	388
10.66.3.1	body1Force	388
10.66.3.2	body1Torque	388
10.66.3.3	body2Force	388
10.66.3.4	body2Torque	388
10.67	gazebo::common::KeyFrame Class Reference	389
10.67.1	Detailed Description	389
10.67.2	Constructor & Destructor Documentation	389
10.67.2.1	KeyFrame	389
10.67.2.2	~KeyFrame	390
10.67.3	Member Function Documentation	390
10.67.3.1	GetTime	390
10.67.4	Member Data Documentation	390
10.67.4.1	time	390
10.68	gazebo::rendering::LaserVisual Class Reference	390
10.68.1	Detailed Description	391
10.68.2	Constructor & Destructor Documentation	391

10.68.2.1 LaserVisual	391
10.68.2.2 ~LaserVisual	392
10.68.3 Member Function Documentation	392
10.68.3.1 SetEmissive	392
10.69gazebo::rendering::Light Class Reference	392
10.69.1 Detailed Description	393
10.69.2 Constructor & Destructor Documentation	394
10.69.2.1 Light	394
10.69.2.2 ~Light	394
10.69.3 Member Function Documentation	394
10.69.3.1 FillMsg	394
10.69.3.2 GetDiffuseColor	394
10.69.3.3 GetDirection	394
10.69.3.4 GetName	394
10.69.3.5 GetPosition	395
10.69.3.6 GetSpecularColor	395
10.69.3.7 GetType	395
10.69.3.8 Load	395
10.69.3.9 Load	395
10.69.3.10 LoadFromMsg	395
10.69.3.11 OnPoseChange	395
10.69.3.12 SetAttenuation	396
10.69.3.13 SetCastShadows	396
10.69.3.14 SetDiffuseColor	396
10.69.3.15 SetDirection	396
10.69.3.16 SetLightType	396
10.69.3.17 SetName	396
10.69.3.18 SetPosition	397
10.69.3.19 SetRange	397
10.69.3.20 SetSelected	397
10.69.3.21 SetSpecularColor	397
10.69.3.22 SetSpotFalloff	397
10.69.3.23 SetSpotInnerAngle	397
10.69.3.24 SetSpotOuterAngle	398
10.69.3.25 ShowVisual	398
10.69.3.26 ToggleShowVisual	398
10.69.3.27 UpdateFromMsg	398

10.70gazebo::physics::Link Class Reference	398
10.70.1 Detailed Description	403
10.70.2 Constructor & Destructor Documentation	403
10.70.2.1 Link	403
10.70.2.2 ~Link	403
10.70.3 Member Function Documentation	403
10.70.3.1 AddChildJoint	403
10.70.3.2 AddForce	403
10.70.3.3 AddForceAtRelativePosition	403
10.70.3.4 AddForceAtWorldPosition	404
10.70.3.5 AddParentJoint	404
10.70.3.6 AddRelativeForce	404
10.70.3.7 AddRelativeTorque	404
10.70.3.8 AddTorque	404
10.70.3.9 AttachStaticModel	405
10.70.3.10ConnectEnabled	405
10.70.3.11DetachAllStaticModels	405
10.70.3.12DetachStaticModel	405
10.70.3.13DisconnectEnabled	405
10.70.3.14FillLinkMsg	405
10.70.3.15FillMsg	406
10.70.3.16Fini	406
10.70.3.17GetAngularDamping	406
10.70.3.18GetBoundingBox	406
10.70.3.19GetChildJointsLinks	406
10.70.3.20GetCollision	406
10.70.3.21GetCollision	407
10.70.3.22GetCollisionById	407
10.70.3.23GetCollisions	407
10.70.3.24GetEnabled	407
10.70.3.25GetGravityMode	407
10.70.3.26GetInertial	408
10.70.3.27GetKinematic	408
10.70.3.28GetLinearDamping	408
10.70.3.29GetModel	408
10.70.3.30GetParentJointsLinks	408
10.70.3.31GetRelativeAngularAccel	408

10.70.3.32GetRelativeAngularVel 409

10.70.3.33GetRelativeForce 409

10.70.3.34GetRelativeLinearAccel 409

10.70.3.35GetRelativeLinearVel 409

10.70.3.36GetRelativeTorque 409

10.70.3.37GetSelfCollide 409

10.70.3.38GetSensorCount 410

10.70.3.39GetSensorName 410

10.70.3.40GetWorldAngularAccel 410

10.70.3.41GetWorldForce 410

10.70.3.42GetWorldLinearAccel 411

10.70.3.43GetWorldTorque 411

10.70.3.44Init 411

10.70.3.45Load 411

10.70.3.46OnPoseChange 411

10.70.3.47ProcessMsg 411

10.70.3.48RemoveChildJoint 412

10.70.3.49RemoveParentJoint 412

10.70.3.50Reset 412

10.70.3.51SetAngularAccel 412

10.70.3.52SetAngularDamping 412

10.70.3.53SetAngularVel 412

10.70.3.54SetAutoDisable 413

10.70.3.55SetCollideMode 413

10.70.3.56SetEnabled 413

10.70.3.57SetForce 413

10.70.3.58SetGravityMode 413

10.70.3.59SetInertial 413

10.70.3.60SetKinematic 414

10.70.3.61SetLaserRetro 414

10.70.3.62SetLinearAccel 414

10.70.3.63SetLinearDamping 414

10.70.3.64SetLinearVel 414

10.70.3.65SetSelected 414

10.70.3.66SetSelfCollide 415

10.70.3.67SetState 415

10.70.3.68SetTorque 415

10.70.3.69Update	415
10.70.3.70UpdateMass	415
10.70.3.71UpdateParameters	415
10.70.3.72UpdateSurface	416
10.70.4 Member Data Documentation	416
10.70.4.1 angularAccel	416
10.70.4.2 attachedModelsOffset	416
10.70.4.3 cgVisuals	416
10.70.4.4 inertial	416
10.70.4.5 linearAccel	416
10.70.4.6 visuals	416
10.71 gazebo::physics::LinkState Class Reference	416
10.71.1 Detailed Description	418
10.71.2 Constructor & Destructor Documentation	418
10.71.2.1 LinkState	418
10.71.2.2 LinkState	418
10.71.2.3 LinkState	418
10.71.2.4 ~LinkState	419
10.71.3 Member Function Documentation	419
10.71.3.1 GetAcceleration	419
10.71.3.2 GetCollisionState	419
10.71.3.3 GetCollisionState	419
10.71.3.4 GetCollisionStateCount	420
10.71.3.5 GetCollisionStates	420
10.71.3.6 GetPose	420
10.71.3.7 GetVelocity	420
10.71.3.8 GetWrench	420
10.71.3.9 IsZero	420
10.71.3.10 Load	421
10.71.3.11 operator+	421
10.71.3.12 operator-	421
10.71.3.13 operator=	421
10.71.4 Friends And Related Function Documentation	421
10.71.4.1 operator<<	422
10.72 gazebo::common::LogPlay Class Reference	422
10.72.1 Member Function Documentation	423
10.72.1.1 IsOpen	423

10.72.1.2 Open	423
10.72.1.3 Step	423
10.73Logplay Class Reference	423
10.73.1 Detailed Description	423
10.74gazebo::common::LogRecord Class Reference	424
10.74.1 Detailed Description	425
10.74.2 Member Function Documentation	425
10.74.2.1 Add	425
10.74.2.2 GetEncoding	425
10.74.2.3 Init	425
10.74.2.4 Remove	426
10.74.2.5 Start	426
10.74.2.6 Stop	426
10.75gazebo::Master Class Reference	426
10.75.1 Detailed Description	427
10.75.2 Constructor & Destructor Documentation	427
10.75.2.1 Master	427
10.75.2.2 ~Master	427
10.75.3 Member Function Documentation	427
10.75.3.1 Fini	427
10.75.3.2 Init	427
10.75.3.3 Run	428
10.75.3.4 RunOnce	428
10.75.3.5 RunThread	428
10.75.3.6 Stop	428
10.76gazebo::common::Material Class Reference	428
10.76.1 Detailed Description	430
10.76.2 Member Enumeration Documentation	431
10.76.2.1 BlendMode	431
10.76.2.2 ShadeMode	431
10.76.3 Constructor & Destructor Documentation	431
10.76.3.1 Material	431
10.76.3.2 ~Material	431
10.76.3.3 Material	431
10.76.4 Member Function Documentation	431
10.76.4.1 GetAmbient	431
10.76.4.2 GetBlendFactors	432

10.76.4.3	GetBlendMode	432
10.76.4.4	GetDepthWrite	432
10.76.4.5	GetDiffuse	432
10.76.4.6	GetEmissive	432
10.76.4.7	GetLighting	432
10.76.4.8	GetName	433
10.76.4.9	GetPointSize	433
10.76.4.10	GetShadeMode	433
10.76.4.11	GetShininess	433
10.76.4.12	GetSpecular	433
10.76.4.13	GetTextureImage	433
10.76.4.14	GetTransparency	434
10.76.4.15	SetAmbient	434
10.76.4.16	SetBlendFactors	434
10.76.4.17	SetBlendMode	434
10.76.4.18	SetDepthWrite	434
10.76.4.19	SetDiffuse	434
10.76.4.20	SetEmissive	435
10.76.4.21	SetLighting	435
10.76.4.22	SetPointSize	435
10.76.4.23	SetShadeMode	435
10.76.4.24	SetShininess	435
10.76.4.25	SetSpecular	435
10.76.4.26	SetTextureImage	436
10.76.4.27	SetTextureImage	436
10.76.4.28	SetTransparency	436
10.76.5	Friends And Related Function Documentation	436
10.76.5.1	operator<<	436
10.76.6	Member Data Documentation	436
10.76.6.1	ambient	436
10.76.6.2	blendMode	436
10.76.6.3	BlendModeStr	436
10.76.6.4	diffuse	436
10.76.6.5	emissive	437
10.76.6.6	name	437
10.76.6.7	pointSize	437
10.76.6.8	shadeMode	437

10.76.6.9	ShadeModeStr	437
10.76.6.10	shininess	437
10.76.6.11	specular	437
10.76.6.12	texImage	437
10.76.6.13	transparency	437
10.77	gazebo::math::Matrix3 Class Reference	437
10.77.1	Detailed Description	438
10.77.2	Constructor & Destructor Documentation	438
10.77.2.1	Matrix3	438
10.77.2.2	Matrix3	439
10.77.2.3	Matrix3	439
10.77.2.4	~Matrix3	439
10.77.3	Member Function Documentation	439
10.77.3.1	operator==	439
10.77.3.2	operator[]	439
10.77.3.3	operator[]	440
10.77.3.4	SetCol	440
10.77.3.5	SetFromAxes	440
10.77.3.6	SetFromAxis	440
10.77.4	Friends And Related Function Documentation	440
10.77.4.1	operator<<	441
10.77.5	Member Data Documentation	441
10.77.5.1	m	441
10.78	gazebo::math::Matrix4 Class Reference	441
10.78.1	Detailed Description	443
10.78.2	Constructor & Destructor Documentation	443
10.78.2.1	Matrix4	443
10.78.2.2	Matrix4	443
10.78.2.3	Matrix4	443
10.78.2.4	~Matrix4	443
10.78.3	Member Function Documentation	443
10.78.3.1	GetAsPose	444
10.78.3.2	GetEulerRotation	444
10.78.3.3	GetRotation	444
10.78.3.4	GetTranslation	444
10.78.3.5	Inverse	444
10.78.3.6	IsAffine	444

10.78.3.7 operator*	444
10.78.3.8 operator*	445
10.78.3.9 operator*	445
10.78.3.10operator=	445
10.78.3.11operator=	445
10.78.3.12operator==	446
10.78.3.13operator[]	446
10.78.3.14operator[]	446
10.78.3.15Set	446
10.78.3.16SetScale	447
10.78.3.17SetTranslate	447
10.78.3.18TransformAffine	447
10.78.4 Friends And Related Function Documentation	447
10.78.4.1 operator<<	448
10.78.5 Member Data Documentation	448
10.78.5.1 IDENTITY	448
10.78.5.2 m	448
10.78.5.3 ZERO	448
10.79gazebo::common::Mesh Class Reference	448
10.79.1 Detailed Description	449
10.79.2 Constructor & Destructor Documentation	450
10.79.2.1 Mesh	450
10.79.2.2 ~Mesh	450
10.79.3 Member Function Documentation	450
10.79.3.1 AddMaterial	450
10.79.3.2 AddSubMesh	450
10.79.3.3 FillArrays	450
10.79.3.4 GenSphericalTexCoord	450
10.79.3.5 GetAABB	451
10.79.3.6 GetIndexCount	451
10.79.3.7 GetMaterial	451
10.79.3.8 GetMaterialCount	451
10.79.3.9 GetMax	451
10.79.3.10GetMin	452
10.79.3.11GetName	452
10.79.3.12GetNormalCount	452
10.79.3.13GetPath	452

10.79.3.14	GetSkeleton	452
10.79.3.15	GetSubMesh	452
10.79.3.16	GetSubMeshCount	453
10.79.3.17	GetTexCoordCount	453
10.79.3.18	GetVertexCount	453
10.79.3.19	HasSkeleton	453
10.79.3.20	RecalculateNormals	453
10.79.3.21	Scale	453
10.79.3.22	SetName	453
10.79.3.23	SetPath	454
10.79.3.24	SetSkeleton	454
10.80	gazebo::common::MeshLoader Class Reference	454
10.80.1	Detailed Description	455
10.80.2	Constructor & Destructor Documentation	455
10.80.2.1	MeshLoader	455
10.80.2.2	~MeshLoader	455
10.80.3	Member Function Documentation	455
10.80.3.1	Load	455
10.81	gazebo::common::MeshManager Class Reference	455
10.81.1	Detailed Description	457
10.81.2	Member Function Documentation	457
10.81.2.1	AddMesh	457
10.81.2.2	CreateBox	457
10.81.2.3	CreateCamera	457
10.81.2.4	CreateCone	457
10.81.2.5	CreateCylinder	458
10.81.2.6	CreatePlane	458
10.81.2.7	CreatePlane	458
10.81.2.8	CreateSphere	458
10.81.2.9	CreateTube	459
10.81.2.10	GenSphericalTexCoord	459
10.81.2.11	GetMesh	459
10.81.2.12	GetMeshAABB	459
10.81.2.13	HasMesh	460
10.81.2.14	IsValidFilename	460
10.81.2.15	Load	460
10.82	gazebo::physics::Model Class Reference	460

10.82.1 Detailed Description	464
10.82.2 Constructor & Destructor Documentation	464
10.82.2.1 Model	464
10.82.2.2 ~Model	464
10.82.3 Member Function Documentation	464
10.82.3.1 AttachStaticModel	464
10.82.3.2 DetachStaticModel	465
10.82.3.3 FillModelMsg	465
10.82.3.4 FillMsg	465
10.82.3.5 Fini	465
10.82.3.6 GetAllLinks	465
10.82.3.7 GetAutoDisable	465
10.82.3.8 GetBoundingBox	465
10.82.3.9 GetJoint	466
10.82.3.10GetJoint	466
10.82.3.11GetJointCount	466
10.82.3.12GetJoints	466
10.82.3.13GetLink	466
10.82.3.14GetLink	467
10.82.3.15GetLinkById	467
10.82.3.16GetLinks	467
10.82.3.17GetPluginCount	467
10.82.3.18GetRelativeAngularAccel	467
10.82.3.19GetRelativeAngularVel	468
10.82.3.20GetRelativeLinearAccel	468
10.82.3.21GetRelativeLinearVel	468
10.82.3.22GetSDF	468
10.82.3.23GetSensorCount	468
10.82.3.24GetWorldAngularAccel	469
10.82.3.25GetWorldAngularVel	469
10.82.3.26GetWorldLinearAccel	469
10.82.3.27GetWorldLinearVel	469
10.82.3.28Init	469
10.82.3.29Load	469
10.82.3.30LoadPlugins	470
10.82.3.31OnPoseChange	470
10.82.3.32ProcessMsg	470

10.82.3.33	RemoveChild	470
10.82.3.34	Reset	470
10.82.3.35	SetAngularAccel	470
10.82.3.36	SetAngularVel	471
10.82.3.37	SetAutoDisable	471
10.82.3.38	SetCollideMode	471
10.82.3.39	SetEnabled	471
10.82.3.40	SetGravityMode	471
10.82.3.41	SetJointAnimation	471
10.82.3.42	SetJointPosition	472
10.82.3.43	SetJointPositions	472
10.82.3.44	SetLaserRetro	472
10.82.3.45	SetLinearAccel	472
10.82.3.46	SetLinearVel	473
10.82.3.47	SetLinkWorldPose	473
10.82.3.48	SetLinkWorldPose	473
10.82.3.49	SetState	473
10.82.3.50	StopAnimation	473
10.82.3.51	Update	473
10.82.3.52	UpdateParameters	474
10.82.4	Member Data Documentation	474
10.82.4.1	attachedModels	474
10.82.4.2	attachedModelsOffset	474
10.83	gazebo::common::ModelDatabase Class Reference	474
10.83.1	Detailed Description	475
10.84	gazebo::ModelPlugin Class Reference	475
10.84.1	Detailed Description	476
10.84.2	Constructor & Destructor Documentation	476
10.84.2.1	ModelPlugin	476
10.84.2.2	~ModelPlugin	476
10.84.3	Member Function Documentation	476
10.84.3.1	Init	476
10.84.3.2	Load	476
10.84.3.3	Reset	477
10.85	gazebo::physics::ModelState Class Reference	477
10.85.1	Detailed Description	478
10.85.2	Constructor & Destructor Documentation	478

10.85.2.1	ModelState	478
10.85.2.2	ModelState	478
10.85.2.3	ModelState	479
10.85.2.4	~ModelState	479
10.85.3	Member Function Documentation	479
10.85.3.1	GetJointState	479
10.85.3.2	GetJointState	479
10.85.3.3	GetJointStateCount	480
10.85.3.4	GetJointStates	480
10.85.3.5	GetLinkState	480
10.85.3.6	GetLinkState	480
10.85.3.7	GetLinkStateCount	481
10.85.3.8	GetLinkStates	481
10.85.3.9	GetPose	481
10.85.3.10	IsZero	481
10.85.3.11	Load	481
10.85.3.12	operator+	482
10.85.3.13	operator-	482
10.85.3.14	operator=	482
10.85.4	Friends And Related Function Documentation	482
10.85.4.1	operator<<	482
10.86	gazebo::common::MouseEvent Class Reference	483
10.86.1	Detailed Description	484
10.86.2	Member Enumeration Documentation	484
10.86.2.1	Buttons	484
10.86.2.2	EventType	484
10.86.3	Constructor & Destructor Documentation	484
10.86.3.1	MouseEvent	484
10.86.4	Member Data Documentation	484
10.86.4.1	alt	484
10.86.4.2	button	485
10.86.4.3	buttons	485
10.86.4.4	control	485
10.86.4.5	dragging	485
10.86.4.6	moveScale	485
10.86.4.7	pos	485
10.86.4.8	pressPos	485

10.86.4.9 prevPos	485
10.86.4.10 scroll	485
10.86.4.11 shift	485
10.86.4.12 type	485
10.87 gazebo::rendering::MovableText Class Reference	486
10.87.1 Detailed Description	487
10.87.2 Member Enumeration Documentation	487
10.87.2.1 HorizAlign	487
10.87.2.2 VertAlign	488
10.87.3 Constructor & Destructor Documentation	488
10.87.3.1 MovableText	488
10.87.3.2 ~MovableText	488
10.87.4 Member Function Documentation	488
10.87.4.1 _setupGeometry	488
10.87.4.2 _updateColors	488
10.87.4.3 GetAABB	488
10.87.4.4 GetBaseline	488
10.87.4.5 getBoundingRadius	489
10.87.4.6 GetCharHeight	489
10.87.4.7 GetColor	489
10.87.4.8 GetFont	489
10.87.4.9 getLights	489
10.87.4.10 getMaterial	489
10.87.4.11 getRenderOperation	489
10.87.4.12 GetShowOnTop	489
10.87.4.13 GetSpaceWidth	489
10.87.4.14 getSquaredViewDepth	489
10.87.4.15 GetText	489
10.87.4.16 getWorldTransforms	490
10.87.4.17 Load	490
10.87.4.18 SetBaseline	490
10.87.4.19 SetCharHeight	490
10.87.4.20 SetColor	490
10.87.4.21 SetFontName	490
10.87.4.22 SetShowOnTop	491
10.87.4.23 SetSpaceWidth	491
10.87.4.24 SetText	491

10.87.4.25	SetTextAlignment	491
10.87.4.26	Update	491
10.87.4.27	visitRenderables	491
10.88	gazebo::physics::MultiRayShape Class Reference	492
10.88.1	Detailed Description	494
10.88.2	Constructor & Destructor Documentation	494
10.88.2.1	MultiRayShape	494
10.88.2.2	~MultiRayShape	494
10.88.3	Member Function Documentation	494
10.88.3.1	AddRay	494
10.88.3.2	ConnectNewLaserScans	494
10.88.3.3	DisconnectNewLaserScans	495
10.88.3.4	FillMsg	495
10.88.3.5	GetFiducial	495
10.88.3.6	GetMaxAngle	495
10.88.3.7	GetMaxRange	496
10.88.3.8	GetMinAngle	496
10.88.3.9	GetMinRange	496
10.88.3.10	GetRange	496
10.88.3.11	GetResRange	496
10.88.3.12	GetRetro	496
10.88.3.13	GetSampleCount	497
10.88.3.14	GetScanResolution	497
10.88.3.15	GetVerticalMaxAngle	497
10.88.3.16	GetVerticalMinAngle	497
10.88.3.17	GetVerticalSampleCount	497
10.88.3.18	GetVerticalScanResolution	498
10.88.3.19	init	498
10.88.3.20	ProcessMsg	498
10.88.3.21	Update	498
10.88.3.22	UpdateRays	498
10.88.4	Member Data Documentation	498
10.88.4.1	horzElem	498
10.88.4.2	newLaserScans	498
10.88.4.3	offset	498
10.88.4.4	rangeElem	499
10.88.4.5	rayElem	499

10.88.4.6 rays	499
10.88.4.7 scanElem	499
10.88.4.8 vertElem	499
10.89 gazebo::transport::Node Class Reference	499
10.89.1 Detailed Description	500
10.89.2 Constructor & Destructor Documentation	500
10.89.2.1 Node	500
10.89.2.2 ~Node	501
10.89.3 Member Function Documentation	501
10.89.3.1 Advertise	501
10.89.3.2 DecodeTopicName	501
10.89.3.3 EncodeTopicName	501
10.89.3.4 Fini	502
10.89.3.5 GetId	502
10.89.3.6 GetMsgType	502
10.89.3.7 GetTopicNamespace	502
10.89.3.8 HandleData	502
10.89.3.9 Init	502
10.89.3.10 InsertLatchedMsg	503
10.89.3.11 ProcessIncoming	503
10.89.3.12 ProcessPublishers	503
10.89.3.13 Subscribe	503
10.89.3.14 Subscribe	503
10.90 gazebo::common::NodeAnimation Class Reference	504
10.90.1 Detailed Description	505
10.90.2 Constructor & Destructor Documentation	505
10.90.2.1 NodeAnimation	505
10.90.2.2 ~NodeAnimation	505
10.90.3 Member Function Documentation	505
10.90.3.1 AddKeyFrame	505
10.90.3.2 AddKeyFrame	505
10.90.3.3 GetFrameAt	506
10.90.3.4 GetFrameCount	506
10.90.3.5 GetKeyFrame	506
10.90.3.6 GetKeyFrame	506
10.90.3.7 GetLength	506
10.90.3.8 GetName	507

10.90.3.9 GetTimeAtX	507
10.90.3.10Scale	507
10.90.3.11SetName	507
10.90.4 Member Data Documentation	507
10.90.4.1 keyFrames	507
10.90.4.2 length	507
10.90.4.3 name	508
10.91 gazebo::common::NodeAssignment Struct Reference	508
10.91.1 Detailed Description	508
10.91.2 Member Data Documentation	508
10.91.2.1 nodeIndex	508
10.91.2.2 vertexIndex	508
10.91.2.3 weight	508
10.92 gazebo::common::NodeTransform Class Reference	509
10.92.1 Detailed Description	510
10.92.2 Member Enumeration Documentation	510
10.92.2.1 TransformType	510
10.92.3 Constructor & Destructor Documentation	510
10.92.3.1 NodeTransform	510
10.92.3.2 NodeTransform	510
10.92.3.3 ~NodeTransform	511
10.92.4 Member Function Documentation	511
10.92.4.1 Get	511
10.92.4.2 GetSID	511
10.92.4.3 GetType	511
10.92.4.4 operator()	511
10.92.4.5 operator*	511
10.92.4.6 operator*	512
10.92.4.7 PrintSource	512
10.92.4.8 RecalculateMatrix	512
10.92.4.9 Set	512
10.92.4.10SetComponent	512
10.92.4.11SetSID	512
10.92.4.12SetSourceValues	513
10.92.4.13SetSourceValues	513
10.92.4.14SetSourceValues	513
10.92.4.15SetType	513

10.92.5 Member Data Documentation	513
10.92.5.1 sid	513
10.92.5.2 source	513
10.92.5.3 transform	513
10.92.5.4 type	513
10.93 gazebo::common::NumericAnimation Class Reference	514
10.93.1 Detailed Description	514
10.93.2 Constructor & Destructor Documentation	514
10.93.2.1 NumericAnimation	514
10.93.2.2 ~NumericAnimation	515
10.93.3 Member Function Documentation	515
10.93.3.1 CreateKeyFrame	515
10.93.3.2 GetInterpolatedKeyFrame	515
10.94 gazebo::common::NumericKeyFrame Class Reference	515
10.94.1 Detailed Description	516
10.94.2 Constructor & Destructor Documentation	516
10.94.2.1 NumericKeyFrame	516
10.94.2.2 ~NumericKeyFrame	517
10.94.3 Member Function Documentation	517
10.94.3.1 GetValue	517
10.94.3.2 SetValue	517
10.94.4 Member Data Documentation	517
10.94.4.1 value	517
10.95 gazebo::rendering::OrbitViewController Class Reference	517
10.95.1 Detailed Description	519
10.95.2 Constructor & Destructor Documentation	519
10.95.2.1 OrbitViewController	519
10.95.2.2 ~OrbitViewController	519
10.95.3 Member Function Documentation	519
10.95.3.1 GetFocalPoint	519
10.95.3.2 GetTypeString	519
10.95.3.3 HandleKeyPressEvent	519
10.95.3.4 HandleKeyReleaseEvent	520
10.95.3.5 HandleMouseEvent	520
10.95.3.6 Init	520
10.95.3.7 Init	520
10.95.3.8 SetDistance	520

10.95.3.9 SetDistanceRange	521
10.95.3.10SetFocalPoint	521
10.95.3.11SetPitch	521
10.95.3.12SetYaw	521
10.95.3.13Update	521
10.96sdf::Param Class Reference	521
10.96.1 Detailed Description	523
10.96.2 Constructor & Destructor Documentation	524
10.96.2.1 Param	524
10.96.2.2 ~Param	524
10.96.3 Member Function Documentation	524
10.96.3.1 Clone	524
10.96.3.2 Get	524
10.96.3.3 Get	524
10.96.3.4 Get	524
10.96.3.5 Get	524
10.96.3.6 Get	524
10.96.3.7 Get	524
10.96.3.8 Get	524
10.96.3.9 Get	524
10.96.3.10Get	524
10.96.3.11Get	524
10.96.3.12Get	524
10.96.3.13Get	524
10.96.3.14Get	524
10.96.3.15Get	524
10.96.3.16GetAsString	524
10.96.3.17GetDefaultAsString	525
10.96.3.18GetDescription	525
10.96.3.19GetKey	525
10.96.3.20GetRequired	525
10.96.3.21GetSet	525
10.96.3.22GetTypeNames	525
10.96.3.23sBool	525
10.96.3.24sChar	525
10.96.3.25sColor	525
10.96.3.26sDouble	525

10.96.3.27	sFloat	525
10.96.3.28	sInt	525
10.96.3.29	sPose	525
10.96.3.30	sQuaternion	525
10.96.3.31	sStr	525
10.96.3.32	sTime	525
10.96.3.33	sUInt	525
10.96.3.34	sVector2d	525
10.96.3.35	sVector2i	525
10.96.3.36	sVector3	526
10.96.3.37	Reset	526
10.96.3.38	Set	526
10.96.3.39	Set	526
10.96.3.40	Set	526
10.96.3.41	Set	526
10.96.3.42	Set	526
10.96.3.43	Set	526
10.96.3.44	Set	526
10.96.3.45	Set	526
10.96.3.46	Set	526
10.96.3.47	Set	526
10.96.3.48	Set	526
10.96.3.49	Set	526
10.96.3.50	Set	526
10.96.3.51	Set	526
10.96.3.52	Set	526
10.96.3.53	setDescription	526
10.96.3.54	setFromString	526
10.96.3.55	setUpdateFunc	527
10.96.3.56	Update	527
10.96.4	Member Data Documentation	527
10.96.4.1	description	527
10.96.4.2	key	527
10.96.4.3	required	527
10.96.4.4	set	527
10.96.4.5	typeName	527
10.96.4.6	updateFunc	527

10.97sdf::ParamT< T > Class Template Reference	527
10.97.1 Detailed Description	529
10.97.2 Constructor & Destructor Documentation	529
10.97.2.1 ParamT	529
10.97.2.2 ~ParamT	529
10.97.3 Member Function Documentation	529
10.97.3.1 Clone	529
10.97.3.2 GetAsString	529
10.97.3.3 GetDefaultAsString	529
10.97.3.4 GetDefaultValue	529
10.97.3.5 GetValue	529
10.97.3.6 operator*	530
10.97.3.7 Reset	530
10.97.3.8 Set	530
10.97.3.9 SetFromString	530
10.97.3.10SetValue	530
10.97.3.11Update	530
10.97.4 Friends And Related Function Documentation	530
10.97.4.1 operator<<	530
10.97.5 Member Data Documentation	530
10.97.5.1 defaultValue	530
10.97.5.2 value	530
10.98ParamT< T > Class Template Reference	531
10.99gazebo::physics::PhysicsEngine Class Reference	531
10.99.1 Detailed Description	533
10.99.2 Constructor & Destructor Documentation	533
10.99.2.1 PhysicsEngine	533
10.99.2.2 ~PhysicsEngine	534
10.99.3 Member Function Documentation	534
10.99.3.1 CreateCollision	534
10.99.3.2 CreateCollision	534
10.99.3.3 CreateJoint	534
10.99.3.4 CreateLink	534
10.99.3.5 CreateShape	534
10.99.3.6 DebugPrint	535
10.99.3.7 Fini	535
10.99.3.8 GetAutoDisableFlag	535

10.99.3.9	GetContactManager	535
10.99.3.10	GetContactMaxCorrectingVel	535
10.99.3.11	GetContactSurfaceLayer	535
10.99.3.12	GetGravity	536
10.99.3.13	GetMaxContacts	536
10.99.3.14	GetPhysicsUpdateMutex	536
10.99.3.15	GetSORPGSIters	536
10.99.3.16	GetSORPGSPreconIters	536
10.99.3.17	GetSORPGSW	537
10.99.3.18	GetStepTime	537
10.99.3.19	GetUpdatePeriod	537
10.99.3.20	GetUpdateRate	537
10.99.3.21	GetWorldCFM	537
10.99.3.22	GetWorldERP	537
10.99.3.23	Init	538
10.99.3.24	InitForThread	538
10.99.3.25	Load	538
10.99.3.26	OnPhysicsMsg	538
10.99.3.27	OnRequest	538
10.99.3.28	Reset	538
10.99.3.29	SetAutoDisableFlag	538
10.99.3.30	SetContactMaxCorrectingVel	539
10.99.3.31	SetContactSurfaceLayer	539
10.99.3.32	SetGravity	539
10.99.3.33	SetMaxContacts	539
10.99.3.34	SetSORPGSIters	539
10.99.3.35	SetSORPGSPreconIters	540
10.99.3.36	SetSORPGSW	540
10.99.3.37	SetStepTime	540
10.99.3.38	SetUpdateRate	540
10.99.3.39	SetWorldCFM	540
10.99.3.40	SetWorldERP	540
10.99.3.41	UpdateCollision	541
10.99.3.42	UpdatePhysics	541
10.99.4	Member Data Documentation	541
10.99.4.1	contactManager	541
10.99.4.2	node	541

10.99.4.3 physicsSub	541
10.99.4.4 physicsUpdateMutex	541
10.99.4.5 requestSub	541
10.99.4.6 responsePub	541
10.99.4.7 sdf	541
10.99.4.8 world	542
10.100 gazebo::physics::PhysicsFactory Class Reference	542
10.100.1 Detailed Description	542
10.100.2 Member Function Documentation	542
10.100.2.1 NewPhysicsEngine	542
10.100.2.2 RegisterAll	542
10.100.2.3 RegisterPhysicsEngine	542
10.101 gazebo::common::PID Class Reference	543
10.101.1 Detailed Description	544
10.101.2 Constructor & Destructor Documentation	544
10.101.2.1 PID	544
10.101.2.2 ~PID	544
10.101.3 Member Function Documentation	544
10.101.3.1 GetCmd	544
10.101.3.2 GetErrors	544
10.101.3.3 Init	545
10.101.3.4 operator=	545
10.101.3.5 Reset	545
10.101.3.6 SetCmd	545
10.101.3.7 SetCmdMax	545
10.101.3.8 SetCmdMin	546
10.101.3.9 SetDGain	546
10.101.3.10 SetIGain	546
10.101.3.11 SetIMax	546
10.101.3.12 SetIMin	546
10.101.3.13 SetPGain	546
10.101.3.14 Update	547
10.102 gazebo::math::Plane Class Reference	547
10.102.1 Detailed Description	548
10.102.2 Constructor & Destructor Documentation	548
10.102.2.1 Plane	548
10.102.2.2 ~Plane	548

10.102.2.3	Plane	548
10.102.2.4	~Plane	548
10.102.3	Member Function Documentation	548
10.102.3.1	Distance	548
10.102.3.2	operator=	549
10.102.3.3	Set	549
10.102.4	Member Data Documentation	549
10.102.4.1	id	549
10.102.4.2	normal	549
10.102.4.3	size	549
10.103	Gazebo::physics::PlaneShape Class Reference	549
10.103.1	Detailed Description	551
10.103.2	Constructor & Destructor Documentation	551
10.103.2.1	PlaneShape	551
10.103.2.2	~PlaneShape	551
10.103.3	Member Function Documentation	551
10.103.3.1	CreatePlane	551
10.103.3.2	FillMsg	551
10.103.3.3	GetNormal	551
10.103.3.4	GetSize	552
10.103.3.5	Init	552
10.103.3.6	ProcessMsg	552
10.103.3.7	SetAltitude	552
10.103.3.8	SetNormal	552
10.103.3.9	SetSize	552
10.104	df::Plugin Class Reference	553
10.104.1	Constructor & Destructor Documentation	553
10.104.1.1	Plugin	553
10.104.2	Member Function Documentation	553
10.104.2.1	Clear	553
10.104.2.2	Print	553
10.104.3	Member Data Documentation	554
10.104.3.1	data	554
10.104.3.2	filename	554
10.104.3.3	name	554
10.105	Gazebo::PluginT< T > Class Template Reference	554
10.105.1	Detailed Description	555

10.105.2	Member Typedef Documentation	555
10.105.2.1	TPtr	555
10.105.3	Member Function Documentation	555
10.105.3.1	Create	555
10.105.3.2	GetFilename	555
10.105.3.3	GetHandle	555
10.105.3.4	GetType	556
10.105.4	Member Data Documentation	556
10.105.4.1	filename	556
10.105.4.2	handle	556
10.105.4.3	type	556
10.106	Gazebo::math::Pose Class Reference	556
10.106.1	Detailed Description	558
10.106.2	Constructor & Destructor Documentation	558
10.106.2.1	Pose	558
10.106.2.2	Pose	558
10.106.2.3	Pose	558
10.106.2.4	Pose	559
10.106.2.5	~Pose	559
10.106.3	Member Function Documentation	559
10.106.3.1	CoordPoseSolve	559
10.106.3.2	CoordPositionAdd	559
10.106.3.3	CoordPositionAdd	559
10.106.3.4	CoordPositionSub	559
10.106.3.5	CoordRotationAdd	560
10.106.3.6	CoordRotationSub	560
10.106.3.7	Correct	560
10.106.3.8	GetInverse	560
10.106.3.9	IsFinite	561
10.106.3.10	operator!=	561
10.106.3.11	operator*	561
10.106.3.12	operator+	561
10.106.3.13	operator+=	561
10.106.3.14	operator-	562
10.106.3.15	operator-=	562
10.106.3.16	operator==	562
10.106.3.17	Reset	562

10.106.3.1	RotatePositionAboutOrigin	563
10.106.3.1	Round	563
10.106.3.2	Set	563
10.106.3.2	Set	563
10.106.4	Friends And Related Function Documentation	563
10.106.4.1	operator<<	563
10.106.4.2	operator>>	564
10.106.5	Member Data Documentation	564
10.106.5.1	pos	564
10.106.5.2	rot	564
10.106.5.3	Zero	564
10.107	Gazebo::common::PoseAnimation Class Reference	564
10.107.1	Detailed Description	565
10.107.2	Constructor & Destructor Documentation	565
10.107.2.1	PoseAnimation	565
10.107.2.2	~PoseAnimation	566
10.107.3	Member Function Documentation	566
10.107.3.1	BuildInterpolationSplines	566
10.107.3.2	CreateKeyFrame	566
10.107.3.3	GetInterpolatedKeyFrame	566
10.107.3.4	GetInterpolatedKeyFrame	566
10.108	Gazebo::common::PoseKeyFrame Class Reference	567
10.108.1	Detailed Description	567
10.108.2	Constructor & Destructor Documentation	568
10.108.2.1	PoseKeyFrame	568
10.108.2.2	~PoseKeyFrame	568
10.108.3	Member Function Documentation	568
10.108.3.1	GetRotation	568
10.108.3.2	GetTranslation	568
10.108.3.3	SetRotation	568
10.108.3.4	SetTranslation	568
10.108.4	Member Data Documentation	569
10.108.4.1	rotate	569
10.108.4.2	translate	569
10.109	Gazebo::rendering::Projector Class Reference	569
10.109.1	Detailed Description	569
10.109.2	Constructor & Destructor Documentation	570

10.109.2.1	Projector	570
10.109.2.2	~Projector	570
10.109.3	Member Function Documentation	570
10.109.3.1	GetParent	570
10.109.3.2	Load	570
10.109.3.3	Load	570
10.109.3.4	Load	570
10.109.3.5	SetEnabled	571
10.109.3.6	SetTexture	571
10.109.3.7	Toggle	571
10.110	gazebo::transport::Publication Class Reference	571
10.110.1	Detailed Description	572
10.110.2	Constructor & Destructor Documentation	572
10.110.2.1	Publication	572
10.110.2.2	~Publication	573
10.110.3	Member Function Documentation	573
10.110.3.1	AddPublisher	573
10.110.3.2	AddSubscription	573
10.110.3.3	AddSubscription	573
10.110.3.4	AddTransport	573
10.110.3.5	GetCallbackCount	573
10.110.3.6	GetLocallyAdvertised	574
10.110.3.7	GetMsgType	574
10.110.3.8	GetNodeCount	574
10.110.3.9	GetRemoteSubscriptionCount	574
10.110.3.10	GetTransportCount	574
10.110.3.11	HasTransport	574
10.110.3.12	LocalPublish	575
10.110.3.13	Publish	575
10.110.3.14	RemoveSubscription	575
10.110.3.15	RemoveSubscription	575
10.110.3.16	RemoveTransport	575
10.110.3.17	SetLocallyAdvertised	576
10.111	gazebo::transport::PublicationTransport Class Reference	576
10.111.1	Detailed Description	576
10.111.2	Constructor & Destructor Documentation	577
10.111.2.1	PublicationTransport	577

10.111.2.2~PublicationTransport	577
10.111.3 Member Function Documentation	577
10.111.3.1AddCallback	577
10.111.3.2Fini	577
10.111.3.3GetConnection	577
10.111.3.4GetMsgType	577
10.111.3.5GetTopic	578
10.111.3.6Init	578
10.112 gazebo::transport::Publisher Class Reference	578
10.112.1 Detailed Description	579
10.112.2 Constructor & Destructor Documentation	579
10.112.2.1Publisher	579
10.112.2.2~Publisher	579
10.112.3 Member Function Documentation	579
10.112.3.1GetLatching	579
10.112.3.2GetMsgType	579
10.112.3.3GetOutgoingCount	580
10.112.3.4GetPrevMsg	580
10.112.3.5GetTopic	580
10.112.3.6HasConnections	580
10.112.3.7Publish	580
10.112.3.8Publish	580
10.112.3.9SendMessage	581
10.112.3.10SetPublication	581
10.112.3.11WaitForConnection	581
10.113 gazebo::math::Quaternion Class Reference	581
10.113.1 Detailed Description	584
10.113.2 Constructor & Destructor Documentation	584
10.113.2.1Quaternion	584
10.113.2.2Quaternion	584
10.113.2.3Quaternion	584
10.113.2.4Quaternion	585
10.113.2.5Quaternion	585
10.113.2.6Quaternion	585
10.113.2.7~Quaternion	585
10.113.3 Member Function Documentation	585
10.113.3.1Correct	585

10.113.3.2Dot	585
10.113.3.3EulerToQuaternion	586
10.113.3.4EulerToQuaternion	586
10.113.3.5GetAsAxis	586
10.113.3.6GetAsEuler	586
10.113.3.7GetAsMatrix3	586
10.113.3.8GetAsMatrix4	586
10.113.3.9GetExp	587
10.113.3.10GetInverse	587
10.113.3.11GetLog	587
10.113.3.12GetPitch	587
10.113.3.13GetRoll	587
10.113.3.14GetXAxis	587
10.113.3.15GetYaw	588
10.113.3.16GetYAxis	588
10.113.3.17GetZAxis	588
10.113.3.18Invert	588
10.113.3.19Finite	588
10.113.3.20Normalize	588
10.113.3.21operator!=	588
10.113.3.22operator*	589
10.113.3.23operator*	589
10.113.3.24operator*	589
10.113.3.25operator*=	589
10.113.3.26operator+	590
10.113.3.27operator+=	590
10.113.3.28operator-	590
10.113.3.29operator-	590
10.113.3.30operator-=	590
10.113.3.31operator=	591
10.113.3.32operator==	591
10.113.3.33RotateVector	591
10.113.3.34RotateVectorReverse	591
10.113.3.35Round	592
10.113.3.36Scale	592
10.113.3.37Set	592
10.113.3.38SetFromAxis	592

10.113.3.3	SetFromAxis	592
10.113.3.4	SetFromEuler	593
10.113.3.4	SetToIdentity	593
10.113.3.4	Slerp	593
10.113.3.4	Squad	593
10.113.4	Friends And Related Function Documentation	593
10.113.4.1	operator<<	593
10.113.4.2	operator>>	594
10.113.5	Member Data Documentation	594
10.113.5.1	w	594
10.113.5.2	x	594
10.113.5.3	y	594
10.113.5.4	z	594
10.114	gazebo::math::Rand Class Reference	595
10.114.1	Detailed Description	595
10.114.2	Member Function Documentation	595
10.114.2.1	GetDbfNormal	595
10.114.2.2	GetDbfUniform	595
10.114.2.3	GetIntNormal	595
10.114.2.4	GetIntUniform	596
10.114.2.5	GetSeed	596
10.114.2.6	SetSeed	596
10.115	gazebo::sensors::RaySensor Class Reference	596
10.115.1	Detailed Description	598
10.115.2	Constructor & Destructor Documentation	598
10.115.2.1	RaySensor	598
10.115.2.2	~RaySensor	598
10.115.3	Member Function Documentation	599
10.115.3.1	Fini	599
10.115.3.2	GetAngleMax	599
10.115.3.3	GetAngleMin	599
10.115.3.4	GetAngleResolution	599
10.115.3.5	GetFiducial	599
10.115.3.6	GetLaserShape	600
10.115.3.7	GetRange	600
10.115.3.8	GetRangeCount	600
10.115.3.9	GetRangeMax	600

10.115.3.10	GetRangeMin	600
10.115.3.10	GetRangeResolution	601
10.115.3.10	GetRanges	601
10.115.3.10	GetRayCount	601
10.115.3.10	GetRetro	601
10.115.3.10	GetTopic	601
10.115.3.10	GetVerticalAngleMax	602
10.115.3.10	GetVerticalAngleMin	602
10.115.3.10	GetVerticalRangeCount	602
10.115.3.10	GetVerticalRayCount	602
10.115.3.20	Init	602
10.115.3.21	Load	602
10.115.3.21	UpdateImpl	603
10.116	Gazebo::physics::RayShape Class Reference	603
10.116.1	Detailed Description	605
10.116.2	Constructor & Destructor Documentation	605
10.116.2.1	RayShape	605
10.116.2.2	RayShape	606
10.116.2.3	~RayShape	606
10.116.3	Member Function Documentation	606
10.116.3.1	FillMsg	606
10.116.3.2	GetFiducial	606
10.116.3.3	GetGlobalPoints	606
10.116.3.4	GetIntersection	606
10.116.3.5	GetLength	607
10.116.3.6	GetRelativePoints	607
10.116.3.7	GetRetro	607
10.116.3.8	Init	607
10.116.3.9	ProcessMsg	607
10.116.3.10	SetFiducial	608
10.116.3.11	SetLength	608
10.116.3.12	SetPoints	608
10.116.3.13	SetRetro	608
10.116.3.14	Update	608
10.116.4	Member Data Documentation	608
10.116.4.1	contactFiducial	608
10.116.4.2	contactLen	609

10.116.4.3	contactRetro	609
10.116.4.4	globalEndPos	609
10.116.4.5	globalStartPos	609
10.116.4.6	relativeEndPos	609
10.116.4.7	relativeStartPos	609
10.117	gazebo::rendering::RenderEngine Class Reference	609
10.117.1	Detailed Description	611
10.117.2	Member Enumeration Documentation	611
10.117.2.1	RenderPathType	611
10.117.3	Member Function Documentation	611
10.117.3.1	AddResourcePath	611
10.117.3.2	CreateScene	612
10.117.3.3	Finis	612
10.117.3.4	GetRenderPathType	612
10.117.3.5	GetScene	612
10.117.3.6	GetScene	612
10.117.3.7	GetSceneCount	613
10.117.3.8	init	613
10.117.3.9	Load	613
10.117.3.10	RemoveScene	613
10.117.4	Member Data Documentation	613
10.117.4.1	dummyContext	613
10.117.4.2	dummyDisplay	613
10.117.4.3	dummyWindowId	613
10.117.4.4	root	613
10.118	gazebo::sensors::RFIDSensor Class Reference	614
10.118.1	Detailed Description	615
10.118.2	Constructor & Destructor Documentation	615
10.118.2.1	RFIDSensor	615
10.118.2.2	~RFIDSensor	615
10.118.3	Member Function Documentation	615
10.118.3.1	AddTag	615
10.118.3.2	Finis	615
10.118.3.3	init	615
10.118.3.4	Load	615
10.118.3.5	Load	615
10.118.3.6	UpdateImpl	616

10.119	gazebo::sensors::RFIDTag Class Reference	616
10.119.1	Detailed Description	617
10.119.2	Constructor & Destructor Documentation	617
10.119.2.1	RFIDTag	617
10.119.2.2	~RFIDTag	617
10.119.3	Member Function Documentation	617
10.119.3.1	Finis	617
10.119.3.2	GetTagPose	618
10.119.3.3	Init	618
10.119.3.4	Load	618
10.119.3.5	Load	618
10.119.3.6	UpdateImpl	618
10.120	gazebo::rendering::RFIDTagVisual Class Reference	618
10.120.1	Detailed Description	619
10.120.2	Constructor & Destructor Documentation	619
10.120.2.1	RFIDTagVisual	619
10.120.2.2	~RFIDTagVisual	620
10.121	gazebo::rendering::RFIDVisual Class Reference	620
10.121.1	Detailed Description	621
10.121.2	Constructor & Destructor Documentation	621
10.121.2.1	RFIDVisual	621
10.121.2.2	~RFIDVisual	621
10.122	gazebo::physics::Road Class Reference	621
10.122.1	Detailed Description	622
10.122.2	Constructor & Destructor Documentation	622
10.122.2.1	Road	622
10.122.2.2	~Road	623
10.122.3	Member Function Documentation	623
10.122.3.1	Init	623
10.122.3.2	Load	623
10.123	Road Class Reference	623
10.123.1	Detailed Description	623
10.124	gazebo::rendering::Road2d Class Reference	623
10.124.1	Constructor & Destructor Documentation	624
10.124.1.1	Road2d	624
10.124.1.2	~Road2d	624
10.124.2	Member Function Documentation	624

10.124.2.1	Load	624
10.125	<code>gazebo::math::RotationSpline</code> Class Reference	624
10.125.1	Detailed Description	625
10.125.2	Constructor & Destructor Documentation	625
10.125.2.1	<code>RotationSpline</code>	625
10.125.2.2	<code>~RotationSpline</code>	625
10.125.3	Member Function Documentation	625
10.125.3.1	<code>AddPoint</code>	625
10.125.3.2	<code>Clear</code>	626
10.125.3.3	<code>GetNumPoints</code>	626
10.125.3.4	<code>GetPoint</code>	626
10.125.3.5	<code>Interpolate</code>	626
10.125.3.6	<code>Interpolate</code>	627
10.125.3.7	<code>RecalcTangents</code>	627
10.125.3.8	<code>SetAutoCalculate</code>	627
10.125.3.9	<code>UpdatePoint</code>	627
10.125.4	Member Data Documentation	628
10.125.4.1	<code>autoCalc</code>	628
10.125.4.2	<code>points</code>	628
10.125.4.3	<code>tangents</code>	628
10.126	<code>gazebo::rendering::RTShaderSystem</code> Class Reference	628
10.126.1	Detailed Description	629
10.126.2	Member Enumeration Documentation	630
10.126.2.1	<code>LightingModel</code>	630
10.126.3	Member Function Documentation	630
10.126.3.1	<code>AddScene</code>	630
10.126.3.2	<code>ApplyShadows</code>	630
10.126.3.3	<code>AttachEntity</code>	630
10.126.3.4	<code>AttachViewport</code>	630
10.126.3.5	<code>Clear</code>	631
10.126.3.6	<code>DetachEntity</code>	631
10.126.3.7	<code>DetachViewport</code>	631
10.126.3.8	<code>Quit</code>	631
10.126.3.9	<code>GenerateShaders</code>	631
10.126.3.10	<code>Init</code>	631
10.126.3.11	<code>RemoveScene</code>	631
10.126.3.12	<code>RemoveShadows</code>	632

10.126.3.1	SetPerPixelLighting	632
10.126.3.1	UpdateShaders	632
10.127	Gazebo::rendering::Scene Class Reference	632
10.127.1	Detailed Description	635
10.127.2	Constructor & Destructor Documentation	635
10.127.2.1	Scene	635
10.127.2.2	~Scene	635
10.127.3	Member Function Documentation	635
10.127.3.1	AddVisual	636
10.127.3.2	Clear	636
10.127.3.3	CloneVisual	636
10.127.3.4	CreateCamera	636
10.127.3.5	CreateDepthCamera	636
10.127.3.6	CreateGrid	637
10.127.3.7	CreateUserCamera	637
10.127.3.8	DrawLine	637
10.127.3.9	GetAmbientColor	637
10.127.3.10	GetBackgroundColor	638
10.127.3.10	GetCamera	638
10.127.3.10	GetCamera	638
10.127.3.10	GetCameraCount	638
10.127.3.10	GetFirstContact	639
10.127.3.10	GetGrid	639
10.127.3.10	GetGridCount	639
10.127.3.10	GetHeightBelowPoint	639
10.127.3.10	GetHeightmap	639
10.127.3.10	GetId	640
10.127.3.20	GetIdString	640
10.127.3.20	GetLight	640
10.127.3.20	GetLight	640
10.127.3.20	GetLightCount	640
10.127.3.20	GetManager	641
10.127.3.20	SetModelVisualAt	641
10.127.3.20	GetName	641
10.127.3.20	GetSelectedVisual	641
10.127.3.20	GetShadowsEnabled	641
10.127.3.20	GetUserCamera	642

10.127.3.30	GetUserCameraCount	642
10.127.3.31	GetVisual	642
10.127.3.32	GetVisualAt	642
10.127.3.33	GetVisualAt	642
10.127.3.34	GetVisualBelow	643
10.127.3.35	GetVisualsBelowPoint	643
10.127.3.36	GetWorldVisual	643
10.127.3.37	Init	643
10.127.3.38	Load	643
10.127.3.39	Load	644
10.127.3.40	PreRender	644
10.127.3.41	PrintSceneGraph	644
10.127.3.42	RemoveVisual	644
10.127.3.43	SelectVisual	644
10.127.3.44	SetAmbientColor	644
10.127.3.45	SetBackgroundColor	644
10.127.3.46	SetFog	645
10.127.3.47	SetGrid	645
10.127.3.48	SetShadowsEnabled	645
10.127.3.49	SetVisible	645
10.127.3.50	SnapVisualToNearestBelow	645
10.127.3.51	StripSceneName	646
10.127.3.52	ViewContacts	646
10.127.4	Member Data Documentation	646
10.127.4.1	skyx	646
10.128	Gazebo::physics::ScrewJoint< T > Class Template Reference	646
10.128.1	Detailed Description	647
10.128.2	Constructor & Destructor Documentation	647
10.128.2.1	ScrewJoint	647
10.128.2.2	~ScrewJoint	647
10.128.3	Member Function Documentation	648
10.128.3.1	GetAnchor	648
10.128.3.2	GetAngleCount	648
10.128.3.3	Load	648
10.128.3.4	SetAnchor	648
10.128.3.5	SetThreadPitch	648
10.128.4	Member Data Documentation	649

10.128.4.1fakeAnchor	649
10.128.4.2threadPitch	649
10.129df::SDF Class Reference	649
10.129.1Detailed Description	649
10.129.2Constructor & Destructor Documentation	650
10.129.2.1SDF	650
10.129.3Member Function Documentation	650
10.129.3.1PrintDescription	650
10.129.3.2PrintDoc	650
10.129.3.3PrintValues	650
10.129.3.4PrintWiki	650
10.129.3.5SetFromString	650
10.129.3.6ToString	650
10.129.3.7Write	650
10.129.4Member Data Documentation	650
10.129.4.1root	650
10.129.4.2version	650
10.130gazebo::rendering::SelectionObj Class Reference	650
10.130.1Detailed Description	651
10.130.2Constructor & Destructor Documentation	651
10.130.2.1SelectionObj	651
10.130.2.2~SelectionObj	651
10.130.3Member Function Documentation	651
10.130.3.1Attach	651
10.130.3.2Clear	651
10.130.3.3GetVisualName	651
10.130.3.4Init	652
10.130.3.5IsActive	652
10.130.3.6SetActive	652
10.130.3.7SetHighlight	652
10.131gazebo::sensors::Sensor Class Reference	652
10.131.1Detailed Description	655
10.131.2Constructor & Destructor Documentation	655
10.131.2.1Sensor	655
10.131.2.2~Sensor	655
10.131.3Member Function Documentation	655
10.131.3.1FillMsg	655

10.131.3.2	Finis	655
10.131.3.3	GetLastMeasurementTime	655
10.131.3.4	GetLastUpdateTime	656
10.131.3.5	GetName	656
10.131.3.6	GetParentName	656
10.131.3.7	GetPose	656
10.131.3.8	GetScopedName	656
10.131.3.9	GetTopic	656
10.131.3.10	GetType	657
10.131.3.11	GetVisualize	657
10.131.3.12	GetWorldName	657
10.131.3.13	hit	657
10.131.3.14	IsActive	657
10.131.3.15	load	657
10.131.3.16	load	658
10.131.3.17	SetActive	658
10.131.3.18	SetParent	658
10.131.3.19	SetUpdateRate	658
10.131.3.20	update	658
10.131.3.21	updateImpl	659
10.131.4	Member Data Documentation	659
10.131.4.1	active	659
10.131.4.2	connections	659
10.131.4.3	lastMeasurementTime	659
10.131.4.4	lastUpdateTime	659
10.131.4.5	node	659
10.131.4.6	parentName	659
10.131.4.7	plugins	660
10.131.4.8	pose	660
10.131.4.9	poseSub	660
10.131.4.10	stop	660
10.131.4.11	updatePeriod	660
10.131.4.12	world	660
10.132	SensorFactor Class Reference	660
10.132.1	Detailed Description	660
10.133	Gazebo::sensors::SensorFactory Class Reference	660
10.133.1	Member Function Documentation	661

10.133.1.1	GetSensorTypes	661
10.133.1.2	NewSensor	661
10.133.1.3	RegisterAll	661
10.133.1.4	RegisterSensor	662
10.134	gazebo::sensors::SensorManager Class Reference	662
10.134.1	Detailed Description	663
10.134.2	Member Function Documentation	663
10.134.2.1	CreateSensor	663
10.134.2.2	Fin	663
10.134.2.3	GetSensor	663
10.134.2.4	GetSensors	664
10.134.2.5	GetSensorTypes	664
10.134.2.6	Init	664
10.134.2.7	RemoveSensor	664
10.134.2.8	RemoveSensors	664
10.134.2.9	Run	664
10.134.2.10	SensorsInitialized	664
10.134.2.11	Stop	665
10.134.2.12	Update	665
10.135	gazebo::SensorPlugin Class Reference	665
10.135.1	Detailed Description	666
10.135.2	Constructor & Destructor Documentation	666
10.135.2.1	SensorPlugin	666
10.135.2.2	~SensorPlugin	666
10.135.3	Member Function Documentation	666
10.135.3.1	Init	666
10.135.3.2	Load	666
10.135.3.3	Reset	666
10.136	gazebo::Server Class Reference	667
10.136.1	Constructor & Destructor Documentation	667
10.136.1.1	Server	667
10.136.1.2	~Server	667
10.136.2	Member Function Documentation	667
10.136.2.1	Fin	667
10.136.2.2	GetInitialized	667
10.136.2.3	Init	667
10.136.2.4	LoadFile	667

10.136.2.5	LoadString	667
10.136.2.6	ParseArgs	667
10.136.2.7	PrintUsage	667
10.136.2.8	Run	667
10.136.2.9	SetParams	667
10.136.2.10	Stop	668
10.136.3	Member Data Documentation	668
10.136.3.1	systemPluginsArgc	668
10.136.3.2	systemPluginsArgv	668
10.137	gazebo::physics::Shape Class Reference	668
10.137.1	Detailed Description	669
10.137.2	Constructor & Destructor Documentation	669
10.137.2.1	Shape	669
10.137.2.2	~Shape	669
10.137.3	Member Function Documentation	669
10.137.3.1	FillMsg	669
10.137.3.2	FillShapeMsg	670
10.137.3.3	GetInertial	670
10.137.3.4	GetMass	670
10.137.3.5	Init	670
10.137.3.6	ProcessMsg	670
10.137.4	Member Data Documentation	670
10.137.4.1	collisionParent	670
10.138	SingletonT < T > Class Template Reference	671
10.138.1	Detailed Description	672
10.138.2	Constructor & Destructor Documentation	672
10.138.2.1	SingletonT	672
10.138.2.2	~SingletonT	672
10.138.3	Member Function Documentation	672
10.138.3.1	Instance	672
10.139	gazebo::common::Skeleton Class Reference	672
10.139.1	Detailed Description	674
10.139.2	Constructor & Destructor Documentation	674
10.139.2.1	Skeleton	674
10.139.2.2	~Skeleton	674
10.139.2.3	~Skeleton	674
10.139.3	Member Function Documentation	674

10.139.3.1	AddAnimation	674
10.139.3.2	AddVertNodeWeight	675
10.139.3.3	BuildNodeMap	675
10.139.3.4	GetAnimation	675
10.139.3.5	GetBindShapeTransform	675
10.139.3.6	GetNodeByHandle	675
10.139.3.7	GetNodeById	676
10.139.3.8	GetNodeByName	676
10.139.3.9	GetNodes	676
10.139.3.10	GetNumAnimations	676
10.139.3.11	GetNumJoints	676
10.139.3.12	GetNumNodes	676
10.139.3.13	GetNumVertNodeWeights	677
10.139.3.14	GetRootNode	677
10.139.3.15	GetVertNodeWeight	677
10.139.3.16	PrintTransforms	677
10.139.3.17	Scale	677
10.139.3.18	SetBindShapeTransform	677
10.139.3.19	SetNumVertAttached	678
10.139.3.20	SetRootNode	678
10.139.4	Member Data Documentation	678
10.139.4.1	animations	678
10.139.4.2	bindShapeTransform	678
10.139.4.3	nodes	678
10.139.4.4	rawNW	678
10.139.4.5	root	678
10.140	Gazebo::common::SkeletonAnimation Class Reference	679
10.140.1	Detailed Description	680
10.140.2	Constructor & Destructor Documentation	680
10.140.2.1	SkeletonAnimation	680
10.140.2.2	~SkeletonAnimation	680
10.140.3	Member Function Documentation	680
10.140.3.1	AddKeyFrame	680
10.140.3.2	AddKeyFrame	680
10.140.3.3	GetLength	680
10.140.3.4	GetName	681
10.140.3.5	GetNodeCount	681

10.140.3.6	GetNodePoseAt	681
10.140.3.7	GetPoseAt	681
10.140.3.8	GetPoseAtX	682
10.140.3.9	HasNode	682
10.140.3.10	Scale	682
10.140.3.11	SetName	682
10.140.4	Member Data Documentation	682
10.140.4.1	animations	682
10.140.4.2	length	683
10.140.4.3	name	683
10.141	Gazebo::common::SkeletonNode Class Reference	683
10.141.1	Detailed Description	685
10.141.2	Member Enumeration Documentation	685
10.141.2.1	SkeletonNodeType	685
10.141.3	Constructor & Destructor Documentation	685
10.141.3.1	SkeletonNode	685
10.141.3.2	SkeletonNode	686
10.141.3.3	~SkeletonNode	686
10.141.4	Member Function Documentation	686
10.141.4.1	AddChild	686
10.141.4.2	AddRawTransform	686
10.141.4.3	GetChild	686
10.141.4.4	GetChildById	686
10.141.4.5	GetChildByName	687
10.141.4.6	GetChildCount	687
10.141.4.7	GetHandle	687
10.141.4.8	GetId	687
10.141.4.9	GetInverseBindTransform	687
10.141.4.10	GetModelTransform	688
10.141.4.11	GetName	688
10.141.4.12	GetNumRawTrans	688
10.141.4.13	GetParent	688
10.141.4.14	GetRawTransform	688
10.141.4.15	GetRawTransforms	688
10.141.4.16	GetTransform	689
10.141.4.17	GetTransforms	689
10.141.4.18	Joint	689

10.141.4.1	RootNode	689
10.141.4.2	Reset	689
10.141.4.3	SetHandle	689
10.141.4.4	SetId	690
10.141.4.5	SetInitialTransform	690
10.141.4.6	SetInverseBindTransform	690
10.141.4.7	SetModelTransform	690
10.141.4.8	SetName	690
10.141.4.9	SetParent	690
10.141.4.10	SetTransform	691
10.141.4.11	SetType	691
10.141.4.12	UpdateChildrenTransforms	691
10.141.5	Member Data Documentation	691
10.141.5.1	children	691
10.141.5.2	handle	691
10.141.5.3	id	691
10.141.5.4	initialTransform	691
10.141.5.5	invBindTransform	691
10.141.5.6	modelTransform	691
10.141.5.7	name	692
10.141.5.8	parent	692
10.141.5.9	rawTransforms	692
10.141.5.10	transform	692
10.141.5.11	type	692
10.142	Gazebo::physics::SliderJoint< T > Class Template Reference	692
10.142.1	Detailed Description	693
10.142.2	Constructor & Destructor Documentation	693
10.142.2.1	SliderJoint	693
10.142.2.2	~SliderJoint	693
10.142.3	Member Function Documentation	693
10.142.3.1	GetAnchor	693
10.142.3.2	GetAngleCount	694
10.142.3.3	Load	694
10.142.3.4	SetAnchor	694
10.142.4	Member Data Documentation	694
10.142.4.1	fakeAnchor	694
10.143	Gazebo::physics::SphereShape Class Reference	694

10.143.1	Detailed Description	696
10.143.2	Constructor & Destructor Documentation	696
10.143.2.1	SphereShape	696
10.143.2.2	~SphereShape	696
10.143.3	Member Function Documentation	696
10.143.3.1	FillMsg	696
10.143.3.2	GetInertial	696
10.143.3.3	GetMass	696
10.143.3.4	GetRadius	696
10.143.3.5	nit	697
10.143.3.6	ProcessMsg	697
10.143.3.7	SetRadius	697
10.144	gazebo::math::Spline Class Reference	697
10.144.1	Detailed Description	698
10.144.2	Constructor & Destructor Documentation	698
10.144.2.1	Spline	698
10.144.2.2	~Spline	698
10.144.3	Member Function Documentation	699
10.144.3.1	AddPoint	699
10.144.3.2	Clear	699
10.144.3.3	GetPoint	699
10.144.3.4	GetPointCount	699
10.144.3.5	GetTangent	699
10.144.3.6	GetTension	699
10.144.3.7	Interpolate	700
10.144.3.8	Interpolate	700
10.144.3.9	RecalcTangents	700
10.144.3.10	SetAutoCalculate	700
10.144.3.11	SetTension	701
10.144.3.12	UpdatePoint	701
10.144.4	Member Data Documentation	701
10.144.4.1	autoCalc	701
10.144.4.2	coeffs	701
10.144.4.3	points	701
10.144.4.4	tangents	701
10.144.4.5	tension	701
10.145	gazebo::physics::State Class Reference	702

10.145.1	Detailed Description	703
10.145.2	Constructor & Destructor Documentation	703
10.145.2.1	State	703
10.145.2.2	State	703
10.145.2.3	~State	703
10.145.3	Member Function Documentation	703
10.145.3.1	GetName	703
10.145.3.2	GetRealTime	704
10.145.3.3	GetSimTime	704
10.145.3.4	GetWallTime	704
10.145.3.5	Load	704
10.145.3.6	operator-	704
10.145.3.7	operator=	705
10.145.3.8	SetName	705
10.145.4	Member Data Documentation	705
10.145.4.1	name	705
10.145.4.2	realTime	705
10.145.4.3	simTime	705
10.145.4.4	wallTime	705
10.146	Gazebo::common::STLLoader Class Reference	705
10.146.1	Detailed Description	706
10.146.2	Constructor & Destructor Documentation	706
10.146.2.1	STLLoader	706
10.146.2.2	~STLLoader	706
10.146.3	Member Function Documentation	706
10.146.3.1	Load	706
10.147	Gazebo::common::SubMesh Class Reference	707
10.147.1	Detailed Description	709
10.147.2	Member Enumeration Documentation	709
10.147.2.1	PrimitiveType	709
10.147.3	Constructor & Destructor Documentation	709
10.147.3.1	SubMesh	709
10.147.3.2	~SubMesh	709
10.147.4	Member Function Documentation	709
10.147.4.1	AddIndex	709
10.147.4.2	AddNodeAssignment	710
10.147.4.3	AddNormal	710

10.147.4.4	AddNormal	710
10.147.4.5	AddTexCoord	710
10.147.4.6	AddVertex	710
10.147.4.7	AddVertex	711
10.147.4.8	CopyNormals	711
10.147.4.9	CopyVertices	711
10.147.4.10	FillArrays	711
10.147.4.11	GenSphericalTexCoord	711
10.147.4.12	GetIndex	711
10.147.4.13	GetIndexCount	712
10.147.4.14	GetMaterialIndex	712
10.147.4.15	GetMax	712
10.147.4.16	GetMaxIndex	712
10.147.4.17	GetMin	712
10.147.4.18	GetNodeAssignment	712
10.147.4.19	GetNodeAssignmentsCount	712
10.147.4.20	GetNormal	712
10.147.4.21	GetNormalCount	713
10.147.4.22	GetPrimitiveType	713
10.147.4.23	GetTexCoord	713
10.147.4.24	GetTexCoordCount	713
10.147.4.25	GetVertex	713
10.147.4.26	GetVertexCount	714
10.147.4.27	GetVertexIndex	714
10.147.4.28	HasVertex	714
10.147.4.29	RecalculateNormals	714
10.147.4.30	Scale	714
10.147.4.31	SetIndexCount	714
10.147.4.32	SetMaterialIndex	714
10.147.4.33	SetNormal	715
10.147.4.34	SetNormalCount	715
10.147.4.35	SetPrimitiveType	715
10.147.4.36	SetSubMeshCenter	715
10.147.4.37	SetTexCoord	715
10.147.4.38	SetTexCoordCount	716
10.147.4.39	SetVertex	716
10.147.4.40	SetVertexCount	716

10.148	gazebo::transport::SubscribeOptions Class Reference	716
10.148.1	Detailed Description	717
10.148.2	Constructor & Destructor Documentation	717
10.148.2.1	SubscribeOptions	717
10.148.3	Member Function Documentation	717
10.148.3.1	GetLatching	717
10.148.3.2	GetMsgType	717
10.148.3.3	GetNode	717
10.148.3.4	GetTopic	717
10.148.3.5	Init	718
10.149	gazebo::transport::Subscriber Class Reference	718
10.149.1	Detailed Description	718
10.149.2	Constructor & Destructor Documentation	718
10.149.2.1	Subscriber	718
10.149.2.2	~Subscriber	719
10.149.3	Member Function Documentation	719
10.149.3.1	GetTopic	719
10.149.3.2	Unsubscribe	719
10.150	gazebo::transport::SubscriptionTransport Class Reference	719
10.150.1	Detailed Description	720
10.150.2	Constructor & Destructor Documentation	720
10.150.2.1	SubscriptionTransport	720
10.150.2.2	~SubscriptionTransport	720
10.150.3	Member Function Documentation	720
10.150.3.1	GetConnection	720
10.150.3.2	HandleData	720
10.150.3.3	Init	721
10.150.3.4	IsLocal	721
10.151	gazebo::physics::SurfaceParams Class Reference	721
10.151.1	Detailed Description	722
10.151.2	Constructor & Destructor Documentation	722
10.151.2.1	SurfaceParams	722
10.151.2.2	~SurfaceParams	722
10.151.3	Member Function Documentation	723
10.151.3.1	FillMsg	723
10.151.3.2	FillSurfaceMsg	723
10.151.3.3	Load	723

10.151.3.4	ProcessMsg	723
10.151.4	Member Data Documentation	723
10.151.4.1	bounce	723
10.151.4.2	bounceThreshold	723
10.151.4.3	cfm	723
10.151.4.4	erp	724
10.151.4.5	dir1	724
10.151.4.6	kd	724
10.151.4.7	kp	724
10.151.4.8	maxVel	724
10.151.4.9	minDepth	725
10.151.4.10	u1	725
10.151.4.11	ru2	725
10.151.4.12	ip1	725
10.151.4.13	ip2	725
10.152	Gazebo::common::SystemPaths Class Reference	726
10.152.1	Detailed Description	727
10.152.2	Member Function Documentation	727
10.152.2.1	AddGazeboPaths	727
10.152.2.2	AddOgrePaths	727
10.152.2.3	AddPluginPaths	728
10.152.2.4	AddSearchPathSuffix	728
10.152.2.5	ClearGazeboPaths	728
10.152.2.6	ClearOgrePaths	728
10.152.2.7	ClearPluginPaths	728
10.152.2.8	FindFile	728
10.152.2.9	FindFileURI	728
10.152.2.10	GetGazeboPaths	729
10.152.2.11	GetLogPath	729
10.152.2.12	GetModelPaths	729
10.152.2.13	GetOgrePaths	729
10.152.2.14	GetPluginPaths	729
10.152.2.15	GetWorldPathExtension	730
10.152.3	Member Data Documentation	730
10.152.3.1	gazeboPathsFromEnv	730
10.152.3.2	modelPathsFromEnv	730
10.152.3.3	ogrePathsFromEnv	730

10.152.3.4	pluginPathsFromEnv	730
10.153	Gazebo::SystemPlugin Class Reference	730
10.153.1	Detailed Description	731
10.153.2	Constructor & Destructor Documentation	731
10.153.2.1	SystemPlugin	731
10.153.2.2	~SystemPlugin	731
10.153.3	Member Function Documentation	731
10.153.3.1	Init	731
10.153.3.2	Load	731
10.153.3.3	Reset	732
10.154	Gazebo::common::Time Class Reference	732
10.154.1	Detailed Description	735
10.154.2	Constructor & Destructor Documentation	735
10.154.2.1	Time	735
10.154.2.2	Time	736
10.154.2.3	Time	736
10.154.2.4	Time	736
10.154.2.5	Time	736
10.154.2.6	Time	736
10.154.2.7	~Time	736
10.154.3	Member Function Documentation	736
10.154.3.1	Double	737
10.154.3.2	Float	737
10.154.3.3	GetWallTime	737
10.154.3.4	MicToNano	737
10.154.3.5	MilToNano	737
10.154.3.6	MSleep	738
10.154.3.7	NSleep	738
10.154.3.8	NSleep	738
10.154.3.9	operator!=	738
10.154.3.10	operator!=	738
10.154.3.11	operator!=	739
10.154.3.12	operator!=	739
10.154.3.13	operator*	739
10.154.3.14	operator*	739
10.154.3.15	operator*	740
10.154.3.16	operator*=	740

10.154.3.17operator*= 10.154.3.18operator*= 10.154.3.19operator+ 10.154.3.20operator+ 10.154.3.21operator+ 10.154.3.22operator+= 10.154.3.23operator+= 10.154.3.24operator+= 10.154.3.25operator- 10.154.3.26operator- 10.154.3.27operator- 10.154.3.28operator=- 10.154.3.29operator=- 10.154.3.30operator=- 10.154.3.31operator/ 10.154.3.32operator/ 10.154.3.33operator/ 10.154.3.34operator/= 10.154.3.35operator/= 10.154.3.36operator/= 10.154.3.37operator< 10.154.3.38operator< 10.154.3.39operator< 10.154.3.40operator< 10.154.3.41operator<= 10.154.3.42operator<= 10.154.3.43operator<= 10.154.3.44operator<= 10.154.3.45operator= 10.154.3.46operator= 10.154.3.47operator= 10.154.3.48operator== 10.154.3.49operator== 10.154.3.50operator== 10.154.3.51operator== 10.154.3.52operator> 10.154.3.53operator>	740 740 741 741 741 741 742 742 742 742 743 743 743 743 744 744 744 744 745 745 745 745 746 746 746 746 747 747 747 747 748 748 748 748 749 749 749
---	---

10.154.3.54	operator>	749
10.154.3.55	operator>	750
10.154.3.56	operator>=	750
10.154.3.57	operator>=	750
10.154.3.58	operator>=	750
10.154.3.59	operator>=	751
10.154.3.60	SecToNano	751
10.154.3.61	Set	751
10.154.3.62	Set	751
10.154.3.63	SetToWallTime	751
10.154.4	Friends And Related Function Documentation	751
10.154.4.1	operator<<	752
10.154.4.2	operator>>	752
10.154.5	Member Data Documentation	752
10.154.5.1	insec	752
10.154.5.2	sec	752
10.155	gazebo::common::Timer Class Reference	752
10.155.1	Detailed Description	753
10.155.2	Constructor & Destructor Documentation	753
10.155.2.1	Timer	753
10.155.2.2	~Timer	753
10.155.3	Member Function Documentation	754
10.155.3.1	GetElapsed	754
10.155.3.2	Start	754
10.155.4	Friends And Related Function Documentation	754
10.155.4.1	operator<<	754
10.156	gazebo::transport::TopicManager Class Reference	754
10.156.1	Detailed Description	756
10.156.2	Member Typedef Documentation	756
10.156.2.1	SubNodeMap	756
10.156.3	Member Function Documentation	756
10.156.3.1	AddNode	756
10.156.3.2	Advertise	756
10.156.3.3	ClearBuffers	757
10.156.3.4	ConnectPubToSub	757
10.156.3.5	ConnectSubscribers	757
10.156.3.6	ConnectSubToPub	757

10.156.3.7	DisconnectPubFromSub	757
10.156.3.8	DisconnectSubFromPub	757
10.156.3.9	FindPublication	758
10.156.3.10	Ini	758
10.156.3.11	GetTopicNamespaces	758
10.156.3.12	Init	758
10.156.3.13	Advertised	758
10.156.3.14	PauseIncoming	759
10.156.3.15	ProcessNodes	759
10.156.3.16	Publish	759
10.156.3.17	RegisterTopicNamespace	759
10.156.3.18	RemoveNode	759
10.156.3.19	Subscribe	759
10.156.3.20	Unadvertise	760
10.156.3.21	Unsubscribe	760
10.156.3.22	UpdatePublications	760
10.157	Gazebo::physics::TrajectoryInfo Struct Reference	760
10.157	Member Data Documentation	761
10.157.1	duration	761
10.157.1.2	endTime	761
10.157.1.3	id	761
10.157.1.4	startTime	761
10.157.1.5	translated	761
10.157.1.6	type	761
10.158	Gazebo::physics::TrimeshShape Class Reference	761
10.158.1	Detailed Description	763
10.158.2	Constructor & Destructor Documentation	763
10.158.2.1	TrimeshShape	763
10.158.2.2	~TrimeshShape	763
10.158.3	Member Function Documentation	763
10.158.3.1	FillMsg	763
10.158.3.2	GetFilename	763
10.158.3.3	GetMass	764
10.158.3.4	GetSize	764
10.158.3.5	Init	764
10.158.3.6	ProcessMsg	764
10.158.3.7	SetFilename	764

10.158.3.8	SetScale	764
10.158.3.9	Update	764
10.158.4	Member Data Documentation	765
10.158.4.1	mesh	765
10.159	gazebo::physics::UniversalJoint< T > Class Template Reference	765
10.159.1	Detailed Description	766
10.159.2	Constructor & Destructor Documentation	766
10.159.2.1	UniversalJoint	766
10.159.2.2	~UniversalJoint	766
10.159.3	Member Function Documentation	766
10.159.3.1	GetAngleCount	766
10.159.3.2	Load	766
10.160	urdf2gazebo::URDF2Gazebo Class Reference	766
10.160.1	Constructor & Destructor Documentation	769
10.160.1.1	URDF2Gazebo	769
10.160.1.2	~URDF2Gazebo	769
10.160.2	Member Function Documentation	769
10.160.2.1	addKeyValue	769
10.160.2.2	addTransform	769
10.160.2.3	copyPose	769
10.160.2.4	copyPose	770
10.160.2.5	createCollision	770
10.160.2.6	createCollisions	770
10.160.2.7	createGeometry	770
10.160.2.8	createInertial	770
10.160.2.9	createJoint	770
10.160.2.10	createLink	770
10.160.2.11	createSDF	770
10.160.2.12	createVisual	770
10.160.2.13	createVisuals	770
10.160.2.14	getGeometryBoundingBox	771
10.160.2.15	getKeyValueAsString	771
10.160.2.16	loadModelDoc	771
10.160.2.17	loadModelFile	771
10.160.2.18	loadModelString	771
10.160.2.19	loadModelString	771
10.160.2.20	insertGazeboExtensionCollision	771

10.160.2.21	InsertGazeboExtensionJoint	771
10.160.2.22	InsertGazeboExtensionLink	771
10.160.2.23	InsertGazeboExtensionRobot	771
10.160.2.24	InsertGazeboExtensionVisual	771
10.160.2.25	InverseTransformToParentFrame	771
10.160.2.26	IsGazeboExtensions	771
10.160.2.27	IsGazeboExtensions	771
10.160.2.28	ParseGazeboExtension	772
10.160.2.29	ParseVector3	772
10.160.2.30	PrintCollisionGroups	772
10.160.2.31	PrintMass	772
10.160.2.32	PrintMass	772
10.160.2.33	ReduceCollisionsToParent	772
10.160.2.34	ReduceCollisionToParent	772
10.160.2.35	ReduceFixedJoints	772
10.160.2.36	ReduceGazeboExtensionContactSensorFrameReplace	772
10.160.2.37	ReduceGazeboExtensionFrameReplace	772
10.160.2.38	ReduceGazeboExtensionGripperFrameReplace	773
10.160.2.39	ReduceGazeboExtensionJointFrameReplace	773
10.160.2.40	ReduceGazeboExtensionPluginFrameReplace	773
10.160.2.41	ReduceGazeboExtensionProjectorFrameReplace	773
10.160.2.42	ReduceGazeboExtensionProjectorTransformReduction	773
10.160.2.43	ReduceGazeboExtensionSensorTransformReduction	773
10.160.2.44	ReduceGazeboExtensionsTransformReduction	773
10.160.2.45	ReduceGazeboExtensionToParent	773
10.160.2.46	ReduceInertialToParent	773
10.160.2.47	ReduceJointsToParent	773
10.160.2.48	ReduceVisualsToParent	774
10.160.2.49	ReduceVisualToParent	774
10.160.2.50	TransformToParentFrame	774
10.160.2.51	TransformToParentFrame	774
10.160.2.52	TransformToParentFrame	774
10.160.2.53	Values2str	774
10.160.2.54	Vector32str	774
10.160.3	Member Data Documentation	774
10.160.3.1	gazebo_extensions_	774
10.160	gazebo::rendering::UserCamera Class Reference	774

10.161.1	Detailed Description	776
10.161.2	Constructor & Destructor Documentation	777
10.161.2.1	UserCamera	777
10.161.2.2	~UserCamera	777
10.161.3	Member Function Documentation	777
10.161.3.1	AnimationComplete	777
10.161.3.2	AttachToVisualImpl	777
10.161.3.3	EnableViewController	777
10.161.3.4	Finis	778
10.161.3.5	GetAvgFPS	778
10.161.3.6	GetGUIOverlay	778
10.161.3.7	GetTriangleCount	778
10.161.3.8	GetVisual	778
10.161.3.9	GetVisual	778
10.161.3.10	HandleKeyPressEvent	779
10.161.3.11	HandleKeyReleaseEvent	779
10.161.3.12	HandleMouseEvent	779
10.161.3.13	Hit	779
10.161.3.14	Load	779
10.161.3.15	Load	779
10.161.3.16	MoveToPosition	780
10.161.3.17	MoveToVisual	780
10.161.3.18	MoveToVisual	780
10.161.3.19	PostRender	780
10.161.3.20	Resize	780
10.161.3.21	SetFocalPoint	780
10.161.3.22	SetRenderTarget	781
10.161.3.23	SetViewController	781
10.161.3.24	SetViewController	781
10.161.3.25	SetViewportDimensions	781
10.161.3.26	SetWorldPose	781
10.161.3.27	TrackVisualImpl	782
10.161.3.28	Update	782
10.162	gazebo::math::Vector2d Class Reference	782
10.162.1	Detailed Description	784
10.162.2	Constructor & Destructor Documentation	784
10.162.2.1	Vector2d	784

10.162.2.2	Vector2d	784
10.162.2.3	Vector2d	784
10.162.2.4	Vector2d	784
10.162.3	Member Function Documentation	784
10.162.3.1	Cross	784
10.162.3.2	Distance	785
10.162.3.3	IsFinite	785
10.162.3.4	Normalize	785
10.162.3.5	operator!=	785
10.162.3.6	operator*	785
10.162.3.7	operator*	786
10.162.3.8	operator*==	786
10.162.3.9	operator*==	786
10.162.3.10	operator+	786
10.162.3.11	operator+=	787
10.162.3.12	operator-	787
10.162.3.13	operator-=	787
10.162.3.14	operator/	787
10.162.3.15	operator/	788
10.162.3.16	operator/=	788
10.162.3.17	operator/=	788
10.162.3.18	operator=	788
10.162.3.19	operator=	789
10.162.3.20	operator==	789
10.162.3.21	operator[]	789
10.162.3.22	set	789
10.162.4	Friends And Related Function Documentation	790
10.162.4.1	operator<<	790
10.162.4.2	operator>>	790
10.162.5	Member Data Documentation	790
10.162.5.1	x	790
10.162.5.2	y	790
10.163	Gazebo::math::Vector2i Class Reference	790
10.163.1	Detailed Description	792
10.163.2	Constructor & Destructor Documentation	792
10.163.2.1	Vector2i	792
10.163.2.2	Vector2i	792

10.163.2.3	Vector2i	792
10.163.2.4	Vector2i	793
10.163.3	Member Function Documentation	793
10.163.3.1	Cross	793
10.163.3.2	Distance	793
10.163.3.3	IsFinite	793
10.163.3.4	Normalize	793
10.163.3.5	operator!=	793
10.163.3.6	operator*	794
10.163.3.7	operator*	794
10.163.3.8	operator*==	794
10.163.3.9	operator*==	795
10.163.3.10	operator+	795
10.163.3.11	operator+=	795
10.163.3.12	operator-	795
10.163.3.13	operator-=	796
10.163.3.14	operator/	796
10.163.3.15	operator/	796
10.163.3.16	operator/=	797
10.163.3.17	operator/=	797
10.163.3.18	operator=	797
10.163.3.19	operator=	797
10.163.3.20	operator==	798
10.163.3.21	operator[]	798
10.163.3.22	set	798
10.163.4	Friends And Related Function Documentation	798
10.163.4.1	operator<<	798
10.163.4.2	operator>>	799
10.163.5	Member Data Documentation	799
10.163.5.1	x	799
10.163.5.2	y	799
10.164	gazebo::math::Vector3 Class Reference	799
10.164.1	Detailed Description	802
10.164.2	Constructor & Destructor Documentation	802
10.164.2.1	Vector3	802
10.164.2.2	Vector3	802
10.164.2.3	Vector3	802

10.164.2.4	Vector3	802
10.164.3	Member Function Documentation	803
10.164.3.1	Correct	803
10.164.3.2	Cross	803
10.164.3.3	Distance	803
10.164.3.4	Distance	803
10.164.3.5	Dot	803
10.164.3.6	Equal	804
10.164.3.7	GetAbs	804
10.164.3.8	GetDistToLine	804
10.164.3.9	GetLength	804
10.164.3.10	GetMax	804
10.164.3.11	GetMin	805
10.164.3.12	GetNormal	805
10.164.3.13	GetPerpendicular	805
10.164.3.14	GetRounded	805
10.164.3.15	GetSquaredLength	805
10.164.3.16	GetSum	806
10.164.3.17	IsFinite	806
10.164.3.18	Normalize	806
10.164.3.19	operator!=	806
10.164.3.20	operator*	806
10.164.3.21	operator*	806
10.164.3.22	operator*= operator*+=	807
10.164.3.23	operator*= operator*+=	807
10.164.3.24	operator+	807
10.164.3.25	operator+= operator+	808
10.164.3.26	operator- operator-	808
10.164.3.27	operator-= operator-	808
10.164.3.28	operator/ operator/	808
10.164.3.29	operator/ operator/	808
10.164.3.30	operator/=	809
10.164.3.31	operator/=	809
10.164.3.32	operator= operator=	809
10.164.3.33	operator= operator=	809
10.164.3.34	operator==	810
10.164.3.35	operator[]	810

10.164.3.36	Bound	810
10.164.3.37	Bound	810
10.164.3.38	Set	810
10.164.3.39	SetToMax	811
10.164.3.40	SetToMin	811
10.164.4	Friends And Related Function Documentation	811
10.164.4.1	operator<<	811
10.164.4.2	operator>>	811
10.164.5	Member Data Documentation	811
10.164.5.1	x	811
10.164.5.2	y	812
10.164.5.3	z	812
10.164.5.4	Zero	812
10.165	gazebo::math::Vector4 Class Reference	812
10.165.1	Detailed Description	814
10.165.2	Constructor & Destructor Documentation	814
10.165.2.1	Vector4	814
10.165.2.2	Vector4	814
10.165.2.3	Vector4	814
10.165.2.4	~Vector4	815
10.165.3	Member Function Documentation	815
10.165.3.1	Distance	815
10.165.3.2	GetLength	815
10.165.3.3	GetSquaredLength	815
10.165.3.4	IsFinite	815
10.165.3.5	Normalize	815
10.165.3.6	operator!=	815
10.165.3.7	operator*	816
10.165.3.8	operator*	816
10.165.3.9	operator*	816
10.165.3.10	operator*= operator*=	816
10.165.3.11	operator*= operator*=	817
10.165.3.12	operator+	817
10.165.3.13	operator+= operator+=	817
10.165.3.14	operator-	818
10.165.3.15	operator-= operator-=-	818
10.165.3.16	operator/ operator/	818

10.165.3.17	operator/	818
10.165.3.18	operator/=	819
10.165.3.19	operator/=	819
10.165.3.20	operator=	819
10.165.3.21	operator=	820
10.165.3.22	operator==	820
10.165.3.23	operator[]	820
10.165.3.24	set	820
10.165.4	Friends And Related Function Documentation	820
10.165.4.1	operator<<	820
10.165.4.2	operator>>	821
10.165.5	Member Data Documentation	821
10.165.5.1	w	821
10.165.5.2	x	821
10.165.5.3	y	821
10.165.5.4	z	821
10.166	gazebo::common::Video Class Reference	821
10.166.1	Detailed Description	822
10.166.2	Constructor & Destructor Documentation	822
10.166.2.1	Video	822
10.166.2.2	~Video	822
10.166.3	Member Function Documentation	822
10.166.3.1	GetHeight	822
10.166.3.2	GetNextFrame	822
10.166.3.3	GetWidth	823
10.166.3.4	Load	823
10.167	gazebo::rendering::VideoVisual Class Reference	823
10.167.1	Detailed Description	824
10.167.2	Constructor & Destructor Documentation	824
10.167.2.1	VideoVisual	824
10.167.2.2	~VideoVisual	825
10.168	gazebo::rendering::ViewController Class Reference	825
10.168.1	Detailed Description	826
10.168.2	Constructor & Destructor Documentation	826
10.168.2.1	ViewController	826
10.168.2.2	~ViewController	826
10.168.3	Member Function Documentation	826

10.168.3.1	GetTypeString	826
10.168.3.2	HandleKeyPressEvent	826
10.168.3.3	HandleKeyReleaseEvent	827
10.168.3.4	HandleMouseEvent	827
10.168.3.5	Init	827
10.168.3.6	Init	827
10.168.3.7	SetEnabled	827
10.168.3.8	Update	828
10.168.4	Member Data Documentation	828
10.168.4.1	camera	828
10.168.4.2	enabled	828
10.168.4.3	typeString	828
10.169	Gazebo::rendering::Visual Class Reference	828
10.169.1	Detailed Description	833
10.169.2	Constructor & Destructor Documentation	833
10.169.2.1	Visual	833
10.169.2.2	Visual	834
10.169.2.3	~Visual	834
10.169.3	Member Function Documentation	834
10.169.3.1	AttachAxes	834
10.169.3.2	AttachLineVertex	834
10.169.3.3	AttachMesh	834
10.169.3.4	AttachObject	834
10.169.3.5	AttachVisual	835
10.169.3.6	ClearParent	835
10.169.3.7	Clone	835
10.169.3.8	CreateDynamicLine	835
10.169.3.9	DeleteDynamicLine	835
10.169.3.10	DetachObjects	835
10.169.3.11	DetachVisual	836
10.169.3.12	DetachVisual	836
10.169.3.13	DisableTrackVisual	836
10.169.3.14	EnableTrackVisual	836
10.169.3.15	Init	836
10.169.3.16	GetAttachedObjectCount	836
10.169.3.17	GetBoundingBox	836
10.169.3.18	GetChild	837

10.169.3.10	GetChildCount	837
10.169.3.20	GetMaterialName	837
10.169.3.23	GetName	837
10.169.3.22	GetNormalMap	837
10.169.3.23	GetParent	837
10.169.3.24	GetPose	838
10.169.3.25	GetPosition	838
10.169.3.26	GetRootVisual	838
10.169.3.27	GetRotation	838
10.169.3.28	GetScale	838
10.169.3.29	GetScene	838
10.169.3.30	GetSceneNode	839
10.169.3.30	GetShaderType	839
10.169.3.30	GetTransparency	839
10.169.3.30	GetVisibilityFlags	839
10.169.3.30	GetVisible	839
10.169.3.35	SetWorldPose	840
10.169.3.36	HasAttachedObject	840
10.169.3.37	Hit	840
10.169.3.38	InsertMesh	840
10.169.3.39	InsertMesh	840
10.169.3.40	Plane	840
10.169.3.41	IsStatic	840
10.169.3.42	Load	841
10.169.3.43	Load	841
10.169.3.44	LoadFromMsg	841
10.169.3.45	LoadPlugin	841
10.169.3.46	MakeStatic	841
10.169.3.47	MoveToPosition	841
10.169.3.48	MoveToPositions	842
10.169.3.49	RemovePlugin	842
10.169.3.50	SetAmbient	842
10.169.3.53	SetCastShadows	842
10.169.3.52	SetDiffuse	842
10.169.3.53	SetEmissive	842
10.169.3.53	SetHighlighted	843
10.169.3.55	SetMaterial	843

10.169.3.56	SetName	843
10.169.3.57	SetNormalMap	843
10.169.3.58	SetPose	843
10.169.3.59	SetPosition	844
10.169.3.60	SetRibbonTrail	844
10.169.3.61	SetRotation	844
10.169.3.62	SetScale	844
10.169.3.63	SetScene	844
10.169.3.64	SetShaderType	844
10.169.3.65	SetSkeletonPose	845
10.169.3.66	SetSpecular	845
10.169.3.67	SetTransparency	845
10.169.3.68	SetVisibilityFlags	845
10.169.3.69	SetVisible	845
10.169.3.70	SetWorldPose	846
10.169.3.71	SetWorldPosition	846
10.169.3.72	SetWorldRotation	846
10.169.3.73	ShowBoundingBox	846
10.169.3.74	ShowCollision	846
10.169.3.75	ShowCOM	846
10.169.3.76	ShowJoints	847
10.169.3.77	ShowSkeleton	847
10.169.3.78	ToggleVisible	847
10.169.3.79	Update	847
10.169.3.80	UpdateFromMsg	847
10.169.4	Member Data Documentation	847
10.169.4.1	parent	847
10.169.4.2	scene	847
10.169.4.3	sceneNode	847
10.170	gazebo::VisualPlugin Class Reference	848
10.170.1	Detailed Description	848
10.170.2	Constructor & Destructor Documentation	848
10.170.2.1	VisualPlugin	848
10.170.3	Member Function Documentation	848
10.170.3.1	Init	849
10.170.3.2	Load	849
10.170.3.3	Reset	849

10.170	gazebo::rendering::WindowManager Class Reference	849
10.171.1	Detailed Description	850
10.171.2	Member Function Documentation	850
10.171.2.1	CreateWindow	850
10.171.2.2	Finis	850
10.171.2.3	GetAvgFPS	850
10.171.2.4	GetTriangleCount	851
10.171.2.5	GetWindow	851
10.171.2.6	Moved	851
10.171.2.7	Resize	851
10.171.2.8	SetCamera	852
10.172	gazebo::rendering::WireBox Class Reference	852
10.172.1	Detailed Description	852
10.172.2	Constructor & Destructor Documentation	852
10.172.2.1	WireBox	852
10.172.2.2	~WireBox	852
10.172.3	Member Function Documentation	853
10.172.3.1	Init	853
10.172.3.2	SetVisible	853
10.173	gazebo::physics::World Class Reference	853
10.173.1	Detailed Description	856
10.173.2	Constructor & Destructor Documentation	856
10.173.2.1	World	856
10.173.2.2	~World	856
10.173.3	Member Function Documentation	856
10.173.3.1	Clear	856
10.173.3.2	DisableAllModels	856
10.173.3.3	EnableAllModels	856
10.173.3.4	EnablePhysicsEngine	857
10.173.3.5	Finis	857
10.173.3.6	GetByName	857
10.173.3.7	GetEnablePhysicsEngine	857
10.173.3.8	GetEntity	857
10.173.3.9	GetEntityBelowPoint	858
10.173.3.10	GetEntityByName	858
10.173.3.11	GetModel	858
10.173.3.12	GetModel	858

10.173.3.10	GetModelBelowPoint	858
10.173.3.10	GetModelByName	859
10.173.3.10	GetModelCount	859
10.173.3.10	GetModels	859
10.173.3.10	GetName	859
10.173.3.10	GetPauseTime	859
10.173.3.10	GetPhysicsEngine	859
10.173.3.20	GetRealTime	860
10.173.3.20	GetSelectedEntity	860
10.173.3.20	GetSetWorldPoseMutex	860
10.173.3.20	GetSimTime	860
10.173.3.20	GetStartTime	860
10.173.3.25	Hit	860
10.173.3.26	InsertModelFile	861
10.173.3.27	InsertModelSDF	861
10.173.3.26	InsertModelString	861
10.173.3.29	Paused	861
10.173.3.30	Load	861
10.173.3.31	LoadPlugin	862
10.173.3.32	PrintEntityTree	862
10.173.3.33	RemovePlugin	862
10.173.3.34	Reset	862
10.173.3.35	ResetEntities	862
10.173.3.36	ResetTime	862
10.173.3.37	Run	862
10.173.3.38	Save	863
10.173.3.39	SetPaused	863
10.173.3.40	SetSimTime	863
10.173.3.41	SetState	863
10.173.3.42	StepWorld	863
10.173.3.43	Stop	863
10.173.3.43	StripWorldName	864
10.173.3.45	UpdateStateSDF	864
10.173.4	Member Data Documentation	864
10.173.4.1	dirtyPoses	864
10.174	gazebo::WorldPlugin Class Reference	864
10.174.1	Detailed Description	865

10.174.2	Constructor & Destructor Documentation	865
10.174.2.1	WorldPlugin	865
10.174.2.2	~WorldPlugin	865
10.174.3	Member Function Documentation	865
10.174.3.1	Init	865
10.174.3.2	Load	865
10.174.3.3	Reset	865
10.175	Gazebo::physics::WorldState Class Reference	866
10.175.1	Detailed Description	867
10.175.2	Constructor & Destructor Documentation	867
10.175.2.1	WorldState	867
10.175.2.2	WorldState	867
10.175.2.3	WorldState	867
10.175.2.4	~WorldState	868
10.175.3	Member Function Documentation	868
10.175.3.1	GetModelState	868
10.175.3.2	GetModelState	868
10.175.3.3	GetModelStateCount	868
10.175.3.4	GetModelStates	868
10.175.3.5	HasModelState	869
10.175.3.6	IsZero	869
10.175.3.7	Load	869
10.175.3.8	operator+	869
10.175.3.9	operator-	870
10.175.3.10	operator=	870
10.175.4	Friends And Related Function Documentation	870
10.175.4.1	operator<<	870
11	File Documentation	871
11.1	Actor.hh File Reference	871
11.2	Angle.hh File Reference	872
11.2.1	Macro Definition Documentation	874
11.2.1.1	GZ_DTOR	874
11.2.1.2	GZ_NORMALIZE	874
11.2.1.3	GZ_RTOD	874
11.3	Animation.hh File Reference	875
11.4	ArrowVisual.hh File Reference	876

11.5 AxisVisual.hh File Reference	877
11.6 BallJoint.hh File Reference	877
11.7 Base.hh File Reference	878
11.8 Box.hh File Reference	879
11.9 BoxShape.hh File Reference	880
11.10BVHLoader.hh File Reference	882
11.10.1 Macro Definition Documentation	883
11.10.1.1 X_POSITION	883
11.10.1.2 X_ROTATION	883
11.10.1.3 Y_POSITION	883
11.10.1.4 Y_ROTATION	883
11.10.1.5 Z_POSITION	883
11.10.1.6 Z_ROTATION	883
11.11 CallbackHelper.hh File Reference	883
11.12 Camera.hh File Reference	885
11.13 CameraSensor.hh File Reference	886
11.14 CameraVisual.hh File Reference	887
11.15 cegui.h File Reference	887
11.16 ColladaLoader.hh File Reference	888
11.17 Collision.hh File Reference	890
11.18 CollisionState.hh File Reference	891
11.19 Color.hh File Reference	892
11.20 Common.hh File Reference	893
11.21 CommonTypes.hh File Reference	895
11.21.1 Macro Definition Documentation	896
11.21.1.1 GAZEBO_DEPRECATED	896
11.21.1.2 GAZEBO_FORCEINLINE	896
11.21.1.3 NULL	896
11.22 COMVisual.hh File Reference	896
11.23 Connection.hh File Reference	897
11.23.1 Macro Definition Documentation	899
11.23.1.1 HEADER_LENGTH	899
11.24 ConnectionManager.hh File Reference	899
11.25 Console.hh File Reference	900
11.26 Contact.hh File Reference	902
11.26.1 Macro Definition Documentation	903
11.26.1.1 MAX_COLLIDE_RETURNS	903

11.26.1.2 MAX_CONTACT_JOINTS	903
11.27ContactManager.hh File Reference	903
11.28ContactSensor.hh File Reference	904
11.29ContactVisual.hh File Reference	905
11.30Conversions.hh File Reference	906
11.31Converter.hh File Reference	906
11.32CylinderShape.hh File Reference	907
11.33DepthCamera.hh File Reference	908
11.34DepthCameraSensor.hh File Reference	909
11.35Diagnostics.hh File Reference	910
11.36DynamicLines.hh File Reference	912
11.37DynamicRenderable.hh File Reference	912
11.38Entity.hh File Reference	913
11.39Event.hh File Reference	915
11.40Events.hh File Reference	915
11.41Exception.hh File Reference	916
11.42FPSViewController.hh File Reference	918
11.43gazebo.hh File Reference	918
11.44gazebo_core.hh File Reference	920
11.45GazeboGenerator.hh File Reference	920
11.46GpuLaser.hh File Reference	921
11.47GpuRaySensor.hh File Reference	922
11.48Grid.hh File Reference	923
11.49Gripper.hh File Reference	923
11.50GUIOverlay.hh File Reference	925
11.51Heightmap.hh File Reference	925
11.52HeightmapShape.hh File Reference	926
11.53Helpers.hh File Reference	927
11.53.1 Macro Definition Documentation	929
11.53.1.1 GZ_DBL_MAX	929
11.53.1.2 GZ_DBL_MIN	929
11.53.1.3 GZ_FLT_MAX	929
11.53.1.4 GZ_FLT_MIN	929
11.54Hinge2Joint.hh File Reference	929
11.55HingeJoint.hh File Reference	931
11.56Image.hh File Reference	932
11.57ImuSensor.hh File Reference	933

11.58Inertial.hh File Reference	934
11.59IOManager.hh File Reference	935
11.60Joint.hh File Reference	936
11.61JointController.hh File Reference	937
11.62JointState.hh File Reference	939
11.63JointVisual.hh File Reference	940
11.64JointWrench.hh File Reference	940
11.65KeyFrame.hh File Reference	942
11.66LaserVisual.hh File Reference	943
11.67Light.hh File Reference	944
11.68Link.hh File Reference	944
11.69LinkState.hh File Reference	946
11.70LogPlay.hh File Reference	947
11.71LogRecord.hh File Reference	949
11.71.1 Macro Definition Documentation	950
11.71.1.1 GZ_LOG_VERSION	950
11.72mainpage.html File Reference	950
11.73MapShape.hh File Reference	951
11.74Master.hh File Reference	951
11.75Material.hh File Reference	953
11.76Material.hh File Reference	954
11.77MathTypes.hh File Reference	954
11.77.1 Detailed Description	955
11.78Matrix3.hh File Reference	955
11.79Matrix4.hh File Reference	956
11.80Mesh.hh File Reference	957
11.81MeshLoader.hh File Reference	958
11.82MeshManager.hh File Reference	960
11.83Model.hh File Reference	961
11.84ModelDatabase.hh File Reference	963
11.85ModelState.hh File Reference	963
11.86MouseEvent.hh File Reference	965
11.87MovableText.hh File Reference	966
11.88msgs.hh File Reference	967
11.89MultiRayShape.hh File Reference	969
11.90Node.hh File Reference	971
11.91ogre_gazebo.h File Reference	972

11.92OrbitViewController.hh File Reference	973
11.93Param.hh File Reference	973
11.94parser.hh File Reference	975
11.95parser_urdf.hh File Reference	976
11.96Physics.hh File Reference	977
11.97PhysicsEngine.hh File Reference	979
11.98PhysicsFactory.hh File Reference	980
11.99PhysicsTypes.hh File Reference	982
11.99.1 Detailed Description	983
11.99.2 Macro Definition Documentation	984
11.99.2.1 GZ_ALL_COLLIDE	984
11.99.2.2 GZ_FIXED_COLLIDE	984
11.99.2.3 GZ_GHOST_COLLIDE	984
11.99.2.4 GZ_NONE_COLLIDE	984
11.99.2.5 GZ_SENSOR_COLLIDE	984
11.100PID.hh File Reference	984
11.101Plane.hh File Reference	985
11.102PlaneShape.hh File Reference	986
11.103Plugin.hh File Reference	988
11.103.1 Macro Definition Documentation	990
11.103.1.1GZ_REGISTER_MODEL_PLUGIN	990
11.103.1.2GZ_REGISTER_SENSOR_PLUGIN	990
11.103.1.3GZ_REGISTER_SYSTEM_PLUGIN	991
11.103.1.4GZ_REGISTER_VISUAL_PLUGIN	991
11.103.1.5GZ_REGISTER_WORLD_PLUGIN	991
11.104Plugin.hh File Reference	992
11.105Pose.hh File Reference	992
11.106Projector.hh File Reference	993
11.107Publication.hh File Reference	994
11.108PublicationTransport.hh File Reference	995
11.109Publisher.hh File Reference	997
11.110Quaternion.hh File Reference	999
11.111Rand.hh File Reference	1000
11.112RaySensor.hh File Reference	1001
11.113RayShape.hh File Reference	1002
11.114RenderEngine.hh File Reference	1003
11.115RenderEvents.hh File Reference	1004

11.116	Rendering.h File Reference	1005
11.117	Rendering.hh File Reference	1005
11.118	RenderTypes.hh File Reference	1007
11.118.1	Macro Definition Documentation	1008
11.118.1.1	GZ_VISIBILITY_ALL	1008
11.118.1.2	GZ_VISIBILITY_GUI	1008
11.118.1.3	GZ_VISIBILITY_NOT_SELECTABLE	1008
11.118.1.4	GZ_VISIBILITY_SELECTION	1008
11.119	RFIDSensor.hh File Reference	1009
11.120	RFIDTag.hh File Reference	1009
11.121	RFIDTagVisual.hh File Reference	1010
11.122	RFIDVisual.hh File Reference	1011
11.123	Road.hh File Reference	1011
11.124	Road2d.hh File Reference	1013
11.125	RotationSpline.hh File Reference	1013
11.126	RTShaderSystem.hh File Reference	1014
11.127	Scene.hh File Reference	1015
11.128	ScrewJoint.hh File Reference	1016
11.129	sdg.hh File Reference	1017
11.130	SDF.hh File Reference	1018
11.130.1	Macro Definition Documentation	1019
11.130.1.1	SDF_VERSION	1019
11.131	SelectionObj.hh File Reference	1019
11.132	Sensor.hh File Reference	1020
11.133	SensorFactory.hh File Reference	1021
11.134	SensorManager.hh File Reference	1022
11.135	Sensors.hh File Reference	1023
11.136	SensorTypes.hh File Reference	1024
11.136.1	Detailed Description	1026
11.137	Server.hh File Reference	1026
11.138	Shape.hh File Reference	1027
11.139	SingletonT.hh File Reference	1029
11.140	Skeleton.hh File Reference	1029
11.141	SkeletonAnimation.hh File Reference	1031
11.142	SliderJoint.hh File Reference	1032
11.143	SphereShape.hh File Reference	1034
11.144	Spline.hh File Reference	1035

11.145	State.hh File Reference	1036
11.146	STLLoader.hh File Reference	1037
11.146.1	Macro Definition Documentation	1039
11.146.1.1	COR3_MAX	1039
11.146.1.2	FACE_MAX	1039
11.146.1.3	LINE_MAX_LEN	1039
11.146.1.4	ORDER_MAX	1039
11.147	SubscribeOptions.hh File Reference	1039
11.148	Subscriber.hh File Reference	1040
11.149	SubscriptionTransport.hh File Reference	1042
11.150	SurfaceParams.hh File Reference	1044
11.151	SystemPaths.hh File Reference	1045
11.151.1	Macro Definition Documentation	1047
11.151.1.1	GetCurrentDir	1047
11.151.1.2	LINUX	1047
11.152	Time.hh File Reference	1047
11.153	Timer.hh File Reference	1048
11.154	TopicManager.hh File Reference	1049
11.155	Transport.hh File Reference	1051
11.156	TransportTypes.hh File Reference	1053
11.156.1	Detailed Description	1054
11.157	TrimeshShape.hh File Reference	1054
11.158	UniversalJoint.hh File Reference	1055
11.159	UserCamera.hh File Reference	1056
11.160	Vector2d.hh File Reference	1057
11.161	Vector2i.hh File Reference	1058
11.162	Vector3.hh File Reference	1059
11.163	Vector4.hh File Reference	1060
11.164	Video.hh File Reference	1062
11.165	VideoVisual.hh File Reference	1063
11.166	ViewController.hh File Reference	1064
11.167	Visual.hh File Reference	1065
11.168	WindowManager.hh File Reference	1066
11.169	WireBox.hh File Reference	1067
11.170	World.hh File Reference	1068
11.171	WorldState.hh File Reference	1069

Index

1071

Chapter 1

Gazebo API Reference

Gazebo is a multi-robot simulator for both indoor and outdoor environments. It is capable of simulating a population of robots, sensors and objects, but does so in a three-dimensional world. It generates realistic sensor feedback, object collisions and dynamics.

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

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

Wiki: A collection of user supported documentation.

Chapter 2

Todo List

Member gazebo::physics::Joint::GetForce (p. 372) (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. 176) () const

to be implemented

Class gazebo::SystemPlugin (p. 730)

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

Member urdf2gazebo::URDF2Gazebo::parseGazeboExtension (p. 772) (TiXmlDocument &urdf_xml)

: do this using sdf definitions, not hard coded stuff

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Common	27
Events	36
Classes for physics and dynamics	39
Math	46
Messages	52
Rendering	61
Gazebo_parser	65
Sensors	66
Transport	70

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

boost	75
gazebo	
Forward declarations for the common classes	75
gazebo::common	
Common namespace	77
gazebo::event	
Event (p. 277) namespace	80
gazebo::math	
Math namespace	81
gazebo::msgs	
Messages namespace	83
gazebo::physics	
Namespace for physics	85
gazebo::rendering	
Rendering namespace	89
gazebo::sensors	
Sensors namespace	93
gazebo::transport	95
google	97
google::protobuf	97
google::protobuf::compiler	97
google::protobuf::compiler::cpp	98
Ogre	98
ogre	98
sdf	
Namespace for Simulation Description Format parser	98
SkyX	100
urdf2gazebo	
Namespace for URDF to SDF parser	100

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

gazebo::math::Angle	107
gazebo::common::Animation	115
gazebo::common::NumericAnimation	514
gazebo::common::PoseAnimation	564
gazebo::math::Box	136
gazebo::common::BVHLoader	143
gazebo::transport::CallbackHelper	144
gazebo::transport::CallbackHelperT< M >	147
gazebo::transport::DebugCallbackHelper	236
gazebo::transport::SubscriptionTransport	719
CodeGenerator	
google::protobuf::compiler::cpp::GazeboGenerator	312
gazebo::common::Color	193
gazebo::event::Connection	206
gazebo::common::Console	218
gazebo::physics::Contact	218
gazebo::physics::ContactManager	222
gazebo::rendering::Conversions	230
sdf::Converter	232
enable_shared_from_this	
gazebo::physics::Base	125
gazebo::physics::Entity	265
gazebo::physics::Collision	180
gazebo::physics::Link	398
gazebo::physics::Model	460
gazebo::physics::Actor	101
gazebo::physics::Joint	366
gazebo::physics::Road	621
gazebo::physics::Shape	668
gazebo::physics::BoxShape	140
gazebo::physics::CylinderShape	233
gazebo::physics::HeightmapShape	341
gazebo::physics::MultiRayShape	492

gazebo::physics::PlaneShape	549
gazebo::physics::RayShape	603
gazebo::physics::SphereShape	694
gazebo::physics::TrimeshShape	761
gazebo::physics::World	853
gazebo::rendering::Camera	149
gazebo::rendering::DepthCamera	238
gazebo::rendering::GpuLaser	313
gazebo::rendering::UserCamera	774
gazebo::rendering::Scene	632
gazebo::rendering::Visual	828
gazebo::rendering::ArrowVisual	119
gazebo::rendering::AxisVisual	120
gazebo::rendering::CameraVisual	177
gazebo::rendering::COMVisual	204
gazebo::rendering::ContactVisual	228
gazebo::rendering::JointVisual	385
gazebo::rendering::LaserVisual	390
gazebo::rendering::RFIDTagVisual	618
gazebo::rendering::RFIDVisual	620
gazebo::rendering::VideoVisual	823
gazebo::sensors::Sensor	652
gazebo::sensors::CameraSensor	173
gazebo::sensors::ContactSensor	224
gazebo::sensors::DepthCameraSensor	242
gazebo::sensors::GpuRaySensor	316
gazebo::sensors::ImuSensor	354
gazebo::sensors::RaySensor	596
gazebo::sensors::RFIDSensor	614
gazebo::sensors::RFIDTag	616
gazebo::transport::Connection	207
gazebo::transport::Node	499
sdf::Element	258
gazebo::event::Event	277
gazebo::event::EventT< void(>>	294
gazebo::event::EventT< void(bool)>	294
gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>	294
gazebo::event::EventT< void(const std::string &)>	294
gazebo::event::EventT< void(const std::string &, const Contact &)>	294
gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>	294
gazebo::event::EventT< void(std::string)>	294
gazebo::event::EventT< void(std::string, std::string)>	294
gazebo::event::EventT< T >	294
gazebo::rendering::Events	280
gazebo::event::Events	282
gazebo::common::Exception	303
urdf2gazebo::GazeboExtension	307
gazebo::rendering::Grid	330
gazebo::physics::Gripper	334
gazebo::rendering::GUIOverlay	335
gazebo::rendering::Heightmap	339
gazebo::common::Image	349
gazebo::physics::Inertial	357

gazebo::transport::IOManager	364
gazebo::physics::JointController	380
gazebo::physics::JointWrench	387
gazebo::common::KeyFrame	389
gazebo::common::NumericKeyFrame	515
gazebo::common::PoseKeyFrame	567
gazebo::rendering::Light	392
Logplay	423
gazebo::Master	426
gazebo::common::Material	428
gazebo::math::Matrix3	437
gazebo::math::Matrix4	441
gazebo::common::Mesh	448
gazebo::common::MeshLoader	454
gazebo::common::ColladaLoader	179
gazebo::common::STLLoader	705
gazebo::common::MouseEvent	483
MovableObject	
gazebo::rendering::MovableText	486
gazebo::common::NodeAnimation	504
gazebo::common::NodeAssignment	508
gazebo::common::NodeTransform	509
sdf::Param	521
sdf::ParamT< std::string >	527
sdf::ParamT< T >	527
ParamT< T >	531
gazebo::physics::PhysicsEngine	531
gazebo::physics::PhysicsFactory	542
gazebo::common::PID	543
gazebo::math::Plane	547
gazebo::PluginT< T >	554
gazebo::PluginT< ModelPlugin >	554
gazebo::ModelPlugin	475
gazebo::PluginT< SensorPlugin >	554
gazebo::SensorPlugin	665
gazebo::PluginT< SystemPlugin >	554
gazebo::SystemPlugin	730
gazebo::PluginT< VisualPlugin >	554
gazebo::VisualPlugin	848
gazebo::PluginT< WorldPlugin >	554
gazebo::WorldPlugin	864
gazebo::math::Pose	556
gazebo::rendering::Projector	569
gazebo::transport::Publication	571
gazebo::transport::PublicationTransport	576
gazebo::transport::Publisher	578
gazebo::math::Quaternion	581
gazebo::math::Rand	595
Renderable	
gazebo::rendering::MovableText	486
RenderObjectListener	
gazebo::rendering::GpuLaser	313

Road	623
gazebo::rendering::Road2d	623
gazebo::math::RotationSpline	624
sdf::SDF	649
SDFBase	
sdf::Plugin	553
gazebo::rendering::SelectionObj	650
SensorFactor	660
gazebo::sensors::SensorFactory	660
gazebo::Server	667
SimpleRenderable	
gazebo::rendering::DynamicRenderable	254
gazebo::rendering::DynamicLines	250
SingletonT< T >	671
gazebo::common::DiagnosticManager	246
gazebo::common::LogPlay	422
gazebo::common::LogRecord	424
gazebo::common::MeshManager	455
gazebo::common::ModelDatabase	474
gazebo::common::SystemPaths	726
gazebo::rendering::RenderEngine	609
gazebo::rendering::RTShaderSystem	628
gazebo::rendering::WindowManager	849
gazebo::sensors::SensorManager	662
gazebo::transport::ConnectionManager	213
gazebo::transport::TopicManager	754
SingletonT< ConnectionManager >	671
SingletonT< DiagnosticManager >	671
SingletonT< LogPlay >	671
SingletonT< LogRecord >	671
SingletonT< MeshManager >	671
SingletonT< ModelDatabase >	671
SingletonT< RenderEngine >	671
SingletonT< RTShaderSystem >	671
SingletonT< SensorManager >	671
SingletonT< SystemPaths >	671
SingletonT< TopicManager >	671
SingletonT< WindowManager >	671
gazebo::common::Skeleton	672
gazebo::common::SkeletonAnimation	679
gazebo::common::SkeletonNode	683
gazebo::math::Spline	697
gazebo::physics::State	702
gazebo::physics::CollisionState	190
gazebo::physics::JointState	381
gazebo::physics::LinkState	416
gazebo::physics::ModelState	477
gazebo::physics::WorldState	866
gazebo::common::SubMesh	707
gazebo::transport::SubscribeOptions	716
gazebo::transport::Subscriber	718
gazebo::physics::SurfaceParams	721
T	
gazebo::physics::BallJoint< T >	123

gazebo::physics::Hinge2Joint< T >	345
gazebo::physics::HingeJoint< T >	347
gazebo::physics::ScrewJoint< T >	646
gazebo::physics::SliderJoint< T >	692
gazebo::physics::UniversalJoint< T >	765
gazebo::common::Time	732
gazebo::common::Timer	752
gazebo::common::DiagnosticTimer	249
gazebo::physics::TrajectoryInfo	760
urdf2gazebo::URDF2Gazebo	766
gazebo::math::Vector2d	782
gazebo::math::Vector2i	790
gazebo::math::Vector3	799
gazebo::math::Vector4	812
gazebo::common::Video	821
gazebo::rendering::ViewController	825
gazebo::rendering::FPSViewController	305
gazebo::rendering::OrbitViewController	517
gazebo::rendering::WireBox	852

Chapter 6

Class Index

6.1 Class List

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

gazebo::physics::Actor	
Actor (p. 101) class enables GPU based mesh model / skeleton scriptable animation	101
gazebo::math::Angle	
An angle and related functions	107
gazebo::common::Animation	
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes	115
gazebo::rendering::ArrowVisual	
Basic arrow visualization	119
gazebo::rendering::AxisVisual	
Basic axis visualization	120
gazebo::physics::BallJoint< T >	
Base (p. 125) class for a ball joint	123
gazebo::physics::Base	
Base (p. 125) class for most physics classes	125
gazebo::math::Box	
Mathematical representation of a box and related functions	136
gazebo::physics::BoxShape	
Box geometry primitive	140
gazebo::common::BVHLoader	
Handles loading BVH animation files	143
gazebo::transport::CallbackHelper	
A helper class to handle callbacks when messages arrive	144
gazebo::transport::CallbackHelperT< M >	
Callback helper Template	147
gazebo::rendering::Camera	
Basic camera sensor	149
gazebo::sensors::CameraSensor	
Basic camera sensor	173
gazebo::rendering::CameraVisual	
Basic camera visualization	177
gazebo::common::ColladaLoader	
Class used to load Collada mesh files	179

gazebo::physics::Collision	
Base (p. 125) class for all collision entities	180
gazebo::physics::CollisionState	
Store state information of a physics::Collision (p. 180) object	190
gazebo::common::Color	
Defines a color	193
gazebo::rendering::COMVisual	
Basic Center of Mass visualization	204
gazebo::event::Connection	
A class that encapsulates a connection	206
gazebo::transport::Connection	
Single TCP/IP connection manager	207
gazebo::transport::ConnectionManager	
Manager of connections	213
gazebo::common::Console	
Message, error, warning functionality	218
gazebo::physics::Contact	
A contact between two collisions	218
gazebo::physics::ContactManager	
Aggregates all the contact information generated by the collision detection engine	222
gazebo::sensors::ContactSensor	
Contact sensor	224
gazebo::rendering::ContactVisual	
Contact visualization	228
gazebo::rendering::Conversions	
Conversions (p. 230) Conversions.hh (p. 906) rendering/Conversions.hh (p. 906)	230
sdf::Converter	
Convert from one version of SDF (p. 649) to another	232
gazebo::physics::CylinderShape	
Cylinder collision	233
gazebo::transport::DebugCallbackHelper	
CallbackHelper (p. 144) subclass with debug facilities	236
gazebo::rendering::DepthCamera	
Depth camera used to render depth data into an image buffer	238
gazebo::sensors::DepthCameraSensor	242
gazebo::common::DiagnosticManager	
A diagnostic manager class	246
gazebo::common::DiagnosticTimer	
A timer designed for diagnostics	249
gazebo::rendering::DynamicLines	
Class for drawing lines that can change	250
gazebo::rendering::DynamicRenderable	
Abstract base class providing mechanisms for dynamically growing hardware buffers	254
sdf::Element	
SDF (p. 649) Element (p. 258) class	258
gazebo::physics::Entity	
Base (p. 125) class for all physics objects in Gazebo	265
gazebo::event::Event	
Base class for all events	277
gazebo::rendering::Events	
Base class for rendering events	280
gazebo::event::Events	
An Event (p. 277) class to get notifications for simulator events	282

gazebo::event::EventT< T >	
A class for event processing	294
gazebo::common::Exception	
Class for generating exceptions	303
gazebo::rendering::FPSViewController	
First Person Shooter style view controller	305
urdf2gazebo::GazeboExtension	307
google::protobuf::compiler::cpp::GazeboGenerator	
Google protobuf message generator for gazebo::msgs (p. 83)	312
gazebo::rendering::GpuLaser	
GPU based laser distance sensor	313
gazebo::sensors::GpuRaySensor	316
gazebo::rendering::Grid	
Displays a grid of cells, drawn with lines	330
gazebo::physics::Gripper	
A gripper abstraction	334
gazebo::rendering::GUIOverlay	
A class that creates a CEGUI overlay on a render window	335
gazebo::rendering::Heightmap	
Rendering a terrain using heightmap information	339
gazebo::physics::HeightmapShape	
HeightmapShape (p. 341) collision shape builds a heightmap from an image	341
gazebo::physics::Hinge2Joint< T >	
A two axis hinge joint	345
gazebo::physics::HingeJoint< T >	
A single axis hinge joint	347
gazebo::common::Image	
Encapsulates an image	349
gazebo::sensors::ImuSensor	
An IMU sensor	354
gazebo::physics::Inertial	
A class for inertial information about a link	357
gazebo::transport::IOManager	
Manages boost::asio IO	364
gazebo::physics::Joint	
Base (p. 125) class for all joints	366
gazebo::physics::JointController	
A class for manipulating physics::Joint (p. 366)	380
gazebo::physics::JointState	
Keeps track of state of a physics::Joint (p. 366)	381
gazebo::rendering::JointVisual	
Visualization for joints	385
gazebo::physics::JointWrench	
Wrench information from a joint	387
gazebo::common::KeyFrame	
A key frame in an animation	389
gazebo::rendering::LaserVisual	
Visualization for laser data	390
gazebo::rendering::Light	
A light source	392
gazebo::physics::Link	
Link (p. 398) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body	398

gazebo::physics::LinkState	Store state information of a physics::Link (p. 398) object	416
gazebo::common::LogPlay	422
Logplay	Open and playback log files that were recorded using LogRecord	423
gazebo::common::LogRecord	Addtogroup gazebo_common	424
gazebo::Master	A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication	426
gazebo::common::Material	Encapsulates description of a material	428
gazebo::math::Matrix3	A 3x3 matrix class	437
gazebo::math::Matrix4	A 3x3 matrix class	441
gazebo::common::Mesh	A 3D mesh	448
gazebo::common::MeshLoader	Base class for loading meshes	454
gazebo::common::MeshManager	Maintains and manages all meshes	455
gazebo::physics::Model	A model is a collection of links, joints, and plugins	460
gazebo::common::ModelDatabase	Connects to model database, and has utility functions to find models	474
gazebo::ModelPlugin	A plugin with access to physics::Model (p. 460)	475
gazebo::physics::ModelState	Store state information of a physics::Model (p. 460) object	477
gazebo::common::MouseEvent	Generic description of a mouse event	483
gazebo::rendering::MovableText	Movable text	486
gazebo::physics::MultiRayShape	Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner	492
gazebo::transport::Node	A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics	499
gazebo::common::NodeAnimation	Node animation	504
gazebo::common::NodeAssignment	Vertex to node weighted assignement for skeleton animation visualization	508
gazebo::common::NodeTransform	NodeTransform (p. 509) Skeleton.hh (p. 1029) common/common.hh	509
gazebo::common::NumericAnimation	A numeric animation	514
gazebo::common::NumericKeyFrame	A keyframe for a NumericAnimation (p. 514)	515
gazebo::rendering::OrbitViewController	Orbit view controller	517
sdf::Param	A parameter class	521

sdf::ParamT< T >	
Templatized parameter class	527
ParamT< T >	531
gazebo::physics::PhysicsEngine	
Base (p. 125) class for a physics engine	531
gazebo::physics::PhysicsFactory	
The physics factory instantiates different physics engines	542
gazebo::common::PID	
Generic PID (p. 543) controller class	543
gazebo::math::Plane	
A plane and related functions	547
gazebo::physics::PlaneShape	
Collision (p. 180) for an infinite plane	549
sdf::Plugin	553
gazebo::PluginT< T >	
A class which all plugins must inherit from	554
gazebo::math::Pose	
Encapsulates a position and rotation in three space	556
gazebo::common::PoseAnimation	
A pose animation	564
gazebo::common::PoseKeyFrame	
A keyframe for a PoseAnimation (p. 564)	567
gazebo::rendering::Projector	
Projects a material onto surface, light a light projector	569
gazebo::transport::Publication	
A publication for a topic	571
gazebo::transport::PublicationTransport	
Transport/transport.hh	576
gazebo::transport::Publisher	
A publisher of messages on a topic	578
gazebo::math::Quaternion	
A quaternion class	581
gazebo::math::Rand	
Random number generator class	595
gazebo::sensors::RaySensor	
Sensor (p. 652) with one or more rays	596
gazebo::physics::RayShape	
Base (p. 125) class for Ray collision geometry	603
gazebo::rendering::RenderEngine	
Adaptor to Ogre3d	609
gazebo::sensors::RFIDSensor	
Sensor (p. 652) class for RFID type of sensor	614
gazebo::sensors::RFIDTag	
RFIDTag (p. 616) to interact with RFIDTagSensors	616
gazebo::rendering::RFIDTagVisual	
Visualization for RFID tags sensor	618
gazebo::rendering::RFIDVisual	
Visualization for RFID sensor	620
gazebo::physics::Road	
For building a Road (p. 621) from SDF	621
Road	
Used to render a strip of road	623
gazebo::rendering::Road2d	623

gazebo::math::RotationSpline	
Spline (p. 697) for rotations	624
gazebo::rendering::RTShaderSystem	
Implements Ogre (p. 98)'s Run-Time Shader system	628
gazebo::rendering::Scene	
Representation of an entire scene graph	632
gazebo::physics::ScrewJoint< T >	
A screw joint, which has both prismatic and rotational DOFs	646
sdf::SDF	
Base SDF (p. 649) class	649
gazebo::rendering::SelectionObj	
A graphical selection object	650
gazebo::sensors::Sensor	
Base class for sensors	652
SensorFactor	
The sensor factory; the class is just for namespacing purposes	660
gazebo::sensors::SensorFactory	660
gazebo::sensors::SensorManager	
Class to manage and update all sensors	662
gazebo::SensorPlugin	
A plugin with access to physics::Sensor	665
gazebo::Server	667
gazebo::physics::Shape	
Base (p. 125) class for all shapes	668
SingletonT< T >	
Singleton template class	671
gazebo::common::Skeleton	
A skeleton	672
gazebo::common::SkeletonAnimation	
Skeleton (p. 672) animation	679
gazebo::common::SkeletonNode	
A skeleton node	683
gazebo::physics::SliderJoint< T >	
A slider joint	692
gazebo::physics::SphereShape	
Sphere collision shape	694
gazebo::math::Spline	
Splines	697
gazebo::physics::State	
State (p. 702) of an entity	702
gazebo::common::STLLoader	
Class used to load STL mesh files	705
gazebo::common::SubMesh	
A child mesh	707
gazebo::transport::SubscribeOptions	
Options for a subscription	716
gazebo::transport::Subscriber	
A subscriber to a topic	718
gazebo::transport::SubscriptionTransport	
Transport/transport.hh	719
gazebo::physics::SurfaceParams	
SurfaceParams (p. 721) defines various Surface contact parameters	721
gazebo::common::SystemPaths	
Functions to handle getting system paths, keeps track of:	726

gazebo::SystemPlugin	
A plugin loaded within the gzserver on startup	730
gazebo::common::Time	
A Time (p. 732) class, can be used to hold wall- or sim-time	732
gazebo::common::Timer	
A timer class, used to time things in real world walltime	752
gazebo::transport::TopicManager	
Manages topics and their subscriptions	754
gazebo::physics::TrajectoryInfo	760
gazebo::physics::TrimeshShape	
Triangle mesh collision shape	761
gazebo::physics::UniversalJoint< T >	
A universal joint	765
urdf2gazebo::URDF2Gazebo	766
gazebo::rendering::UserCamera	
A camera used for user visualization of a scene	774
gazebo::math::Vector2d	
Generic double x, y vector	782
gazebo::math::Vector2i	
Generic integer x, y vector	790
gazebo::math::Vector3	
Generic vector containing 3 elements	799
gazebo::math::Vector4	
Double Generic x, y, z, w vector	812
gazebo::common::Video	
Handle video encoding and decoding using libavcodec	821
gazebo::rendering::VideoVisual	
A visual element that displays a video as a texture	823
gazebo::rendering::ViewController	
Base class for view controllers	825
gazebo::rendering::Visual	
A renderable object	828
gazebo::VisualPlugin	
A plugin loaded within the gzserver on startup	848
gazebo::rendering::WindowManager	
Class to manage render windows	849
gazebo::rendering::WireBox	
Draws a wireframe box	852
gazebo::physics::World	
The world provides access to all other object within a simulated environment	853
gazebo::WorldPlugin	
A plugin with access to physics::World (p. 853)	864
gazebo::physics::WorldState	
Store state information of a physics::World (p. 853) object	866

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Actor.hh	871
Angle.hh	872
Animation.hh	875
ArrowVisual.hh	876
AxisVisual.hh	877
BallJoint.hh	877
Base.hh	878
Box.hh	879
BoxShape.hh	880
BVHLoader.hh	882
CallbackHelper.hh	883
Camera.hh	885
CameraSensor.hh	886
CameraVisual.hh	887
cegui.h	887
ColladaLoader.hh	888
Collision.hh	890
CollisionState.hh	891
Color.hh	892
Common.hh	893
CommonTypes.hh	895
COMVisual.hh	896
Connection.hh	897
ConnectionManager.hh	899
Console.hh	900
Contact.hh	902
ContactManager.hh	903
ContactSensor.hh	904
ContactVisual.hh	905
Conversions.hh	906
Converter.hh	906
CylinderShape.hh	907
DepthCamera.hh	908
DepthCameraSensor.hh	909

Diagnostics.hh	910
DynamicLines.hh	912
DynamicRenderable.hh	912
Entity.hh	913
Event.hh	915
Events.hh	915
Exception.hh	916
FPSViewController.hh	918
gazebo.hh	918
gazebo_core.hh	920
GazeboGenerator.hh	920
GpuLaser.hh	921
GpuRaySensor.hh	922
Grid.hh	923
Gripper.hh	923
GUIOverlay.hh	925
Heightmap.hh	925
HeightmapShape.hh	926
Helpers.hh	927
Hinge2Joint.hh	929
HingeJoint.hh	931
Image.hh	932
ImuSensor.hh	933
Inertial.hh	934
IOManager.hh	935
Joint.hh	936
JointController.hh	937
JointState.hh	939
JointVisual.hh	940
JointWrench.hh	940
KeyFrame.hh	942
LaserVisual.hh	943
Light.hh	944
Link.hh	944
LinkState.hh	946
LogPlay.hh	947
LogRecord.hh	949
mainpage.html	950
MapShape.hh	951
Master.hh	951
common/Material.hh	953
rendering/Material.hh	954
MathTypes.hh	
Forward declarations for the math classes	954
Matrix3.hh	955
Matrix4.hh	956
Mesh.hh	957
MeshLoader.hh	958
MeshManager.hh	960
Model.hh	961
ModelDatabase.hh	963
ModelState.hh	963
MouseEvent.hh	965
MovableText.hh	966

msgs.hh	967
MultiRayShape.hh	969
Node.hh	971
ogre_gazebo.h	972
OrbitViewController.hh	973
Param.hh	973
parser.hh	975
parser_urdf.hh	976
Physics.hh	977
PhysicsEngine.hh	979
PhysicsFactory.hh	980
PhysicsTypes.hh	
Default namespace for gazebo	982
PID.hh	984
Plane.hh	985
PlaneShape.hh	986
common/Plugin.hh	988
sdf/interface/Plugin.hh	992
Pose.hh	992
Projector.hh	993
Publication.hh	994
PublicationTransport.hh	995
Publisher.hh	997
Quaternion.hh	999
Rand.hh	1000
RaySensor.hh	1001
RayShape.hh	1002
RenderEngine.hh	1003
RenderEvents.hh	1004
rendering.h	1005
Rendering.hh	1005
RenderTypes.hh	1007
RFIDSensor.hh	1009
RFIDTag.hh	1009
RFIDTagVisual.hh	1010
RFIDVisual.hh	1011
Road.hh	1011
Road2d.hh	1013
RotationSpline.hh	1013
RTShaderSystem.hh	1014
Scene.hh	1015
ScrewJoint.hh	1016
sdf.hh	1017
SDF.hh	1018
SelectionObj.hh	1019
Sensor.hh	1020
SensorFactory.hh	1021
SensorManager.hh	1022
Sensors.hh	1023
SensorTypes.hh	
Forward declarations and typedefs for sensors	1024
Server.hh	1026
Shape.hh	1027
SingletonT.hh	1029

Skeleton.hh	1029
SkeletonAnimation.hh	1031
SliderJoint.hh	1032
SphereShape.hh	1034
Spline.hh	1035
State.hh	1036
STLLoader.hh	1037
SubscribeOptions.hh	1039
Subscriber.hh	1040
SubscriptionTransport.hh	1042
SurfaceParams.hh	1044
SystemPaths.hh	1045
Time.hh	1047
Timer.hh	1048
TopicManager.hh	1049
Transport.hh	1051
TransportTypes.hh	
Forward declarations for transport	1053
TrimeshShape.hh	1054
UniversalJoint.hh	1055
UserCamera.hh	1056
Vector2d.hh	1057
Vector2i.hh	1058
Vector3.hh	1059
Vector4.hh	1060
Video.hh	1062
VideoVisual.hh	1063
ViewController.hh	1064
Visual.hh	1065
WindowManager.hh	1066
WireBox.hh	1067
World.hh	1068
WorldState.hh	1069

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::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 diagnostic manager class.
- class **gazebo::common::DiagnosticTimer**
A timer designed for diagnostics.
- class **gazebo::common::Exception**
Class for generating exceptions.
- class **gazebo::common::Image**
Encapsulates an image.
- class **gazebo::common::KeyFrame**

- A key frame in an animation.*
- class **gazebo::common::Material**

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

A 3D mesh.
- class **gazebo::common::MeshLoader**

Base class for loading meshes.
- class **gazebo::common::MeshManager**

Maintains and manages all meshes.
- class **gazebo::common::ModelDatabase**

Connects to model database, and has utility functions to find models.
- class **gazebo::ModelPlugin**

*A plugin with access to **physics::Model** (p. 460).*
- class **gazebo::common::MouseEvent**

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

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

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

NodeTransform (p. 509) **Skeleton.hh** (p. 1029) *common/common.hh*
- class **gazebo::common::NumericAnimation**

A numeric animation.
- class **gazebo::common::NumericKeyFrame**

*A keyframe for a **NumericAnimation** (p. 514).*
- class **gazebo::common::PID**

*Generic **PID** (p. 543) controller class.*
- class **gazebo::PluginT < T >**

A class which all plugins must inherit from.
- class **gazebo::common::PoseAnimation**

A pose animation.
- class **gazebo::common::PoseKeyFrame**

*A keyframe for a **PoseAnimation** (p. 564).*
- class **gazebo::SensorPlugin**

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

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

A skeleton.
- class **gazebo::common::SkeletonAnimation**

Skeleton (p. 672) *animation.*
- class **gazebo::common::SkeletonNode**

A skeleton node.
- class **gazebo::common::STLLoader**

Class used to load STL mesh files.
- class **gazebo::common::SubMesh**

A child mesh.

- class **gazebo::common::SystemPaths**
Functions to handle getting system paths, keeps track of:
- class **gazebo::SystemPlugin**
A plugin loaded within the gzserver on startup.
- class **gazebo::common::Time**
*A **Time** (p. 732) class, can be used to hold wall- or sim-time.*
- class **gazebo::common::Timer**
A timer class, used to time things in real world walltime.
- class **gazebo::common::Video**
Handle video encoding and decoding using libavcodec.
- class **gazebo::VisualPlugin**
A plugin loaded within the gzserver on startup.
- class **gazebo::WorldPlugin**
*A plugin with access to **physics::World** (p. 853).*

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 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. 726)*
- 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. 726)*
- std::string **gazebo::common::find_file_path** (const std::string &_file)
*search for a file in **common::SystemPaths** (p. 726)*
- std::string **gazebo::common::ModelDatabase::GetManifest** (const std::string &_uri)
Return the manifest.xml 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.
- static Console * **gazebo::common::Console::Instance** ()
Return an instance to this class.
- void **gazebo::common::Console::Load** ()
Load the message parameters.
- void **gazebo::common::Console::SetQuiet** (bool _q)
Set quiet output.

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 gzmsg (gazebo::common::Console::Instance()->ColorMsg("Msg", 32))`

Output a message.

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

8.1.2.7 `#define gzthrow(msg)`

Value:

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

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

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

8.1.2.8 `#define gzwarn`

Value:

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

Output a warning message.

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 World plugin.

MODEL_PLUGIN A Model plugin.

SENSOR_PLUGIN A Sensor plugin.

SYSTEM_PLUGIN A System plugin.

VISUAL_PLUGIN A 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. 726)

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

in	<i>_lbl</i>	Text label
in	<i>_file</i>	File containing the error
in	<i>_line</i>	Line containing the error
in	<i>_color</i>	Color (p. 193) 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

in	<code>_lbl</code>	Text label
in	<code>_color</code>	Color (p. 193) 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/manifest.xml`) for all models listed in the `<depend>` block, and downloads the models if necessary.

Parameters

in	<code>_path</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. 726)

Parameters

in	<code>_file</code>	Name of the file to find.
in	<code>_searchLocalPath</code>	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. 726)

Parameters

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

Returns

The path containing the file

8.1.5.8 std::string gazebo::common::ModelDatabase::GetManifest (const std::string & *_uri*)

Return the manifest.xml file as a string.

Returns

the manifest file from the model database.

8.1.5.9 `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

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

Get the name of a model based on a URI.

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

Parameters

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

Returns

the model's name.

8.1.5.11 `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

8.1.5.12 `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. 474) to download the model list.

Returns

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

8.1.5.13 `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. 34)

Parameters

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

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

Returns the the global model database URI.

Returns

the URI.

8.1.5.15 `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.

8.1.5.16 `static Console* gazebo::common::Console::Instance () [static]`

Return an instance to this class.

8.1.5.17 `void gazebo::common::Console::Load ()`

Load the message parameters.

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

Set quiet output.

Parameters

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

8.2 Events

Namespaces

- namespace **gazebo::event**
Event (p. 277) namespace.

Classes

- class **gazebo::event::Connection**
A class that encapsulates a connection.
- class **gazebo::event::Event**
Base class for all events.
- class **gazebo::event::Events**
*An **Event** (p. 277) class to get notifications for simulator events.*
- class **gazebo::event::EventT< T >**
A class for event processing.

Functions

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

8.2.1 Detailed Description

8.2.2 Function Documentation

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

Destructor.

Destructor. Deletes all the associated connections.

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

Connect a callback to this event.

Adds a connection.

Parameters

in	_subscriber	Pointer to a callback function
----	-------------	--------------------------------

Returns

A **Connection** (p.206) object, which will automatically call Disconnect when it goes out of scope

Parameters

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

Referenced by gazebo::event::Events::ConnectAddEntity(), gazebo::physics::Collision::ConnectContact(), gazebo::event::Events::ConnectCreateEntity(), gazebo::rendering::Events::ConnectCreateScene(), gazebo::event::Events::ConnectDeleteEntity(), gazebo::event::Events::ConnectDiagTimerStart(), gazebo::event::Events::ConnectDiagTimerStop(), gazebo::physics::Link::ConnectEnabled(), gazebo::physics::Joint::ConnectJointUpdate(), gazebo::rendering::DepthCamera::ConnectNewDepthFrame(), gazebo::rendering::Camera::ConnectNewImageFrame(), gazebo::rendering::GpuLaser::ConnectNewLaserFrame(), gazebo::physics::MultiRayShape::ConnectNewLaserScans(), gazebo::rendering::DepthCamera::ConnectNewRGBPointCloud(), gazebo::event::Events::ConnectPause(), gazebo::event::Events::ConnectPostRender(), gazebo::event::Events::ConnectPreRender(), gazebo::rendering::Events::ConnectRemoveScene(), gazebo::event::Events::ConnectRender(), gazebo::event::Events::ConnectSetSelectedEntity(), gazebo::event::Events::ConnectStep(), gazebo::event::Events::ConnectStop(), gazebo::transport::Connection::ConnectToShutdown(), gazebo::rendering::Events::ConnectViewContacts(), 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. 277).

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

Disconnect a callback to this event.

Removes a connection.

Parameters

in	_c	The connection to disconnect
in	_c	the connection

Implements **gazebo::event::Event** (p. 279).

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

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

NewImageFrame(), gazebo::rendering::GpuLaser::DisconnectNewLaserFrame(), gazebo::physics::MultiRayShape::DisconnectNewLaserScans(), gazebo::rendering::DepthCamera::DisconnectNewRGBPointCloud(), gazebo::event::Events::DisconnectPause(), gazebo::event::Events::DisconnectPostRender(), gazebo::event::Events::DisconnectPreRender(), gazebo::rendering::Events::DisconnectRemoveScene(), gazebo::event::Events::DisconnectRender(), gazebo::event::Events::DisconnectSetSelectedEntity(), gazebo::transport::Connection::DisconnectShutdown(), gazebo::event::Events::DisconnectStep(), gazebo::event::Events::DisconnectStop(), gazebo::rendering::Events::DisconnectViewContacts(), 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

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

Implements **gazebo::event::Event** (p. 279).

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. 101) class enables GPU based mesh model / skeleton scriptable animation.
- class **gazebo::physics::BallJoint**< T >
Base (p. 125) class for a ball joint.
- class **gazebo::physics::Base**
Base (p. 125) class for most physics classes.
- class **gazebo::physics::BoxShape**
Box geometry primitive.
- class **gazebo::physics::Collision**
Base (p. 125) class for all collision entities.
- class **gazebo::physics::CollisionState**
*Store state information of a **physics::Collision** (p. 180) object.*
- class **gazebo::physics::Contact**
A contact between two collisions.
- class **gazebo::physics::ContactManager**
Aggregates all the contact information generated by the collision detection engine.
- class **gazebo::physics::CylinderShape**
Cylinder collision.
- class **gazebo::physics::Entity**
Base (p. 125) class for all physics objects in Gazebo.
- class **gazebo::physics::Gripper**
A gripper abstraction.
- class **gazebo::physics::HeightmapShape**
HeightmapShape (p. 341) collision shape builds a heightmap from an image.
- class **gazebo::physics::Hinge2Joint**< T >
A two axis hinge joint.
- class **gazebo::physics::HingeJoint**< T >
A single axis hinge joint.
- class **gazebo::physics::Inertial**
A class for inertial information about a link.
- class **gazebo::physics::Joint**
Base (p. 125) class for all joints.
- class **gazebo::physics::JointController**

- A class for manipulating **physics::Joint** (p. 366).
- class **gazebo::physics::JointState**
 keeps track of state of a **physics::Joint** (p. 366)
- class **gazebo::physics::JointWrench**
 Wrench information from a joint.
- class **gazebo::physics::Link**
Link (p. 398) 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. 398) object.
- class **Logplay**
 Open and playback log files that were recorded using **LogRecord**.
- class **gazebo::common::LogPlay**
- class **gazebo::physics::Model**
 A model is a collection of links, joints, and plugins.
- class **gazebo::physics::ModelState**
 Store state information of a **physics::Model** (p. 460) 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. 125) class for a physics engine.
- class **gazebo::physics::PhysicsFactory**
 The physics factory instantiates different physics engines.
- class **gazebo::physics::PlaneShape**
Collision (p. 180) for an infinite plane.
- class **gazebo::physics::RayShape**
Base (p. 125) class for Ray collision geometry.
- class **gazebo::physics::Road**
 for building a **Road** (p. 621) from **SDF**
- class **gazebo::physics::ScrewJoint** < T >
 A screw joint, which has both prismatic and rotational DOFs.
- class **gazebo::physics::Shape**
Base (p. 125) class for all shapes.
- class **gazebo::physics::SliderJoint** < T >
 A slider joint.
- class **gazebo::physics::SphereShape**
 Sphere collision shape.
- class **gazebo::physics::State**
State (p. 702) of an entity.
- class **gazebo::physics::SurfaceParams**
SurfaceParams (p. 721) defines various Surface contact parameters.
- class **gazebo::physics::TrimeshShape**
 Triangle mesh collision shape.
- class **gazebo::physics::UniversalJoint** < T >
 A universal joint.
- class **gazebo::physics::World**
 The world provides access to all other object within a simulated environment.
- class **gazebo::physics::WorldState**
 Store state information of a **physics::World** (p. 853) 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. 71).*
- 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. 730)'s and call **gazebo::transport::init** (p. 72).*
- void **gazebo::physics::load_world** (WorldPtr _world, sdf::ElementPtr _sdf)
*Load world from **sdf::Element** (p. 258) pointer.*
- void **gazebo::physics::load_worlds** (sdf::ElementPtr _sdf)
*load multiple worlds from single **sdf::Element** (p. 258) pointer*
- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)
*Pause world by calling **World::SetPaused** (p. 863).*
- 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. 862) 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. 863) 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. 71).

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

Returns a pointer to a world by name.

Parameters

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

Returns

Pointer to the world.

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

Init world given a pointer to it.

Parameters

in	<code>_world</code>	World (p. 853) 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. 730)'s and call **gazebo::transport::init** (p. 72).

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

Load world from **sdf::Element** (p. 258) 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. 258) 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. 863).

Parameters

in	<code>_world</code>	World (p. 853) 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. 862) given a pointer to it.

Parameters

in	<i>_world</i>	World (p. 853) 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. 863) given a pointer to it.

Parameters

in	<i>_world</i>	World (p. 853) 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 set of classes that encapsulate math related properties and functions.

Files

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

Namespaces

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

Classes

- class **gazebo::math::Angle**
An angle and related functions.
- class **gazebo::math::Box**
Mathematical representation of a box and related functions.
- class **gazebo::math::Matrix3**
A 3x3 matrix class.
- class **gazebo::math::Matrix4**
A 3x3 matrix class.
- class **gazebo::math::Plane**
A plane and related functions.
- class **gazebo::math::Pose**
Encapsulates a position and rotation in three space.
- class **gazebo::math::Quaternion**
A quaternion class.
- class **gazebo::math::Rand**
Random number generator class.
- class **gazebo::math::RotationSpline**
***Spline** (p. 697) 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. 799) 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 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. 83).*

Functions

- `msgs::Vector3d gazebo::msgs::Convert` (const `math::Vector3` &_v)
*Convert a **math::Vector3** (p. 799) to a `msgs::Vector3d`.*
- `msgs::Quaternion gazebo::msgs::Convert` (const `math::Quaternion` &_q)
*Convert a **math::Quaternion** (p. 581) to a `msgs::Quaternion`.*
- `msgs::Pose gazebo::msgs::Convert` (const `math::Pose` &_p)
*Convert a **math::Pose** (p. 556) to a `msgs::Pose`.*
- `msgs::Color gazebo::msgs::Convert` (const `common::Color` &_c)
*Convert a **common::Color** (p. 193) to a `msgs::Color`.*
- `msgs::Time gazebo::msgs::Convert` (const `common::Time` &_t)
*Convert a **common::Time** (p. 732) to a `msgs::Time`.*
- `msgs::PlaneGeom gazebo::msgs::Convert` (const `math::Plane` &_p)
*Convert a **math::Plane** (p. 547) 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. 581).*
- `math::Pose gazebo::msgs::Convert` (const `msgs::Pose` &_p)
*Convert a `msgs::Pose` to a **math::Pose** (p. 556).*
- `common::Color gazebo::msgs::Convert` (const `msgs::Color` &_c)
*Convert a `msgs::Color` to a **common::Color** (p. 193).*
- `common::Time gazebo::msgs::Convert` (const `msgs::Time` &_t)
*Convert a `msgs::Time` to a **common::Time** (p. 732).*
- `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. 349).*
- `void gazebo::msgs::Set (msgs::Image *_msg, const common::Image &_i)`
*Set a msgs::Image from a **common::Image** (p. 349).*
- `void gazebo::msgs::Set (msgs::Vector3d *_pt, const math::Vector3 &_v)`
*Set a msgs::Vector3d from a **math::Vector3** (p. 799).*
- `void gazebo::msgs::Set (msgs::Vector2d *_pt, const math::Vector2d &_v)`
*Set a msgs::Vector2d from a **math::Vector3** (p. 799).*
- `void gazebo::msgs::Set (msgs::Quaternion *_q, const math::Quaternion &_v)`
*Set a msgs::Quaternion from a **math::Quaternion** (p. 581).*
- `void gazebo::msgs::Set (msgs::Pose *_p, const math::Pose &_v)`
*Set a msgs::Pose from a **math::Pose** (p. 556).*
- `void gazebo::msgs::Set (msgs::Color *_c, const common::Color &_v)`
*Set a msgs::Color from a **common::Color** (p. 193).*
- `void gazebo::msgs::Set (msgs::Time *_t, const common::Time &_v)`
*Set a msgs::Time from a **common::Time** (p. 732).*
- `void gazebo::msgs::Set (msgs::PlaneGeom *_p, const math::Plane &_v)`
*Set a msgs::Plane from a **math::Plane** (p. 547).*
- `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 Function Documentation

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

Convert a **math::Vector3** (p. 799) to a `msgs::Vector3d`.

Parameters

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

Returns

A `msgs::Vector3d` object

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

Convert a **math::Quaternion** (p. 581) to a `msgs::Quaternion`.

Parameters

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

Returns

A `msgs::Quaternion` object

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

Convert a **math::Pose** (p. 556) to a `msgs::Pose`.

Parameters

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

Returns

A `msgs::Pose` object

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

Convert a **common::Color** (p. 193) to a `msgs::Color`.

Parameters

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

Returns

A `msgs::Color` object

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

Convert a **common::Time** (p. 732) to a `msgs::Time`.

Parameters

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

Returns

A `msgs::Time` object

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

Convert a **`math::Plane`** (p. 547) to a `msgs::PlaneGeom`.

Parameters

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

Returns

A `msgs::PlaneGeom` object

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

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

Parameters

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

Returns

A **`math::Vector3`** (p. 799) object

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

Convert a `msgs::Quaternion` to a **`math::Quaternion`** (p. 581).

Parameters

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

Returns

A **`math::Quaternion`** (p. 581) object

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

Convert a `msgs::Pose` to a **`math::Pose`** (p. 556).

Parameters

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

Returns

A **math::Pose** (p. 556) object

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

Convert a `msgs::Color` to a **common::Color** (p. 193).

Parameters

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

Returns

A **common::Color** (p. 193) object

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

Convert a `msgs::Time` to a **common::Time** (p. 732).

Parameters

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

Returns

A **common::Time** (p. 732) object

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

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

Parameters

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

Returns

A `common::Plane` object

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

Create a request message.

Parameters

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

Returns

A Request message

8.5.2.14 `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

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

Get the header from a protobuf message.

Parameters

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

Returns

A pointer to the message's header

8.5.2.16 `msgs::GUI gazebo::msgs::GUIFromSDF (sdf::ElementPtr _sdf)`

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

Parameters

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

Returns

The new `msgs::GUI` object

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

Initialize a message.

Parameters

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

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

8.5.2.18 msgs::Light gazebo::msgs::LightFromSDF (sdf::ElementPtr _sdf)

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

Parameters

in	_sdf	The sdf element
----	------	-----------------

Returns

The new msgs::Light object

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

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

Parameters

in	_sdf	The sdf element
----	------	-----------------

Returns

The new msgs::Scene object

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

Convert a msgs::Image to a **common::Image** (p. 349).

Parameters

out	_img	The common::Image (p. 349) container
in	_msg	The Image message to convert

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

Set a msgs::Image from a **common::Image** (p. 349).

Parameters

out	_msg	A msgs::Image pointer
in	_i	A common::Image (p. 349) reference

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

Set a msgs::Vector3d from a **math::Vector3** (p. 799).

Parameters

out	<code>_pt</code>	A <code>msgs::Vector3d</code> pointer
in	<code>_v</code>	A <code>math::Vector3</code> (p. 799) reference

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

Set a `msgs::Vector2d` from a `math::Vector3` (p. 799).

Parameters

out	<code>_pt</code>	A <code>msgs::Vector2d</code> pointer
in	<code>_v</code>	A <code>math::Vector2d</code> (p. 782) reference

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

Set a `msgs::Quaternion` from a `math::Quaternion` (p. 581).

Parameters

out	<code>_q</code>	A <code>msgs::Quaternion</code> pointer
in	<code>_v</code>	A <code>math::Quaternion</code> (p. 581) reference

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

Set a `msgs::Pose` from a `math::Pose` (p. 556).

Parameters

out	<code>_p</code>	A <code>msgs::Pose</code> pointer
in	<code>_v</code>	A <code>math::Pose</code> (p. 556) reference

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

Set a `msgs::Color` from a `common::Color` (p. 193).

Parameters

out	<code>_p</code>	A <code>msgs::Color</code> pointer
in	<code>_v</code>	A <code>common::Color</code> (p. 193) reference

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

Set a `msgs::Time` from a `common::Time` (p. 732).

Parameters

out	<code>_p</code>	A <code>msgs::Time</code> pointer
in	<code>_v</code>	A <code>common::Time</code> (p. 732) reference

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

Set a `msgs::Plane` from a `math::Plane` (p. 547).

Parameters

out	<code>_p</code>	A <code>msgs::Plane</code> pointer
in	<code>_v</code>	A <code>math::Plane</code> (p. 547) reference

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

Time stamp a header.

Parameters

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

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

Set the time in a time message.

Parameters

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

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

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

Parameters

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

Returns

The new `msgs::TrackVisual` object

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

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

Parameters

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

Returns

The new `msgs::Visual` object

8.6 Rendering

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

Namespaces

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

Classes

- class **gazebo::rendering::ArrowVisual**
Basic arrow visualization.
- class **gazebo::rendering::AxisVisual**
Basic axis visualization.
- class **gazebo::rendering::Camera**
Basic camera sensor.
- class **gazebo::rendering::CameraVisual**
Basic camera visualization.
- class **gazebo::rendering::COMVisual**
Basic Center of Mass visualization.
- class **gazebo::rendering::ContactVisual**
Contact visualization.
- class **gazebo::rendering::Conversions**
Conversions (p. 230) Conversions.hh (p. 906) rendering/Conversions.hh (p. 906).
- class **gazebo::rendering::DepthCamera**
Depth camera used to render depth data into an image buffer.
- class **gazebo::rendering::DynamicLines**
Class for drawing lines that can change.
- class **gazebo::rendering::DynamicRenderable**
Abstract base class providing mechanisms for dynamically growing hardware buffers.
- class **gazebo::rendering::Events**
Base class for rendering events.
- class **gazebo::rendering::FPSViewController**
First Person Shooter style view controller.
- class **gazebo::rendering::GpuLaser**
GPU based laser distance sensor.
- class **gazebo::rendering::Grid**
Displays a grid of cells, drawn with lines.
- class **gazebo::rendering::GUIOverlay**
A class that creates a CEGUI overlay on a render window.
- class **gazebo::rendering::Heightmap**
Rendering a terrain using heightmap information.
- class **gazebo::rendering::JointVisual**
Visualization for joints.
- class **gazebo::rendering::LaserVisual**

Visualization for laser data.

- class **gazebo::rendering::Light**
A light source.
- class **gazebo::rendering::MovableText**
Movable text.
- class **gazebo::rendering::OrbitViewController**
Orbit view controller.
- class **gazebo::rendering::Projector**
Projects a material onto surface, light a light projector.
- class **gazebo::rendering::RenderEngine**
Adaptor to Ogre3d.
- class **gazebo::rendering::RFIDTagVisual**
Visualization for RFID tags sensor.
- class **gazebo::rendering::RFIDVisual**
Visualization for RFID sensor.
- class **Road**
Used to render a strip of road.
- class **gazebo::rendering::Road2d**
- class **gazebo::rendering::RTShaderSystem**
*Implements **Ogre** (p. 98)'s Run-Time Shader system.*
- class **gazebo::rendering::Scene**
Representation of an entire scene graph.
- class **gazebo::rendering::SelectionObj**
A graphical selection object.
- class **gazebo::rendering::UserCamera**
A camera used for user visualization of a scene.
- class **gazebo::rendering::VideoVisual**
A visual element that displays a video as a texture.
- class **gazebo::rendering::ViewController**
Base class for view controllers.
- class **gazebo::rendering::Visual**
A renderable object.
- class **gazebo::rendering::WindowManager**
Class to manage render windows.
- class **gazebo::rendering::WireBox**
Draws a wireframe box.

Functions

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

*create **rendering::Scene** (p. 632) 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. 632) 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. 632) by name*

8.6.1 Detailed Description

A 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. 632) 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. 632) 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. 632) 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 **sdf::Element**
SDF (p. 649) Element (p. 258) class.
- class **urdf2gazebo::GazeboExtension**
- class **sdf::SDF**
Base SDF (p. 649) class.
- class **urdf2gazebo::URDF2Gazebo**

Typedefs

- typedef urdf::Collision * **urdf2gazebo::CollisionPtr**
- typedef urdf::Visual * **urdf2gazebo::VisualPtr**

8.7.1 Detailed Description

8.7.2 Typedef Documentation

8.7.2.1 typedef urdf::Collision* urdf2gazebo::CollisionPtr

8.7.2.2 typedef urdf::Visual* urdf2gazebo::VisualPtr

8.8 Sensors

A set of sensor classes, functions, and definitions.

Files

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

Namespaces

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

Classes

- class **gazebo::sensors::CameraSensor**
Basic camera sensor.
- class **gazebo::sensors::ContactSensor**
Contact sensor.
- class **gazebo::sensors::DepthCameraSensor**
- class **gazebo::sensors::GpuRaySensor**
- class **gazebo::sensors::ImuSensor**
An IMU sensor.
- class **gazebo::sensors::RaySensor**
Sensor (p. 652) with one or more rays.
- class **gazebo::sensors::RFIDSensor**
Sensor (p. 652) class for RFID type of sensor.
- class **gazebo::sensors::RFIDTag**
RFIDTag (p. 616) 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=true)`
Run the sensor generation one step.
- `void gazebo::sensors::stop ()`
Stop the sensor generation loop.

8.8.1 Detailed Description

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

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

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

8.8.2 Macro Definition Documentation

8.8.2.1 #define GZ_REGISTER_STATIC_SENSOR(name, classname)

Value:

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

Static sensor registration macro.

Use this macro to register sensors with the server.

Parameters

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

8.8.3 Function Documentation

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

Create a sensor using SDF.

Parameters

<code>in</code>	<code>_elem</code>	The SDF element that describes the sensor.
<code>in</code>	<code>_worldName</code>	Name of the world in which to create the sensor.
<code>in</code>	<code>_parentName</code>	The 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

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

Returns

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

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

initialize the sensor generation loop.

Returns

True if successfully initialized, false if not

8.8.3.5 bool gazebo::sensors::load ()

Load the sensor library.

Returns

True if successfully loaded, false if not.

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

Remove a sensor by name.

Parameters

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

8.8.3.7 bool gazebo::sensors::remove_sensors ()

Remove all sensors.

Returns

True if all successfully removed, false if not

8.8.3.8 void gazebo::sensors::run ()

Run sensor generation continuously. This is a blocking call.

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

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 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::DebugCallbackHelper**
***CallbackHelper** (p. 144) subclass with debug facilities.*
- class **gazebo::transport::IOManager**
Manages boost::asio IO.
- class **gazebo::transport::Node**
A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.
- class **gazebo::transport::Publication**
A publication for a topic.
- class **gazebo::transport::PublicationTransport**
transport/transport.hh
- class **gazebo::transport::Publisher**
A publisher of messages on a topic.
- class **gazebo::transport::SubscribeOptions**
Options for a subscription.
- class **gazebo::transport::Subscriber**
A subscriber to a topic.
- class **gazebo::transport::SubscriptionTransport**
transport/transport.hh
- class **gazebo::transport::TopicManager**
Manages topics and their subscriptions.

Typedefs

- typedef CallbackHelper * **gazebo::transport::CallbackHelperPtr**
*boost shared pointer to **transport::CallbackHelper** (p. 144)*

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.
- 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 msgs::Request &_request)
Send a request and receive 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. 144)

8.9.3 Function Documentation

8.9.3.1 void gazebo::transport::clear_buffers ()

Clear any remaining communication buffers.

8.9.3.2 void gazebo::transport::fini ()

Cleanup the transport component.

8.9.3.3 `bool gazebo::transport::get_master_uri (std::string & _master_host, unsigned int & _master_port)`

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

Parameters

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

Returns

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

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

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

Parameters

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

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

Initialize the transport system.

Parameters

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

Returns

true if initialization succeeded; false otherwise

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

Is the transport system stopped?

Returns

true if the transport system is stopped; false otherwise

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

Pause or unpaue incoming messages.

When paused, messages are queued for later delivery

Parameters

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

8.9.3.8 `msgs::Response gazebo::transport::request (const std::string & _worldName, const msgs::Request & _request)`

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 request itself

Returns

The response to the request. Can be empty.

8.9.3.9 `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

8.9.3.10 `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. 277) 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 ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.
- class **ModelPlugin**
*A plugin with access to **physics::Model** (p. 460).*
- class **PluginT**

A class which all plugins must inherit from.

- class **SensorPlugin**

A plugin with access to `physics::Sensor`.

- class **Server**

- class **SystemPlugin**

A plugin loaded within the gzserver on startup.

- class **VisualPlugin**

A plugin loaded within the gzserver on startup.

- class **WorldPlugin**

A plugin with access to `physics::World` (p. 853).

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 **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 diagnostic manager class.
- class **DiagnosticTimer**
A timer designed for diagnostics.
- class **Exception**
Class for generating exceptions.
- class **Image**
Encapsulates an image.
- class **KeyFrame**
A key frame in an animation.
- class **LogPlay**
- class **LogRecord**
addtogroup gazebo_common
- class **Material**
Encapsulates description of a material.
- class **Mesh**
A 3D mesh.
- 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 assignment for skeleton animation visualization.
- class **NodeTransform**
***NodeTransform** (p. 509) **Skeleton.hh** (p. 1029) common/common.hh*
- class **NumericAnimation**
A numeric animation.
- class **NumericKeyFrame**
*A keyframe for a **NumericAnimation** (p. 514).*
- class **PID**
*Generic **PID** (p. 543) controller class.*
- class **PoseAnimation**
A pose animation.
- class **PoseKeyFrame**
*A keyframe for a **PoseAnimation** (p. 564).*
- class **Skeleton**
A skeleton.
- class **SkeletonAnimation**
***Skeleton** (p. 672) animation.*
- class **SkeletonNode**

A skeleton node.

- class **STLLoader**

Class used to load STL mesh files.

- class **SubMesh**

A child mesh.

- class **SystemPaths**

Functions to handle getting system paths, keeps track of:

- class **Time**

*A **Time** (p. 732) class, can be used to hold wall- or sim-time.*

- class **Timer**

A 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. 726)*
- std::string **find_file** (const std::string &_file, bool _searchLocalPath=true)
*search for file in **common::SystemPaths** (p. 726)*
- std::string **find_file_path** (const std::string &_file)
*search for a file in **common::SystemPaths** (p. 726)*

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::NodeMapIter

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. 277) namespace.

Classes

- class **Connection**
A class that encapsulates a connection.
- class **Event**
Base class for all events.
- class **Events**
*An **Event** (p. 277) class to get notifications for simulator events.*
- class **EventT**
A class for event processing.

Typedefs

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

9.4.1 Detailed Description

Event (p. 277) 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 3x3 matrix class.
- class **Matrix4**
A 3x3 matrix class.
- class **Plane**
A plane and related functions.
- class **Pose**
Encapsulates a position and rotation in three space.
- class **Quaternion**
A quaternion class.
- class **Rand**
Random number generator class.
- class **RotationSpline**
***Spline** (p. 697) for rotations.*
- class **Spline**
Splines.
- class **Vector2d**
Generic double x, y vector.
- class **Vector2i**
Generic integer x, y vector.
- class **Vector3**
*The **Vector3** (p. 799) 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.

Functions

- msgs::Vector3d **Convert** (const math::Vector3 &_v)
*Convert a **math::Vector3** (p. 799) to a msgs::Vector3d.*
- msgs::Quaternion **Convert** (const math::Quaternion &_q)
*Convert a **math::Quaternion** (p. 581) to a msgs::Quaternion.*
- msgs::Pose **Convert** (const math::Pose &_p)
*Convert a **math::Pose** (p. 556) to a msgs::Pose.*
- msgs::Color **Convert** (const common::Color &_c)
*Convert a **common::Color** (p. 193) to a msgs::Color.*
- msgs::Time **Convert** (const common::Time &_t)
*Convert a **common::Time** (p. 732) to a msgs::Time.*
- msgs::PlaneGeom **Convert** (const math::Plane &_p)
*Convert a **math::Plane** (p. 547) 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. 581).*
- **math::Pose Convert** (const msgs::Pose &_p)
*Convert a msgs::Pose to a **math::Pose** (p. 556).*
- **common::Color Convert** (const msgs::Color &_c)
*Convert a msgs::Color to a **common::Color** (p. 193).*
- **common::Time Convert** (const msgs::Time &_t)
*Convert a msgs::Time to a **common::Time** (p. 732).*
- **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. 349).*
- void **Set** (msgs::Image *_msg, const **common::Image** &_i)
*Set a msgs::Image from a **common::Image** (p. 349).*
- void **Set** (msgs::Vector3d *_pt, const **math::Vector3** &_v)
*Set a msgs::Vector3d from a **math::Vector3** (p. 799).*
- void **Set** (msgs::Vector2d *_pt, const **math::Vector2d** &_v)
*Set a msgs::Vector2d from a **math::Vector3** (p. 799).*
- void **Set** (msgs::Quaternion *_q, const **math::Quaternion** &_v)
*Set a msgs::Quaternion from a **math::Quaternion** (p. 581).*
- void **Set** (msgs::Pose *_p, const **math::Pose** &_v)
*Set a msgs::Pose from a **math::Pose** (p. 556).*
- void **Set** (msgs::Color *_c, const **common::Color** &_v)
*Set a msgs::Color from a **common::Color** (p. 193).*
- void **Set** (msgs::Time *_t, const **common::Time** &_v)
*Set a msgs::Time from a **common::Time** (p. 732).*
- void **Set** (msgs::PlaneGeom *_p, const **math::Plane** &_v)
*Set a msgs::Plane from a **math::Plane** (p. 547).*
- 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.

9.7 gazebo::physics Namespace Reference

namespace for physics

Classes

- class **Actor**
Actor (p. 101) class enables GPU based mesh model / skeleton scriptable animation.
- class **BallJoint**
Base (p. 125) class for a ball joint.
- class **Base**
Base (p. 125) class for most physics classes.
- class **BoxShape**
Box geometry primitive.
- class **Collision**
Base (p. 125) class for all collision entities.
- class **CollisionState**
*Store state information of a **physics::Collision** (p. 180) object.*
- class **Contact**
A 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. 125) class for all physics objects in Gazebo.
- class **Gripper**
A gripper abstraction.
- class **HeightmapShape**
HeightmapShape (p. 341) collision shape builds a heightmap from an image.
- class **Hinge2Joint**
A two axis hinge joint.
- class **HingeJoint**
A single axis hinge joint.
- class **Inertial**
A class for inertial information about a link.
- class **Joint**
Base (p. 125) class for all joints.
- class **JointController**
*A class for manipulating **physics::Joint** (p. 366).*
- class **JointState**
*keeps track of state of a **physics::Joint** (p. 366)*

- class **JointWrench**
Wrench information from a joint.
- class **Link**
***Link** (p. 398) 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. 398) object.*
- class **Model**
A model is a collection of links, joints, and plugins.
- class **ModelState**
*Store state information of a **physics::Model** (p. 460) object.*
- class **MultiRayShape**
Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.
- class **PhysicsEngine**
***Base** (p. 125) class for a physics engine.*
- class **PhysicsFactory**
The physics factory instantiates different physics engines.
- class **PlaneShape**
***Collision** (p. 180) for an infinite plane.*
- class **RayShape**
***Base** (p. 125) class for Ray collision geometry.*
- class **Road**
*for building a **Road** (p. 621) from SDF*
- class **ScrewJoint**
A screw joint, which has both prismatic and rotational DOFs.
- class **Shape**
***Base** (p. 125) class for all shapes.*
- class **SliderJoint**
A slider joint.
- class **SphereShape**
Sphere collision shape.
- class **State**
***State** (p. 702) of an entity.*
- class **SurfaceParams**
***SurfaceParams** (p. 721) defines various Surface contact parameters.*
- struct **TrajectoryInfo**
- class **TrimeshShape**
Triangle mesh collision shape.
- class **UniversalJoint**
A universal joint.
- class **World**
The world provides access to all other object within a simulated environment.
- class **WorldState**
*Store state information of a **physics::World** (p. 853) 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 **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. 71).
- **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. 730)'s and call `gazebo::transport::init` (p. 72).
- void **load_world** (**WorldPtr** _world, **sdf::ElementPtr** _sdf)
Load world from `sdf::Element` (p. 258) pointer.
- void **load_worlds** (**sdf::ElementPtr** _sdf)
load multiple worlds from single `sdf::Element` (p. 258) pointer

- void **pause_world** (**WorldPtr** _world, bool _pause)
*Pause world by calling **World::SetPaused** (p. 863).*
- 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. 862) 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. 863) 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 Joint* gazebo::physics::JointPtr`

9.7.2.15 `typedef std::vector<LinkPtr> gazebo::physics::Link_V`

9.7.2.16 `typedef Link* gazebo::physics::LinkPtr`

9.7.2.17 `typedef MeshShape* gazebo::physics::MeshShapePtr`

9.7.2.18 `typedef std::vector<ModelPtr> gazebo::physics::Model_V`

9.7.2.19 `typedef Model* gazebo::physics::ModelPtr`

9.7.2.20 `typedef MultiRayShape* gazebo::physics::MultiRayShapePtr`

9.7.2.21 `typedef PhysicsEngine* gazebo::physics::PhysicsEnginePtr`

9.7.2.22 `typedef RayShape* gazebo::physics::RayShapePtr`

9.7.2.23 `typedef Road* gazebo::physics::RoadPtr`

9.7.2.24 `typedef Shape* gazebo::physics::ShapePtr`

9.7.2.25 `typedef SphereShape* gazebo::physics::SphereShapePtr`

9.7.2.26 `typedef SurfaceParams* gazebo::physics::SurfaceParamsPtr`

9.7.2.27 `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. 230) **Conversions.hh** (p. 906) **rendering/Conversions.hh** (p. 906).
- 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 class that creates a CEGUI overlay on a render window.
- class **Heightmap**
Rendering a terrain using heightmap information.
- class **JointVisual**
Visualization for joints.
- class **LaserVisual**
Visualization for laser data.
- class **Light**
A light source.
- class **MovableText**
Movable text.
- class **OrbitViewController**
Orbit view controller.
- class **Projector**
Projects a material onto surface, light a light projector.
- class **RenderEngine**
Adaptor to Ogre3d.
- class **RFIDTagVisual**
Visualization for RFID tags sensor.
- class **RFIDVisual**
Visualization for RFID sensor.
- class **Road2d**
- class **RTShaderSystem**
*Implements **Ogre** (p. 98)'s Run-Time Shader system.*
- class **Scene**
Representation of an entire scene graph.
- class **SelectionObj**
A graphical selection object.

- class **UserCamera**
A camera used for user visualization of a scene.
- class **VideoVisual**
A visual element that displays a video as a texture.
- class **ViewController**
Base class for view controllers.
- class **Visual**
A renderable object.
- class **WindowManager**
Class to manage render windows.
- class **WireBox**
Draws a wireframe box.

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. 632) by name.*
- bool **fini** ()
teardown rendering engine.

- **rendering::ScenePtr get_scene** (const std::string &_name)
*get pointer to **rendering::Scene** (p. 632) by name.*
- bool **init** ()
init rendering engine.
- bool **load** ()
load rendering engine.
- void **remove_scene** (const std::string &_name)
*remove a **rendering::Scene** (p. 632) 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 list of points, 1 vertex per point.

RENDERING_LINE_LIST A list of lines, 2 vertices per line.

RENDERING_LINE_STRIP A strip of connected lines, 1 vertex per line plus 1 start vertex.

RENDERING_TRIANGLE_LIST A list of triangles, 3 vertices per triangle.

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

RENDERING_TRIANGLE_FAN A fan of triangles, 3 vertices for the first triangle, and 1 per triangle after that.

RENDERING_MESH_RESOURCE N/A.

9.9 gazebo::sensors Namespace Reference

Sensors namespace.

Classes

- class **CameraSensor**
Basic camera sensor.
- class **ContactSensor**
Contact sensor.
- class **DepthCameraSensor**
- class **GpuRaySensor**
- class **ImuSensor**
An IMU sensor.
- class **RaySensor**
Sensor (p. 652) with one or more rays.
- class **RFIDSensor**
Sensor (p. 652) class for RFID type of sensor.
- class **RFIDTag**
RFIDTag (p. 616) 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=true)
 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 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 **DebugCallbackHelper**
***CallbackHelper** (p. 144) subclass with debug facilities.*
- class **IOManager**
Manages boost::asio IO.
- class **Node**
A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.
- class **Publication**
A publication for a topic.
- class **PublicationTransport**
transport/transport.hh
- class **Publisher**
A publisher of messages on a topic.
- class **SubscribeOptions**
Options for a subscription.
- class **Subscriber**
A subscriber to a topic.
- class **SubscriptionTransport**
transport/transport.hh
- class **TopicManager**
Manages topics and their subscriptions.

Typedefs

- typedef **CallbackHelper** * **CallbackHelperPtr**
*boost shared pointer to **transport::CallbackHelper** (p. 144)*
- 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.
- 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 msgs::Request &_request)
Send a request and receive 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. 83).

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. 649) to another.*
- class **Element**
***SDF** (p. 649) **Element** (p. 258) class.*
- class **Param**
A parameter class.
- class **ParamT**
Templatized parameter class.
- class **Plugin**
- class **SDF**
*Base **SDF** (p. 649) 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. 649) 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. 649) values from a file.*
- bool **readString** (const std::string &_xmlString, **SDFPtr** _sdf)
 - Populate the **SDF** (p. 649) 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. 649) 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. 649) values from a file.

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

Populate the **SDF** (p. 649) 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 urdf::Collision * **CollisionPtr**
- typedef urdf::Visual * **VisualPtr**

9.19.1 Detailed Description

namespace for URDF to SDF parser

Chapter 10

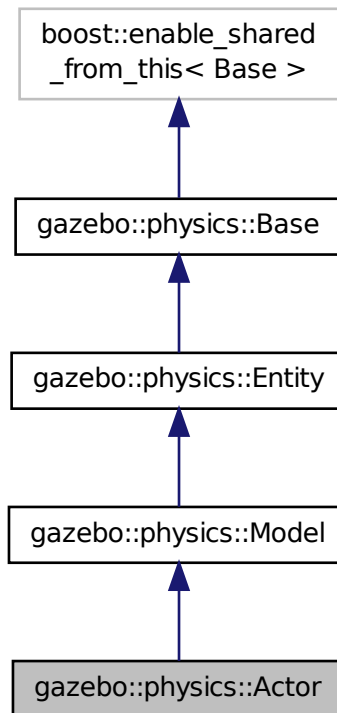
Class Documentation

10.1 gazebo::physics::Actor Class Reference

Actor (p. 101) 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. 125) 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

10.1.1 Detailed Description

Actor (p. 101) class enables GPU based mesh model / skeleton scriptable animation.

10.1.2 Constructor & Destructor Documentation

10.1.2.1 gazebo::physics::Actor::Actor (`BasePtr _parent`) `[explicit]`

Constructor.

Parameters

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

10.1.2.2 `virtual gazebo::physics::Actor::~~Actor () [virtual]`

Destructor.

10.1.3 Member Function Documentation

10.1.3.1 `virtual void gazebo::physics::Actor::Fini () [virtual]`

Finalize the actor.

Reimplemented from **gazebo::physics::Model** (p. 465).

10.1.3.2 `virtual const sdf::ElementPtr gazebo::physics::Actor::GetSDF () [virtual]`

Get the SDF values for the actor.

Returns

Pointer to the SDF values.

Reimplemented from **gazebo::physics::Model** (p. 468).

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

Initialize the actor.

Reimplemented from **gazebo::physics::Model** (p. 469).

10.1.3.4 `virtual bool gazebo::physics::Actor::IsActive () [virtual]`

Returns true when actor is playing animation.

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

Load the actor.

Parameters

in	_sdf	SDF parameters
----	------	----------------

Reimplemented from **gazebo::physics::Entity** (p. 273).

10.1.3.6 `virtual void gazebo::physics::Actor::Play () [virtual]`

Start playing the script.

10.1.3.7 `virtual void gazebo::physics::Actor::Stop () [virtual]`

Stop playing the script.

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

Update the actor.

Reimplemented from **gazebo::physics::Base** (p. 135).

10.1.3.9 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. 474).

10.1.4 Member Data Documentation

10.1.4.1 bool gazebo::physics::Actor::active [protected]

True if the actor is being updated.

10.1.4.2 bool gazebo::physics::Actor::autoStart [protected]

True if the actor should start running automatically.

10.1.4.3 transport::PublisherPtr gazebo::physics::Actor::bonePosePub [protected]

Where to send bone info.

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

True to interpolate along x direction.

10.1.4.5 math::Vector3 gazebo::physics::Actor::lastPos [protected]

Last position of the actor.

10.1.4.6 double gazebo::physics::Actor::lastScriptTime [protected]

Time the script was last updated.

10.1.4.7 unsigned int gazebo::physics::Actor::lastTraj [protected]

The last trajectory.

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

True if the animation should loop.

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

Base (p. 125) link.

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

Pointer to the actor's mesh.

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

The old action.

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

Length of the actor's path.

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

Time when the animation was started.

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

Time of the previous frame.

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

Time length of a script.

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

Skeleton animations.

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

The actor's skeleton.

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

Skeleton to node map.

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

Filename for the skin.

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

Scaling factor to apply to the skin.

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

Amount of time to delay start by.

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

All the trajectories.

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

Trajectory information.

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

Name of the visual.

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

- **Actor.hh**

10.2 gazebo::math::Angle Class Reference

An angle and related functions.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Angle** ()
Constructor.
- **Angle** (double *_radian*)
Copy Constructor.
- **Angle** (const **Angle** &*_angle*)
Copy constructor.
- virtual **~Angle** ()
Destructor.
- double **Degree** () const
Get the angle in degrees.
- double **GetAsDegree** () const *__attribute__((deprecated))*

- Get the angle in degrees.*

 - double **GetAsRadian** () const __attribute__((deprecated))
- Get the angle in radians.*

 - void **Normalize** ()
- Normalize the angle in the range -Pi to Pi.*

 - bool **operator!=** (const **Angle** &_angle) const
- Inequality.*

 - double **operator*** () const
- Dereference operator.*

 - **Angle operator*** (const **Angle** &_angle) const
- Multiplication operator, result = this * _angle.*

 - **Angle operator*=** (const **Angle** &_angle)
- Multiplication set, this = this * _angle.*

 - **Angle operator+** (const **Angle** &_angle) const
- Addition operator, result = this + _angle.*

 - **Angle operator+=** (const **Angle** &_angle)
- Addition set, this = this + _angle.*

 - **Angle operator-** (const **Angle** &_angle) const
- Substraction, result = this - _angle.*

 - **Angle operator-=** (const **Angle** &_angle)
- Subtraction set, this = this - _angle.*

 - **Angle operator/** (const **Angle** &_angle) const
- Division, result = this / _angle.*

 - **Angle operator/=** (const **Angle** &_angle)
- Division set, this = this / _angle.*

 - bool **operator<** (const **Angle** &_angle) const
- Less than operator.*

 - bool **operator<=** (const **Angle** &_angle) const
- Less or equal operator.*

 - bool **operator==** (const **Angle** &_angle) const
- Equality operator, result = this == _angle.*

 - bool **operator>** (const **Angle** &_angle) const
- Greater than operator.*

 - bool **operator>=** (const **Angle** &_angle) const
- Greater or equal operator.*

 - double **Radian** () const
- Get the angle in radians.*

 - void **SetFromDegree** (double _degree)
- Set the value from an angle in degrees.*

 - void **SetFromRadian** (double _radian)
- Set the value from an angle in radians.*

Friends

- std::ostream & **operator<<** (std::ostream &_out, const **gazebo::math::Angle** &_a)

Stream insertion operator.
- std::istream & **operator>>** (std::istream &_in, **gazebo::math::Angle** &_a)

Stream extraction operator.

10.2.1 Detailed Description

An angle and related functions.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 gazebo::math::Angle::Angle ()

Constructor.

10.2.2.2 gazebo::math::Angle::Angle (double *_radian*)

Copy Constructor.

Parameters

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

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

Copy constructor.

Parameters

<i>in</i>	<i>_angle</i>	Angle (p. 107) to copy
-----------	---------------	-------------------------------

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

Destructor.

10.2.3 Member Function Documentation

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

Get the angle in degrees.

Returns

double containing the angle's degree value

10.2.3.2 double gazebo::math::Angle::GetAsDegree () const

Get the angle in degrees.

Returns

double containing the angle's degree value

10.2.3.3 `double gazebo::math::Angle::GetAsRadian () const`

Get the angle in radians.

Returns

double containing the angle's radian value

10.2.3.4 `void gazebo::math::Angle::Normalize ()`

Normalize the angle in the range -Pi to Pi.

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

Inequality.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 107) to check for inequality
-----------------	---------------------	---

Returns

true if this != `_angle`

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

Dereference operator.

Returns

Double containing the angle's radian value

10.2.3.7 `Angle gazebo::math::Angle::operator*(const Angle & _angle) const`

Multiplication operator, result = this * `_angle`.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 107) for multiplication
-----------------	---------------------	--

Returns

the new angle

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

Multiplication set, this = this * `_angle`.

Parameters

in	<i>_angle</i>	Angle (p. 107) for multiplication
----	---------------	--

Returns

angle

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

Addition operator, result = this + *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 107) for addition
----	---------------	------------------------------------

Returns

the new angle

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

Addition set, this = this + *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 107) for addition
----	---------------	------------------------------------

Returns

angle

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

Substraction, result = this - *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 107) for subtraction
----	---------------	---------------------------------------

Returns

the new angle

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

Subtraction set, this = this - *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 107) for subtraction
----	---------------	---------------------------------------

Returns

angle

10.2.3.13 `Angle gazebo::math::Angle::operator/ (const Angle & _angle) const`

Division, result = this / *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 107) for division
----	---------------	------------------------------------

Returns

the new angle

10.2.3.14 `Angle gazebo::math::Angle::operator/= (const Angle & _angle)`

Division set, this = this / *_angle*.

Parameters

in	<i>_angle</i>	Angle (p. 107) for division
----	---------------	------------------------------------

Returns

angle

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

Less than operator.

Parameters

in	<i>_angle</i>	Angle (p. 107) to check
----	---------------	--------------------------------

Returns

true if this < *_angle*

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

Less or equal operator.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 107) to check
-----------------	---------------------	--------------------------------

Returns

true if this \leq `_angle`

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

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

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 107) to check for equality
-----------------	---------------------	---

Returns

true if this == `_angle`

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

Greater than operator.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 107) to check
-----------------	---------------------	--------------------------------

Returns

true if this $>$ `_angle`

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

Greater or equal operator.

Parameters

<code>in</code>	<code>_angle</code>	Angle (p. 107) to check
-----------------	---------------------	--------------------------------

Returns

true if this \geq `_angle`

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

Get the angle in radians.

Returns

double containing the angle's radian value

10.2.3.21 void gazebo::math::Angle::SetFromDegree (double *_degree*)

Set the value from an angle in degrees.

Parameters

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

10.2.3.22 void gazebo::math::Angle::SetFromRadian (double *_radian*)

Set the value from an angle in radians.

Parameters

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

10.2.4 Friends And Related Function Documentation**10.2.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::math::Angle & *_a*) [friend]**

Stream insertion operator.

Outputs in degrees

Parameters

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

Returns

The output stream

10.2.4.2 std::istream& operator>> (std::istream & *_in*, gazebo::math::Angle & *_a*) [friend]

Stream extraction operator.

Assumes input is in degrees

Parameters

<i>in</i>	input stream
<i>pt</i>	angle to read value into

Returns

The input stream

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

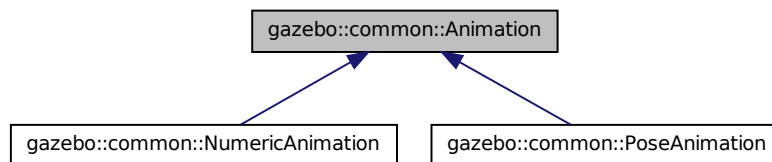
- **Angle.hh**

10.3 gazebo::common::Animation Class Reference

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

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

Inheritance diagram for gazebo::common::Animation:

**Public Member Functions**

- **Animation** (const std::string &_name, double _length, bool _loop)
Constructor.
- virtual **~Animation** ()
Destructor.
- void **AddTime** (double _time)
Add time to the animation.
- **KeyFrame** * **GetKeyFrame** (unsigned int _index) const
Get a key frame using an index value.
- unsigned int **GetKeyFrameCount** () const
Return the number of key frames in the animation.
- double **GetLength** () const
Return the duration of the animation.
- double **GetTime** () const
Return the current time position.
- void **SetLength** (double _len)
Set the duration of the animation.
- void **SetTime** (double _time)
Set the current time position of the animation.

Protected Types

- typedef std::vector< **KeyFrame** * > **KeyFrame_V**
array of keyframe type alias

Protected Member Functions

- double **GetKeyFramesAtTime** (double *_time*, **KeyFrame** **_kf1, **KeyFrame** **_kf2, unsigned int &_firstKeyIndex) const
Get the two key frames that bound a time value.

Protected Attributes

- bool **build**
determines if the interpolation splines need building
- **KeyFrame_V** **keyFrames**
array of key frames
- double **length**
animation duration
- bool **loop**
true if animation repeats
- std::string **name**
animation name
- double **timePos**
current time position

10.3.1 Detailed Description

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

10.3.2 Member Typedef Documentation

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

array of keyframe type alias

10.3.3 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

10.3.3.2 virtual gazebo::common::Animation::~~Animation () [virtual]

Destructor.

10.3.4 Member Function Documentation

10.3.4.1 void gazebo::common::Animation::AddTime (double *_time*)

Add time to the animation.

Parameters

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

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

Get a key frame using an index value.

Parameters

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

Returns

A pointer the keyframe, NULL if the *_index* is invalid

10.3.4.3 unsigned int gazebo::common::Animation::GetKeyFrameCount () const

Return the number of key frames in the animation.

Returns

The number of keyframes

10.3.4.4 double gazebo::common::Animation::GetKeyFramesAtTime (double *_time*, KeyFrame ** *_kf1*, KeyFrame ** *_kf2*, unsigned int & *_firstKeyIndex*) const [protected]

Get the two key frames that bound a time value.

Parameters

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

Returns

The time between the two keyframe

10.3.4.5 `double gazebo::common::Animation::GetLength () const`

Return the duration of the animation.

Returns

Duration of the animation in seconds

10.3.4.6 `double gazebo::common::Animation::GetTime () const`

Return the current time position.

Returns

The time position in seconds

10.3.4.7 `void gazebo::common::Animation::SetLength (double _len)`

Set the duration of the animation.

Parameters

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

10.3.4.8 `void gazebo::common::Animation::SetTime (double _time)`

Set the current time position of the animation.

Parameters

<code>in</code>	<code>_time</code>	The time position in seconds
-----------------	--------------------	------------------------------

10.3.5 Member Data Documentation**10.3.5.1** `bool gazebo::common::Animation::build` `[mutable], [protected]`

determines if the interpolation splines need building

10.3.5.2 `KeyFrame_V gazebo::common::Animation::keyFrames` `[protected]`

array of key frames

10.3.5.3 `double gazebo::common::Animation::length` `[protected]`

animation duration

10.3.5.4 `bool gazebo::common::Animation::loop` [protected]

true if animation repeats

10.3.5.5 `std::string gazebo::common::Animation::name` [protected]

animation name

10.3.5.6 `double gazebo::common::Animation::timePos` [protected]

current time position

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

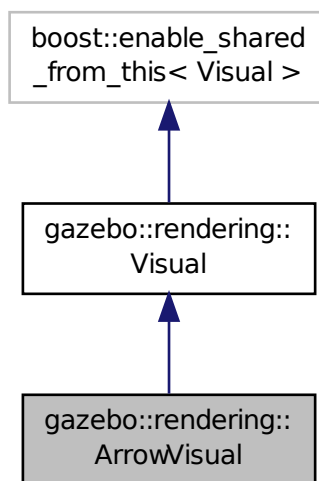
- **Animation.hh**

10.4 gazebo::rendering::ArrowVisual Class Reference

Basic arrow visualization.

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

Inheritance diagram for gazebo::rendering::ArrowVisual:



Public Member Functions

- **ArrowVisual** (const std::string &_name, **VisualPtr** _vis)

Constructor.

- virtual `~ArrowVisual ()`
Destructor.
- virtual void `Load ()`
Load the visual with default parameters.
- void `ShowRotation ()`
Show the rotation of the visual.

Additional Inherited Members

10.4.1 Detailed Description

Basic arrow visualization.

10.4.2 Constructor & Destructor Documentation

10.4.2.1 `gazebo::rendering::ArrowVisual::ArrowVisual (const std::string & _name, VisualPtr _vis)`

Constructor.

Parameters

<code>in</code>	<code>_name</code>	Name of the arrow visual
<code>in</code>	<code>_vis</code>	Pointer to the parent visual

10.4.2.2 `virtual gazebo::rendering::ArrowVisual::~~ArrowVisual () [virtual]`

Destructor.

10.4.3 Member Function Documentation

10.4.3.1 `virtual void gazebo::rendering::ArrowVisual::Load () [virtual]`

Load the visual with default parameters.

Reimplemented from `gazebo::rendering::Visual` (p. 841).

10.4.3.2 `void gazebo::rendering::ArrowVisual::ShowRotation ()`

Show the rotation of the visual.

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

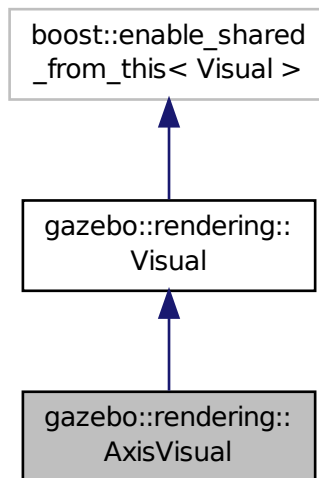
- **ArrowVisual.hh**

10.5 `gazebo::rendering::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

10.5.1 Detailed Description

Basic axis visualization.

10.5.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	<i>_name</i>	Name of the AxisVisual (p. 120)
in	<i>_vis</i>	Parent visual

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

Destructor.

10.5.3 Member Function Documentation

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

Load the axis visual.

Reimplemented from **gazebo::rendering::Visual** (p. 841).

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

Scale the X axis.

Parameters

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

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

Scale the Y axis.

Parameters

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

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

Scale the Z axis.

Parameters

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

10.5.3.5 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

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

Load the rotation tube.

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

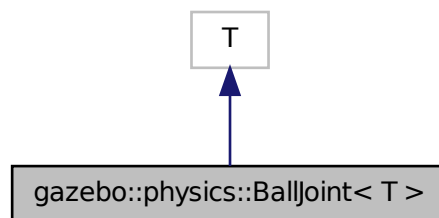
- **AxisVisual.hh**

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

Base (p. 125) 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. 123).*

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

10.6.1 Detailed Description

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

Base (p. 125) class for a ball joint.

Each physics engine should implement this class.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 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.

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

Destructor.

10.6.3 Member Function Documentation

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

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

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

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

Template to ::Load the **BallJoint** (p. 123).

Parameters

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

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

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

10.6.3.7 `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:

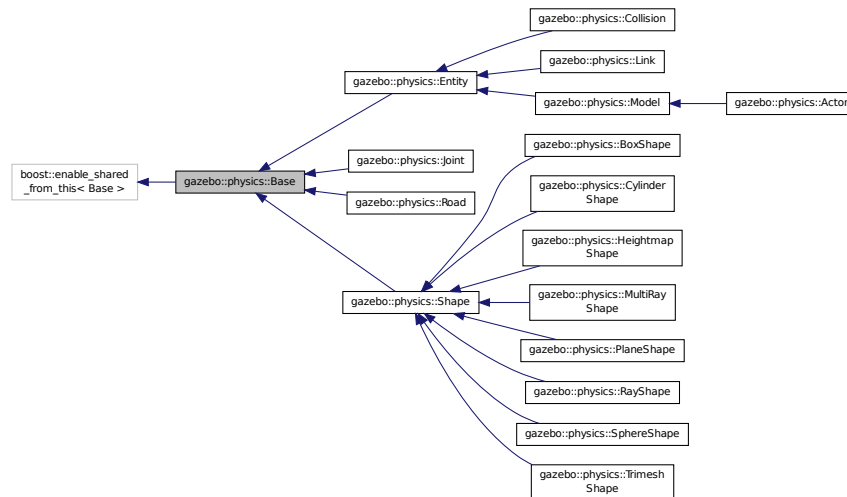
- **BallJoint.hh**

10.7 gazebo::physics::Base Class Reference

Base (p. 125) 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. 853) 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. 128)'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.

10.7.1 Detailed Description

Base (p. 125) class for most physics classes.

10.7.2 Member Enumeration Documentation

10.7.2.1 enum gazebo::physics::Base::EntityType

Unique identifiers for all entity types.

Enumerator:

- BASE** **Base** (p. 125) type.
- ENTITY** **Entity** (p. 265) type.
- MODEL** **Model** (p. 460) type.
- LINK** **Link** (p. 398) type.
- COLLISION** **Collision** (p. 180) type.
- ACTOR** **Actor** (p. 101) type.
- LIGHT** **Light** type.
- VISUAL** **Visual** type.
- JOINT** **Joint** (p. 366) type.
- BALL_JOINT** **BallJoint** (p. 123) type.
- HINGE2_JOINT** **Hing2Joint** type.
- HINGE_JOINT** **HingeJoint** (p. 347) type.
- SLIDER_JOINT** **SliderJoint** (p. 692) type.
- SCREW_JOINT** **ScrewJoint** (p. 646) type.
- UNIVERSAL_JOINT** **UniversalJoint** (p. 765) type.
- SHAPE** **Shape** (p. 668) type.
- BOX_SHAPE** **BoxShape** (p. 140) type.
- CYLINDER_SHAPE** **CylinderShape** (p. 233) type.
- HEIGHTMAP_SHAPE** **HeightmapShape** (p. 341) type.
- MAP_SHAPE** **MapShape** type.
- MULTIRAY_SHAPE** **MultiRayShape** (p. 492) type.
- RAY_SHAPE** **RayShape** (p. 603) type.
- PLANE_SHAPE** **PlaneShape** (p. 549) type.
- SPHERE_SHAPE** **SphereShape** (p. 694) type.
- TRIMESH_SHAPE** **TrimeshShape** (p. 761) type.

10.7.3 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

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

Destructor.

10.7.4 Member Function Documentation

10.7.4.1 void gazebo::physics::Base::AddChild (BasePtr *_child*)

Add a child to this entity.

Parameters

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

10.7.4.2 void gazebo::physics::Base::AddType (EntityType *_type*)

Add a type specifier.

Parameters

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

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

Finalize the object.

Reimplemented in **gazebo::physics::Actor** (p. 104), **gazebo::physics::Model** (p. 465), **gazebo::physics::Entity** (p. 269), **gazebo::physics::Link** (p. 406), and **gazebo::physics::Collision** (p. 184).

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

Get a child or self by id.

Parameters

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

Returns

A pointer to the object, NULL if not found

10.7.4.5 BasePtr gazebo::physics::Base::GetByName (const std::string & *_name*)

Get by name.

Parameters

in	<i>_name</i>	Get a child (or self) object by name
----	--------------	--------------------------------------

Returns

A pointer to the object, NULL if not found

10.7.4.6 **BasePtr gazebo::physics::Base::GetChild (unsigned int *_i*) const**

Get a child by index.

Parameters

in	<i>_i</i>	Index of the child to retrieve.
----	-----------	---------------------------------

Returns

A pointer to the object, NULL if the index is invalid.

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

Get a child by name.

Parameters

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

Returns

A pointer to the object, NULL if not found

10.7.4.8 **unsigned int gazebo::physics::Base::GetChildCount () const**

Get the number of children.

Returns

The number of children.

10.7.4.9 **unsigned int gazebo::physics::Base::GetId () const**

Return the ID of this entity.

This id is unique.

Returns

Integer ID.

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

Return the name of the entity.

Returns

Name of the entity.

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

Get the parent.

Returns

Pointer to the parent entity.

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

Return the ID of the parent.

Returns

Integer ID.

10.7.4.13 `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.

10.7.4.14 `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.

10.7.4.15 `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. 104), and `gazebo::physics::Model` (p. 468).

10.7.4.16 unsigned int gazebo::physics::Base::GetType () const

Get the full type definition.

Returns

The full type definition.

10.7.4.17 const WorldPtr& gazebo::physics::Base::GetWorld () const

Get the **World** (p. 853) this object is in.

Returns

The **World** (p. 853) this object is part of.

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

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

Parameters

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

Returns

True if this object's type definition has the.

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

Initialize the object.

Reimplemented in **gazebo::physics::RayShape** (p. 607), **gazebo::physics::Joint** (p. 375), **gazebo::physics::Actor** (p. 104), **gazebo::physics::Model** (p. 469), **gazebo::physics::Link** (p. 411), **gazebo::physics::Collision** (p. 187), **gazebo::physics::HeightmapShape** (p. 344), **gazebo::physics::TrimeshShape** (p. 764), **gazebo::physics::Multi-RayShape** (p. 498), **gazebo::physics::PlaneShape** (p. 552), **gazebo::physics::Road** (p. 623), **gazebo::physics::Shape** (p. 670), **gazebo::physics::SphereShape** (p. 697), **gazebo::physics::BoxShape** (p. 143), and **gazebo::physics::CylinderShape** (p. 235).

10.7.4.20 bool gazebo::physics::Base::IsSelected () const

True if the entity is selected by the user.

Returns

True if the entity is selected.

10.7.4.21 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.376), **gazebo::physics::Actor** (p.104), **gazebo::physics::Entity** (p.273), **gazebo::physics::Model** (p.469), **gazebo::physics::Link** (p.411), **gazebo::physics::Collision** (p.188), **gazebo::physics::HeightmapShape** (p.345), and **gazebo::physics::Road** (p.623).

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

Returns true if the entities are the same.

Checks only the name.

Parameters

<code>in</code>	<code>_ent</code>	Base (p.125) object to compare with.
-----------------	-------------------	---

Returns

True if the entities are the same.

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

Print this object to screen via gzmsg.

Parameters

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

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

Remove a child from this entity.

Parameters

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

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

Remove a child by name.

Parameters

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

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

Remove all children.

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

Reset the object.

Reimplemented in `gazebo::physics::Joint` (p. 376), `gazebo::physics::Model` (p. 470), `gazebo::physics::Entity` (p. 273), and `gazebo::physics::Link` (p. 412).

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

Calls recursive Reset on one of the `Base::EntityType` (p. 128)'s.

Parameters

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

10.7.4.29 `virtual void gazebo::physics::Base::SetName (const std::string & _name) [virtual]`

Set the name of the entity.

Parameters

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

Reimplemented in `gazebo::physics::Entity` (p. 274).

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

Set the parent.

Parameters

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

10.7.4.31 `void gazebo::physics::Base::SetSaveable (bool _v)`

Set whether the object should be "saved", when the user selects to save the world to xml.

Parameters

in	<code>_v</code>	Set to True if the object should be saved.
----	-----------------	--

10.7.4.32 `virtual bool gazebo::physics::Base::SetSelected (bool _show) [virtual]`

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

Parameters

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

Reimplemented in `gazebo::physics::Link` (p. 414).

10.7.4.33 `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 World (p. 853) this object is part of.
-----------------	------------------------	---

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

Update the object.

Reimplemented in **gazebo::physics::MultiRayShape** (p. 498), **gazebo::physics::Joint** (p. 379), **gazebo::physics::Actor** (p. 105), **gazebo::physics::RayShape** (p. 608), **gazebo::physics::Link** (p. 415), **gazebo::physics::Model** (p. 473), and **gazebo::physics::TrimeshShape** (p. 764).

10.7.4.35 `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. 379), **gazebo::physics::Actor** (p. 105), **gazebo::physics::Model** (p. 474), **gazebo::physics::Entity** (p. 276), **gazebo::physics::Link** (p. 415), and **gazebo::physics::Collision** (p. 189).

10.7.5 Member Data Documentation

10.7.5.1 `Base_V gazebo::physics::Base::children [protected]`

Children of this entity.

10.7.5.2 `Base_V::iterator gazebo::physics::Base::childrenEnd [protected]`

End of the children vector.

10.7.5.3 `BasePtr gazebo::physics::Base::parent [protected]`

Parent of this entity.

10.7.5.4 `sdf::ElementPtr gazebo::physics::Base::sdf [protected]`

The SDF values for this object.

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

Pointer to the world.

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

- **Base.hh**

10.8 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.

10.8.1 Detailed Description

Mathematical representation of a box and related functions.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 gazebo::math::Box::Box ()

Default constructor.

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

Constructor.

Parameters

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

10.8.2.3 gazebo::math::Box::Box (const Box & _b)

Copy Constructor.

Parameters

<code>in</code>	<code>_b</code>	Box (p. 136) to copy
-----------------	-----------------	-----------------------------

10.8.2.4 virtual gazebo::math::Box::~~Box () [virtual]

Destructor.

10.8.3 Member Function Documentation

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

Get the box center.

Returns

The center position of the box

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

Get the size of the box.

Returns

Size of the box

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

Get the length along the x dimension.

Returns

Double value of the length in the x dimension

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

Get the length along the y dimension.

Returns

Double value of the length in the y dimension

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

Get the length along the z dimension.

Returns

Double value of the length in the z dimension

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

Merge a box with this box.

Parameters

<code>in</code>	<code>_box</code>	Box (p. 136) to add to this box
-----------------	-------------------	--

10.8.3.7 Box gazebo::math::Box::operator+ (const Box & _b) const

Addition operator.

result = this + _b

Parameters

in	_b	Box (p. 136) to add
----	----	---------------------

Returns

The new box

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

Addition set operator.

this = this + _b

Parameters

in	_b	Box (p. 136) to add
----	----	---------------------

Returns

This new box

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

Subtract a vector from the min and max values.

Parameters

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

Returns

The new box

10.8.3.10 Box& gazebo::math::Box::operator= (const Box & _b)

Assignment operator.

Set this box to the parameter

Parameters

in	_b	Box (p. 136) to copy
----	----	----------------------

Returns

The new box.

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

Equality test operator.

Parameters

<code>in</code>	<code>_b</code>	Box (p. 136) to test
-----------------	-----------------	-----------------------------

Returns

True if equal

10.8.4 Friends And Related Function Documentation

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

Output operator.

Parameters

<code>in</code>	<code>_out</code>	Output stream
<code>in</code>	<code>_b</code>	Box (p. 136) to output to the stream

Returns

The stream

10.8.5 Member Data Documentation

10.8.5.1 `Vector3 gazebo::math::Box::max`

Maximum corner of the box.

10.8.5.2 `Vector3 gazebo::math::Box::min`

Minimum corner of the box.

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

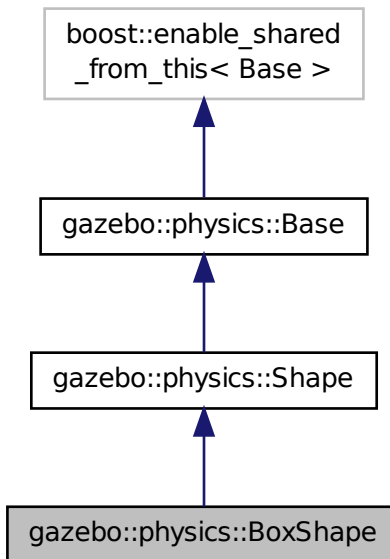
- **Box.hh**

10.9 `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.
- void **FillShapeMsg** (msgs::Geometry &_msg) **GAZEBO_DEPRECATED**
Deprecated.
- virtual void **GetInertial** (double _mass, **InertialPtr** _inertial) const **GAZEBO_DEPRECATED**
Deprecated.
- virtual double **GetMass** (double _density) const **GAZEBO_DEPRECATED**
Deprecated.
- **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

10.9.1 Detailed Description

Box geometry primitive.

10.9.2 Constructor & Destructor Documentation

10.9.2.1 `gazebo::physics::BoxShape::BoxShape (CollisionPtr _parent)` [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent Collision (p. 180).
----	----------------	-----------------------------------

10.9.2.2 `virtual gazebo::physics::BoxShape::~~BoxShape ()` [virtual]

Destructor.

10.9.3 Member Function Documentation

10.9.3.1 `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. 669).

10.9.3.2 `void gazebo::physics::BoxShape::FillShapeMsg (msgs::Geometry & _msg)` [virtual]

Deprecated.

Reimplemented from **gazebo::physics::Shape** (p. 670).

10.9.3.3 `virtual void gazebo::physics::BoxShape::GetInertial (double _mass, InertialPtr _inertial) const` [virtual]

Deprecated.

Reimplemented from **gazebo::physics::Shape** (p. 670).

10.9.3.4 `virtual double gazebo::physics::BoxShape::GetMass (double _density) const` [virtual]

Deprecated.

Reimplemented from **gazebo::physics::Shape** (p. 670).

10.9.3.5 `math::Vector3 gazebo::physics::BoxShape::GetSize () const`

Get the size of the box.

Returns

The size of each side of the box.

10.9.3.6 `virtual void gazebo::physics::BoxShape::Init () [virtual]`

Initialize the box.

Implements **`gazebo::physics::Shape`** (p. 670).

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

Process a geometry message.

Parameters

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

Implements **`gazebo::physics::Shape`** (p. 670).

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

Set the size of the box.

Parameters

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

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

- **`BoxShape.hh`**

10.10 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.

10.10.1 Detailed Description

Handles loading BVH animation files.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 gazebo::common::BVHLoader::BVHLoader ()

Constructor.

10.10.2.2 gazebo::common::BVHLoader::~~BVHLoader ()

Desutrctor.

10.10.3 Member Function Documentation

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

Load a BVH file.

Parameters

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

Returns

A pointer to a new **Skeleton** (p. 672)

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

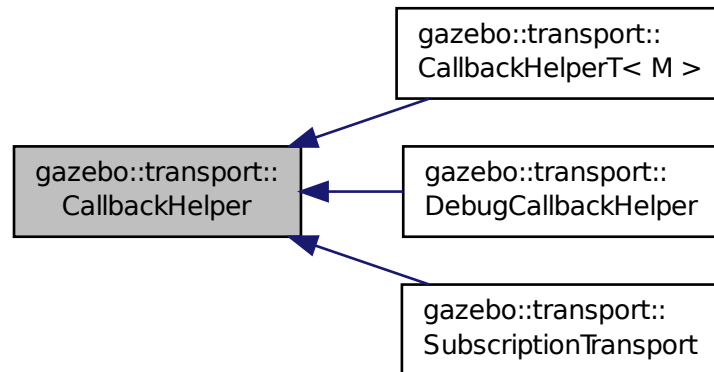
- **BVHLoader.hh**

10.11 gazebo::transport::CallbackHelper Class Reference

A helper class to handle callbacks when messages arrive.

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

Inheritance diagram for gazebo::transport::CallbackHelper:



Public Member Functions

- **CallbackHelper** ()
Constructor.
- virtual **~CallbackHelper** ()
Destructor.
- 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**

10.11.1 Detailed Description

A helper class to handle callbacks when messages arrive.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 gazebo::transport::CallbackHelper::CallbackHelper () [inline]

Constructor.

10.11.2.2 `virtual gazebo::transport::CallbackHelper::~~CallbackHelper () [inline],[virtual]`

Destructor.

10.11.3 Member Function Documentation

10.11.3.1 `bool gazebo::transport::CallbackHelper::GetLatching () const [inline]`

Is the callback latching?

Returns

true if the callback is latching, false otherwise

References latching.

10.11.3.2 `virtual std::string gazebo::transport::CallbackHelper::GetMsgType () const [inline],[virtual]`

Get the typename of the message that is handled.

Returns

String representation of the message type

Reimplemented in `gazebo::transport::DebugCallbackHelper` (p. 237), and `gazebo::transport::CallbackHelperT< M >` (p. 148).

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

Process new incoming data.

Parameters

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

Returns

true if successfully processed; false otherwise

Implemented in `gazebo::transport::DebugCallbackHelper` (p. 237), `gazebo::transport::CallbackHelperT< M >` (p. 148), and `gazebo::transport::SubscriptionTransport` (p. 720).

10.11.3.4 `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::DebugCallbackHelper` (p. 237), `gazebo::transport::CallbackHelperT< M >` (p. 148), and `gazebo::transport::SubscriptionTransport` (p. 721).

10.11.4 Member Data Documentation

10.11.4.1 bool gazebo::transport::CallbackHelper::latching [protected]

Referenced by GetLatching().

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

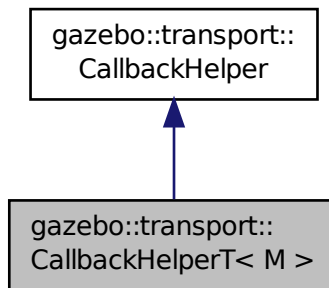
- **CallbackHelper.hh**

10.12 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)
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

10.12.1 Detailed Description

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

Callback helper Template.

10.12.2 Constructor & Destructor Documentation

10.12.2.1 `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
----	------------------	---

10.12.3 Member Function Documentation

10.12.3.1 `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. 146).

References gthrow, and NULL.

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

Process new incoming data.

Parameters

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

Returns

true if successfully processed; false otherwise

Implements `gazebo::transport::CallbackHelper` (p. 146).

10.12.3.3 `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. 146).

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

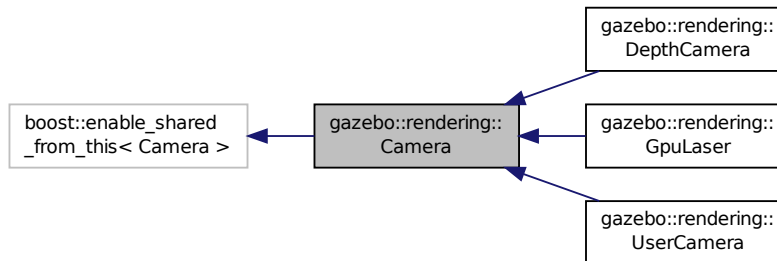
- **CallbackHelper.hh**

10.13 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 height of the image.
- 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.
- 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 **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. 170) 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. 632) *node 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. 632) *node that controls camera position.*
- **sdf::ElementPtr sdf**
Camera (p. 149)'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.

10.13.1 Detailed Description

Basic camera sensor.

This is the base class for all cameras.

10.13.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	<i>_namePrefix</i>	Unique prefix name for the camera.
in	<i>_scene</i>	Scene (p. 632) that will contain the camera
in	<i>_autoRender</i>	Almost everyone should leave this as true.

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

Destructor.

10.13.3 Member Function Documentation

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

Internal function used to indicate that an animation has completed.

Reimplemented in `gazebo::rendering::UserCamera` (p. 777).

10.13.3.2 `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

10.13.3.3 `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

10.13.3.4 `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. 777).

10.13.3.5 `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 pointer to the connection. This must be kept in scope.

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

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

Set the render target.

Parameters

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

10.13.3.7 `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`.

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

Enable or disable saving.

Parameters

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

10.13.3.9 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. 315), `gazebo::rendering::DepthCamera` (p. 241), and `gazebo::rendering::UserCamera` (p. 778).

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

Get the aspect ratio.

Returns

The aspect ratio (width / height) in pixels

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

Get the average FPS.

Returns

The average frames per second

10.13.3.12 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

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

Get the camera's direction vector.

Returns

Direction the camera is facing

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

Get the far clip distance.

Returns

Far clip distance

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

Get the next frame filename based on SDF parameters.

Returns

The frame's filename

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

Get the camera FOV (horizontal)

Returns

The horizontal field of view

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

Get the image size in bytes.

Returns

Size in bytes

10.13.3.18 `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

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

Get a pointer to the image data.

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

Parameters

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

Returns

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

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

Get the depth of the image.

Returns

Depth of the image

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

Get the height of the image.

Returns

String representation of the image format.

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

Get the height of the image.

Returns

Image height

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

Get the width of the image.

Returns

Image width

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

Returns true if initialized.

Returns

True if the camera is initialized

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

Get the last time the camera was rendered.

Returns

Time the camera was last rendered

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

Get the camera's name.

Returns

The name of the camera

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

Get the near clip distance.

Returns

Near clip distance

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

Get a pointer to the ogre camera.

Returns

Pointer to the OGRE camera

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

Get the render Hz rate.

Returns

The Hz rate

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

Get the render texture.

Returns

Pointer to the render texture

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

Get the viewport right vector.

Returns

The viewport right vector

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

Get the scene this camera is in.

Returns

Pointer to scene containing this camera

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

Get the camera's scene node.

Returns

The scene node the camera is attached to

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

Get the height of the off-screen render texture.

Returns

Render texture height

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

Get the width of the off-screen render texture.

Returns

Render texture width

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

Get the triangle count.

Returns

The current triangle count

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

Get the viewport up vector.

Returns

The viewport up vector

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

Get the camera FOV (vertical)

Returns

The vertical field of view

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

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

Returns

Pointer to the `Ogre::Viewport`

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

Get the viewport height in pixels.

Returns

The viewport height

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

Get the viewport width in pixels.

Returns

The viewport width

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

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

Returns

The ID of the window.

10.13.3.43 `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

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

Get the global pose of the camera.

Returns

Pose of the camera in the world coordinate frame

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

Get the camera position in the world.

Returns

The world position of the camera

10.13.3.46 `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. 581)

10.13.3.47 `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.

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

Initialize the camera.

Reimplemented in `gazebo::rendering::GpuLaser` (p. 315), `gazebo::rendering::DepthCamera` (p. 241), and `gazebo::rendering::UserCamera` (p. 779).

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

Return true if the camera has been initialized.

Returns

True if initialized was successful

References initialized.

10.13.3.50 `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

10.13.3.51 `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

10.13.3.52 `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. 779).

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

Load the camera with default parameters.

Reimplemented in `gazebo::rendering::GpuLaser` (p. 315), `gazebo::rendering::DepthCamera` (p. 241), and `gazebo::rendering::UserCamera` (p. 779).

10.13.3.54 `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. 165)

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. 780).

10.13.3.55 `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. 165)

Parameters

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

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

Post render.

Called after the render signal.

Reimplemented in `gazebo::rendering::GpuLaser` (p. 316), `gazebo::rendering::DepthCamera` (p. 242), and `gazebo::rendering::UserCamera` (p. 780).

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

Render the camera.

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

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

Implementation of the render call.

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

Rotate the camera around the pitch axis.

Parameters

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

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

Rotate the camera around the yaw axis.

Parameters

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

10.13.3.61 `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

10.13.3.62 `static bool gazebo::rendering::Camera::SaveFrame (const unsigned char * _image, unsigned int _width, unsigned int _height, int _depth, const std::string & _format, const std::string & _filename) [static]`

Save a frame using an image buffer.

Parameters

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

Returns

True if saving was successful

10.13.3.63 void gazebo::rendering::Camera::SetAspectRatio (float *_ratio*)

Set the aspect ratio.

Parameters

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

10.13.3.64 void gazebo::rendering::Camera::SetCaptureData (bool *_value*)

Set whether to capture data.

Parameters

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

10.13.3.65 void gazebo::rendering::Camera::SetClipDist (float *_near*, float *_far*)

Set the clip distances.

Parameters

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

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

Set the camera FOV (horizontal)

Parameters

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

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

Set the image height.

Parameters

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

10.13.3.68 void gazebo::rendering::Camera::SetImageSize (unsigned int *_w*, unsigned int *_h*)

Set the image size.

Parameters

in	<i>_w</i>	Image width
in	<i>_h</i>	Image height

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

Set the image height.

Parameters

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

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

Set the camera's name.

Parameters

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

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

Set the render Hz rate.

Parameters

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

10.13.3.72 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. 781).

10.13.3.73 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
----	------------------	--

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

Set the scene this camera is viewing.

Parameters

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

10.13.3.75 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
----	--------------	---

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

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

Set the global pose of the camera.

Parameters

in	<i>_pose</i>	The new math::Pose (p. 556) of the camera
----	--------------	--

Reimplemented in **gazebo::rendering::UserCamera** (p. 781).

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

Set the world position.

Parameters

in	<i>_pos</i>	The new position of the camera
----	-------------	--------------------------------

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

Set the world orientation.

Parameters

in	<i>_quat</i>	The new orientation of the camera
----	--------------	-----------------------------------

10.13.3.80 void gazebo::rendering::Camera::ShowWireframe (bool *_s*)

Set whether to view the world in wireframe.

Parameters

in	<i>_s</i>	Set to True to render objects as wireframe
----	-----------	--

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

Toggle whether to view the world in wireframe.

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

Set the camera to track a scene node.

Parameters

in	<i>_visualName</i>	Name of the visual to track
----	--------------------	-----------------------------

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

Implementation of the **Camera::TrackVisual** (p. 170) call.

Parameters

in	<i>_visualName</i>	Name of the visual to track
----	--------------------	-----------------------------

Returns

True if able to track the visual

10.13.3.84 virtual bool gazebo::rendering::Camera::TrackVisualImpl (VisualPtr *_visual*) [protected],[virtual]

Set the camera to track a scene node.

Parameters

in	<i>_visual</i>	The visual to track
----	----------------	---------------------

Returns

True if able to track the visual

Reimplemented in **gazebo::rendering::UserCamera** (p. 782).

10.13.3.85 void gazebo::rendering::Camera::Translate (const math::Vector3 & *_direction*)

Translate the camera.

Parameters

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

10.13.3.86 virtual void gazebo::rendering::Camera::Update () [virtual]

Reimplemented in **gazebo::rendering::UserCamera** (p. 782).

10.13.4 Member Data Documentation

10.13.4.1 Ogre::AnimationState* gazebo::rendering::Camera::animState [protected]

Animation state, used to animate the camera.

10.13.4.2 unsigned char* gazebo::rendering::Camera::bayerFrameBuffer [protected]

Buffer for a bayer image frame.

10.13.4.3 Ogre::Camera* gazebo::rendering::Camera::camera [protected]

The OGRE camera.

10.13.4.4 bool gazebo::rendering::Camera::captureData [protected]

True to capture frames into an image buffer.

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

The camera's event connections.

10.13.4.6 int gazebo::rendering::Camera::imageFormat [protected]

Format for saving images.

10.13.4.7 int gazebo::rendering::Camera::imageHeight [protected]

Save image height.

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

Save image width.

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

True if initialized.

Referenced by `IsInitialized()`.

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

Time the last frame was rendered.

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

Name of the camera.

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

True if new data is available.

10.13.4.13 `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()`.

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

User callback for when an animation completes.

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

Scene (p. 632) node that controls camera pitch.

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

Previous time the camera animation was updated.

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

Target that renders frames.

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

Texture that receives results from rendering.

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

List of requests.

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

Number of saved frames.

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

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

Pointer to the scene.

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

Scene (p. 632) node that controls camera position.

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

Camera (p. 149)'s SDF values.

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

Height of the render texture.

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

Width of the render texture.

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

Viewport the ogre camera uses.

10.13.4.28 `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:

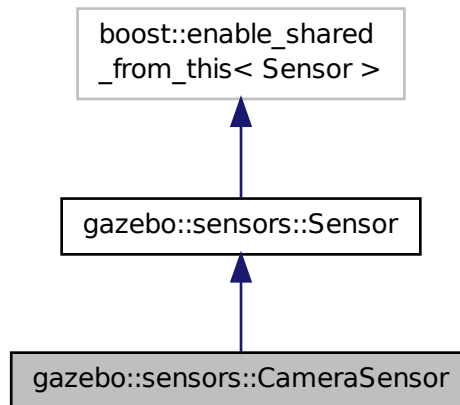
- **Camera.hh**

10.14 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. 149).*
- 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 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

10.14.1 Detailed Description

Basic camera sensor.

This sensor is used for simulating standard monocular cameras

10.14.2 Constructor & Destructor Documentation

10.14.2.1 gazebo::sensors::CameraSensor::CameraSensor ()

Constructor.

10.14.2.2 virtual gazebo::sensors::CameraSensor::~~CameraSensor () [virtual]

Destructor.

10.14.3 Member Function Documentation

10.14.3.1 virtual void gazebo::sensors::CameraSensor::Fini () [protected],[virtual]

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 655).

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

Returns a pointer to the **rendering::Camera** (p. 149).

Returns

The Pointer to the camera sensor.

10.14.3.3 const unsigned char* gazebo::sensors::CameraSensor::GetImageData ()

Gets the raw image data from the sensor.

Returns

The pointer to the image data array.

10.14.3.4 unsigned int gazebo::sensors::CameraSensor::GetImageHeight () const

Gets the height of the image in pixels.

Returns

The image height in pixels.

10.14.3.5 unsigned int gazebo::sensors::CameraSensor::GetImageWidth () const

Gets the width of the image in pixels.

Returns

The image width in pixels.

10.14.3.6 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. 656).

10.14.3.7 virtual void gazebo::sensors::CameraSensor::Init () [virtual]

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 657).

10.14.3.8 virtual void gazebo::sensors::CameraSensor::Load (const std::string & _worldName, sdf::ElementPtr _sdf) [virtual]

Load the sensor with SDF parameters.

Parameters

in	<code>_sdf</code>	SDF Sensor (p. 652) parameters
in	<code>_worldName</code>	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 657).

10.14.3.9 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. 658).

10.14.3.10 `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.

10.14.3.11 `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. 658).

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

Update the sensor information.

Parameters

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

Reimplemented from **gazebo::sensors::Sensor** (p. 659).

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

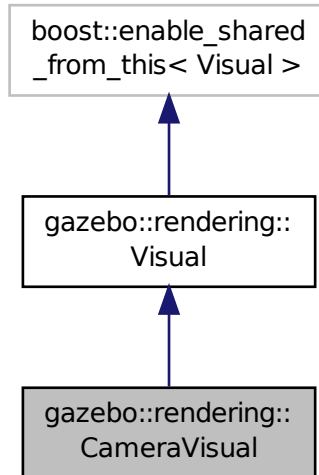
- **CameraSensor.hh**

10.15 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. 828).*

Additional Inherited Members

10.15.1 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.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 gazebo::rendering::CameraVisual::CameraVisual (const std::string & _name, **VisualPtr** _vis)

Constructor.

Parameters

in	<code>_name</code>	Name of the Visual (p. 828)
in	<code>_vis</code>	Pointer to the parent Visual (p. 828)

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

Destructor.

10.15.3 Member Function Documentation

10.15.3.1 `void gazebo::rendering::CameraVisual::Load (unsigned int _width, unsigned int _height)`

Load the **Visual** (p. 828).

Parameters

in	<code>_width</code>	Width of the Camera (p. 149) image
in	<code>_height</code>	Height of the Camera (p. 149) image

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

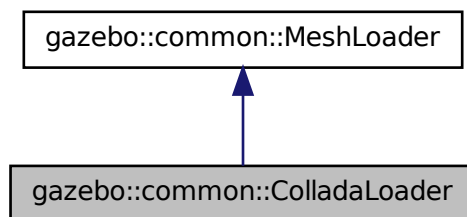
- **CameraVisual.hh**

10.16 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.

10.16.1 Detailed Description

Class used to load Collada mesh files.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 gazebo::common::ColladaLoader::ColladaLoader ()

Constructor.

10.16.2.2 virtual gazebo::common::ColladaLoader::~~ColladaLoader () [virtual]

Destructor.

10.16.3 Member Function Documentation

10.16.3.1 virtual Mesh* gazebo::common::ColladaLoader::Load (const std::string &_filename) [virtual]

Load a mesh.

Parameters

<code>in</code>	<code>_filename</code>	Collada file to load
-----------------	------------------------	----------------------

Returns

Pointer to a new **Mesh** (p. 448)

Implements `gazebo::common::MeshLoader` (p. 455).

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

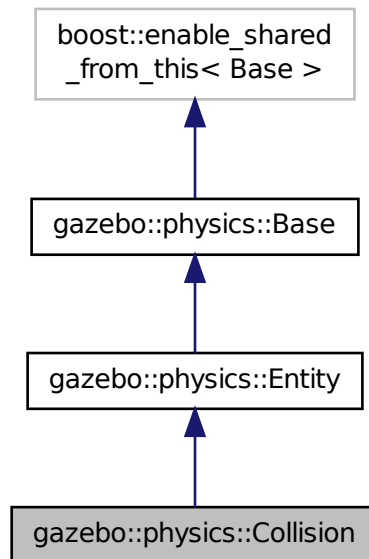
- **ColladaLoader.hh**

10.17 gazebo::physics::Collision Class Reference

Base (p. 125) 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 **FillCollisionMsg** (msgs::Collision &_msg) **GAZEBO_DEPRECATED**
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. 668).*

Additional Inherited Members

10.17.1 Detailed Description

Base (p. 125) class for all collision entities.

10.17.2 Constructor & Destructor Documentation

10.17.2.1 `gazebo::physics::Collision::Collision (LinkPtr .link)` `[explicit]`

Constructor.

Parameters

<i>in</i>	<i>_link</i>	Link (p. 398) that contains this collision object.
-----------	--------------	---

10.17.2.2 `virtual gazebo::physics::Collision::~~Collision ()` `[virtual]`

Destructor.

10.17.3 Member Function Documentation

10.17.3.1 `void gazebo::physics::Collision::AddContact (const Contact & .contact)`

Add an occurrence of a contact to this collision.

Parameters

<i>in</i>	<i>_contact</i>	The contact which was detected by a collision engine.
-----------	-----------------	---

10.17.3.2 `template<typename T> event::ConnectionPtr gazebo::physics::Collision::ConnectContact (T _subscriber)`
`[inline]`

Deprecated.

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

10.17.3.3 `void gazebo::physics::Collision::DisconnectContact (event::ConnectionPtr & _conn)` `[inline]`

Deprecated.

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

10.17.3.4 `void gazebo::physics::Collision::FillCollisionMsg (msgs::Collision & _msg)`

DEPRECATED.

10.17.3.5 `void gazebo::physics::Collision::FillMsg (msgs::Collision & _msg)`

Fill a collision message.

Parameters

out	_msg	The message to fill with this collision's data.
-----	------	---

10.17.3.6 `virtual void gazebo::physics::Collision::Fini ()` `[virtual]`

Finalize the collision.

Reimplemented from `gazebo::physics::Entity` (p. 269).

10.17.3.7 `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. 269).

10.17.3.8 `bool gazebo::physics::Collision::GetContactsEnabled () const`

Return true if contacts are on.

Returns

True if contacts are on.

10.17.3.9 `float gazebo::physics::Collision::GetLaserRetro () const`

Get the laser retro reflectiveness.

Returns

The laser retro value.

10.17.3.10 `LinkPtr gazebo::physics::Collision::GetLink () const`

Get the link this collision belongs to.

Returns

The parent **Link** (p. 398).

10.17.3.11 `ModelPtr gazebo::physics::Collision::GetModel () const`

Get the model this collision belongs to.

Returns

The parent model.

10.17.3.12 `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. 270).

10.17.3.13 `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. 271).

10.17.3.14 `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. 271).

10.17.3.15 `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. 271).

10.17.3.16 `ShapePtr gazebo::physics::Collision::GetShape () const`

Get the collision shape.

Returns

The collision shape.

10.17.3.17 `unsigned int gazebo::physics::Collision::GetShapeType ()`

Get the shape type.

Returns

The shape type.

See Also

`EntityType` (p. 128)

10.17.3.18 `CollisionState gazebo::physics::Collision::GetState ()`

Get the collision state.

Returns

The collision state.

10.17.3.19 `SurfaceParamsPtr gazebo::physics::Collision::GetSurface () const` [inline]

Get the surface parameters.

Returns

The surface parameters.

10.17.3.20 `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. 272).

10.17.3.21 `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. 272).

10.17.3.22 `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. 272).

10.17.3.23 `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. 272).

10.17.3.24 `virtual void gazebo::physics::Collision::Init () [virtual]`

Initialize the collision.

Reimplemented from **gazebo::physics::Base** (p. 132).

10.17.3.25 `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.

10.17.3.26 `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. 273).

10.17.3.27 `void gazebo::physics::Collision::ProcessMsg (const msgs::Collision & _msg)`

Update parameters from a message.

Parameters

in	_msg	Message to update from.
----	------	-------------------------

10.17.3.28 `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.
----	-------	------------------

10.17.3.29 `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.
----	-------	------------------

10.17.3.30 `void gazebo::physics::Collision::SetCollision (bool _placeable)`

Set the encapsulated collision object.

Parameters

in	_placeable	True to make the object m.
----	------------	----------------------------

10.17.3.31 `void gazebo::physics::Collision::SetContactsEnabled (bool _enable)`

Turn contact recording on or off.

Parameters

in	_enable	True to enable collision contacts.
----	---------	------------------------------------

10.17.3.32 void gazebo::physics::Collision::SetLaserRetro (float *_retro*)

Set the laser retro reflectiveness.

Parameters

in	<i>_retro</i>	The laser retro value.
----	---------------	------------------------

10.17.3.33 void gazebo::physics::Collision::SetShape (ShapePtr *_shape*)

Set the shape for this collision.

Parameters

in	<i>_shape</i>	The shape for this collision object.
----	---------------	--------------------------------------

10.17.3.34 void gazebo::physics::Collision::SetState (const CollisionState & *_state*)

Set the current collision state.

Parameters

in	<i>The</i>	collision state.
----	------------	------------------

10.17.3.35 virtual void gazebo::physics::Collision::UpdateParameters (sdf::ElementPtr *_sdf*) [virtual]

Update the parameters using new sdf values.

Parameters

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

Reimplemented from **gazebo::physics::Entity** (p. 276).

10.17.4 Member Data Documentation

10.17.4.1 LinkPtr gazebo::physics::Collision::link [protected]

The link this collision belongs to.

10.17.4.2 bool gazebo::physics::Collision::placeable [protected]

Flag for placeable.

10.17.4.3 ShapePtr gazebo::physics::Collision::shape [protected]

Pointer to **physics::Shape** (p. 668).

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

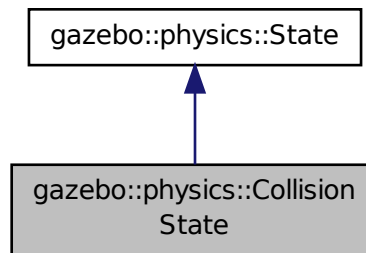
- [Collision.hh](#)

10.18 gazebo::physics::CollisionState Class Reference

Store state information of a [physics::Collision](#) (p. 180) 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.
- const **math::Pose** & **GetPose** () const
*Get the **Collision** (p. 180) 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

10.18.1 Detailed Description

Store state information of a **physics::Collision** (p. 180) object.

This class captures the entire state of a **Collision** (p. 180) at one specific time during a simulation run.

State (p. 702) of a **Collision** (p. 180) is its Pose.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 gazebo::physics::CollisionState::CollisionState ()

Default constructor.

10.18.2.2 gazebo::physics::CollisionState::CollisionState (const CollisionPtr _collision) [explicit]

Constructor.

Build a **CollisionState** (p. 190) from an existing **Collision** (p. 180).

Parameters

in	_model	Pointer to the Link (p. 398) from which to gather state info.
----	--------	--

10.18.2.3 gazebo::physics::CollisionState::CollisionState (const sdf::ElementPtr _sdf) [explicit]

Constructor.

Build a **CollisionState** (p. 190) from SDF data

Parameters

in	_sdf	SDF data to load a collision state from.
----	------	--

10.18.2.4 virtual gazebo::physics::CollisionState::~~CollisionState () [virtual]

Destructor.

10.18.3 Member Function Documentation

10.18.3.1 const math::Pose& gazebo::physics::CollisionState::GetPose () const

Get the **Collision** (p. 180) pose.

Returns

The pose of the **CollisionState** (p. 190)

10.18.3.2 `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.

10.18.3.3 `virtual void gazebo::physics::CollisionState::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Load **CollisionState** (p. 190) 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. 704).

10.18.3.4 `CollisionState gazebo::physics::CollisionState::operator+ (const CollisionState & _state) const`

Addition operator.

Parameters

<code>in</code>	<code>_pt</code>	A state to add.
-----------------	------------------	-----------------

Returns

The resulting state.

10.18.3.5 `CollisionState gazebo::physics::CollisionState::operator- (const CollisionState & _state) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_pt</code>	A state to subtract.
-----------------	------------------	----------------------

Returns

The resulting state.

10.18.3.6 CollisionState& gazebo::physics::CollisionState::operator= (const CollisionState & _state)

Assignment operator.

Parameters

in	<code>_state</code>	State (p. 702) value
----	---------------------	-----------------------------

Returns

Reference to this

10.18.4 Friends And Related Function Documentation

10.18.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::physics::CollisionState & _state) [friend]

Stream insertion operator.

Parameters

in	<code>_out</code>	output stream
in	<code>_state</code>	Collision (p. 180) state to output

Returns

the stream

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

- **CollisionState.hh**

10.19 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.*

10.19.1 Detailed Description

Defines a color.

10.19.2 Member Typedef Documentation

10.19.2.1 `typedef unsigned int gazebo::common::Color::ABGR`

10.19.2.2 `typedef unsigned int gazebo::common::Color::ARGB`

10.19.2.3 `typedef unsigned int gazebo::common::Color::BGRA`

10.19.2.4 `typedef unsigned int gazebo::common::Color::RGBA`

10.19.3 Constructor & Destructor Documentation

10.19.3.1 `gazebo::common::Color::Color ()`

Constructor.

10.19.3.2 `gazebo::common::Color::Color (float _r, float _g, float _b, float _a = 1.0)`

Constructor.

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)

10.19.3.3 `gazebo::common::Color::Color (const Color & _clr)`

Copy Constructor.

Parameters

<code>in</code>	<code>_clr</code>	Color (p. 193) to copy
-----------------	-------------------	-------------------------------

10.19.3.4 `virtual gazebo::common::Color::~~Color () [virtual]`

Destructor.

10.19.4 Member Function Documentation

10.19.4.1 **ABGR** `gazebo::common::Color::GetAsABGR () const`

Get as uint32 ABGR packed value.

Returns

the color

10.19.4.2 ARGB gazebo::common::Color::GetAsARGB () const

Get as uint32 ARGB packed value.

Returns

the color

10.19.4.3 BGRA gazebo::common::Color::GetAsBGRA () const

Get as uint32 BGRA packed value.

Returns

the color

10.19.4.4 math::Vector3 gazebo::common::Color::GetAsHSV () const

Get the color in HSV colorspace.

Returns

HSV values in a **math::Vector3** (p. 799) format

10.19.4.5 RGBA gazebo::common::Color::GetAsRGBA () const

Get as uint32 RGBA packed value.

Returns

the color

10.19.4.6 math::Vector3 gazebo::common::Color::GetAsYUV () const

Get the color in YUV colorspace.

Returns

the YUV color

10.19.4.7 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`

10.19.4.8 `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

10.19.4.9 `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

10.19.4.10 `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

10.19.4.11 `Color gazebo::common::Color::operator+ (const Color & _pt) const`

Addition operator (this + `_pt`)

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 193) to add
-----------------	------------------	------------------------------

Returns

The resulting color

10.19.4.12 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

10.19.4.13 const Color& gazebo::common::Color::operator+= (const Color & _pt)

Addition equal operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 193) to add
-----------------	------------------	------------------------------

Returns

The resulting color

10.19.4.14 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

10.19.4.15 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

10.19.4.16 `const Color& gazebo::common::Color::operator-= (const Color & _pt)`

Subtraction equal operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 193) to subtract
-----------------	------------------	-----------------------------------

Returns

The resulting color

10.19.4.17 `const Color gazebo::common::Color::operator/ (const Color & _pt) const`

Division operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 193) to divide by
-----------------	------------------	------------------------------------

Returns

The resulting color

10.19.4.18 `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

10.19.4.19 `const Color& gazebo::common::Color::operator/= (const Color & _pt)`

Division equal operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 193) to divide by
-----------------	------------------	------------------------------------

Returns

The resulting color

10.19.4.20 `Color& gazebo::common::Color::operator=(const Color & _pt)`

Equal operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 193) to copy
-----------------	------------------	-------------------------------

Returns

Reference to this color

10.19.4.21 `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`

10.19.4.22 `float gazebo::common::Color::operator[](unsigned int _index)`

Array index operator.

Parameters

<code>in</code>	<code>_index</code>	Color (p. 193) component index(0=red, 1=green, 2=blue)
-----------------	---------------------	---

Returns

r, g, b, or a when `_index` is 0, 1, 2 or 3

10.19.4.23 `void gazebo::common::Color::Reset ()`

Reset the color to default values.

10.19.4.24 void gazebo::common::Color::Set (float *_r* = 1, float *_g* = 1, float *_b* = 1, float *_a* = 1)

Set the contents of the vector.

Parameters

in	<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)

10.19.4.25 void gazebo::common::Color::SetFromABGR (const ABGR *_v*)

Set from uint32 ABGR packed value.

Parameters

in	<i>_v</i>	the new color
----	-----------	---------------

10.19.4.26 void gazebo::common::Color::SetFromARGB (const ARGB *_v*)

Set from uint32 ARGB packed value.

Parameters

in	<i>_v</i>	the new color
----	-----------	---------------

10.19.4.27 void gazebo::common::Color::SetFromBGRA (const BGRA *_v*)

Set from uint32 BGRA packed value.

Parameters

in	<i>_v</i>	the new color
----	-----------	---------------

10.19.4.28 void gazebo::common::Color::SetFromHSV (float *_h*, float *_s*, float *_v*)

Set a color based on HSV values.

Parameters

in	<i>_h</i>	Hue(0..360)
in	<i>_s</i>	Saturation(0..1)
in	<i>_v</i>	Value(0..1)

10.19.4.29 void gazebo::common::Color::SetFromRGBA (const RGBA *_v*)

Set from uint32 RGBA packed value.

Parameters

in	<code>_v</code>	the new color
----	-----------------	---------------

10.19.4.30 void gazebo::common::Color::SetFromYUV (float *_y*, float *_u*, float *_v*)

Set from yuv.

Parameters

in	<code>_y</code>	value
in	<code>_u</code>	value
in	<code>_v</code>	value

10.19.5 Friends And Related Function Documentation

10.19.5.1 std::ostream& operator<< (std::ostream & *_out*, const Color & *_pt*) [friend]

Stream insertion operator.

Parameters

in	<code>_out</code>	the output stream
in	<code>_pt</code>	the color

Returns

the output stream

10.19.5.2 std::istream& operator>> (std::istream & *_in*, Color & *_pt*) [friend]

Stream insertion operator.

Parameters

in	<code>_in</code>	the input stream
in	<code>_pt</code>	

10.19.6 Member Data Documentation

10.19.6.1 float gazebo::common::Color::a

10.19.6.2 float gazebo::common::Color::b

10.19.6.3 const Color gazebo::common::Color::Black [static]

(0, 0, 0)

10.19.6.4 `const Color gazebo::common::Color::Blue` `[static]`

(0, 0, 1)

10.19.6.5 `float gazebo::common::Color::g`

10.19.6.6 `const Color gazebo::common::Color::Green` `[static]`

(0, 1, 0)

10.19.6.7 `const Color gazebo::common::Color::Purple` `[static]`

(1, 0, 1)

10.19.6.8 `float gazebo::common::Color::r`

10.19.6.9 `const Color gazebo::common::Color::Red` `[static]`

(1, 0, 0)

10.19.6.10 `const Color gazebo::common::Color::White` `[static]`

(1, 1, 1)

10.19.6.11 `const Color gazebo::common::Color::Yellow` `[static]`

(1, 1, 0)

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

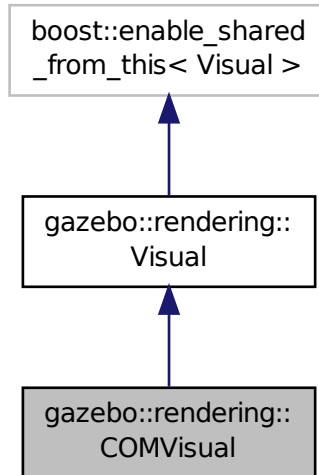
- **Color.hh**

10.20 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. 828) from an SDF pointer.*
- virtual void **Load** (ConstLinkPtr &_msg)
Load from a message.

Additional Inherited Members

10.20.1 Detailed Description

Basic Center of Mass visualization.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 gazebo::rendering::COMVisual::COMVisual (const std::string & _name, VisualPtr _vis)

Constructor.

Parameters

in	<code>_name</code>	Name of the Visual (p. 828)
in	<code>_vis</code>	Parent Visual (p. 828)

10.20.2.2 `virtual gazebo::rendering::COMVisual::~~COMVisual () [virtual]`

Destructor.

10.20.3 Member Function Documentation

10.20.3.1 `virtual void gazebo::rendering::COMVisual::Load (sdf::ElementPtr elem) [virtual]`

Load the **Visual** (p. 828) from an SDF pointer.

Parameters

in	<code>_elem</code>	SDF Element pointer
----	--------------------	---------------------

10.20.3.2 `virtual void gazebo::rendering::COMVisual::Load (ConstLinkPtr & msg) [virtual]`

Load from a message.

Parameters

in	<code>_msg</code>	Pointer to the message
----	-------------------	------------------------

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

- **COMVisual.hh**

10.21 gazebo::event::Connection Class Reference

A class that encapsulates a connection.

```
#include <Event.hh>
```

Public Member Functions

- **Connection** ()
Constructor.
- **Connection** (**Event** **e*, int *i*)
Constructor.
- **~Connection** ()
Destructor.
- int **GetId** () const
Get the id of this connection.

10.21.1 Detailed Description

A class that encapsulates a connection.

10.21.2 Constructor & Destructor Documentation

10.21.2.1 gazebo::event::Connection::Connection () [inline]

Constructor.

10.21.2.2 gazebo::event::Connection::Connection (Event * _e, int _i)

Constructor.

Parameters

in	<code>_e</code>	Event (p. 277) pointer to connect with
in	<code>_i</code>	Unique id

10.21.2.3 gazebo::event::Connection::~~Connection ()

Destructor.

10.21.3 Member Function Documentation

10.21.3.1 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:

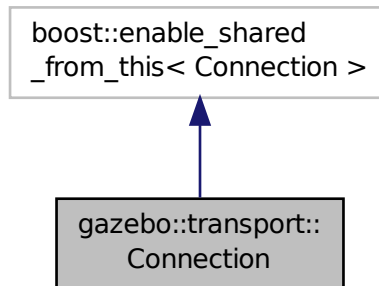
- **Event.hh**

10.22 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.

10.22.1 Detailed Description

Single TCP/IP connection manager.

10.22.2 Member Typedef Documentation

10.22.2.1 typedef boost::function<void(const ConnectionPtr&)> gazebo::transport::Connection::AcceptCallback

The signature of a connection accept callback.

10.22.2.2 typedef boost::function<void(const std::string &_data)> gazebo::transport::Connection::ReadCallback

The signature of a connection read callback.

10.22.3 Constructor & Destructor Documentation

10.22.3.1 gazebo::transport::Connection::Connection ()

Constructor.

10.22.3.2 virtual gazebo::transport::Connection::~~Connection () [virtual]

Destructor.

10.22.4 Member Function Documentation

10.22.4.1 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().

10.22.4.2 void gazebo::transport::Connection::Cancel ()

Cancel all async operations on an open socket.

10.22.4.3 bool gazebo::transport::Connection::Connect (const std::string & *_host*, unsigned int *_port*)

Connect to a remote host.

Parameters

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

Returns

true if connection succeeded, false otherwise

10.22.4.4 event::ConnectionPtr gazebo::transport::Connection::ConnectToShutdown (boost::function< void()> *_subscriber*) [inline]

Register a function to be called when the connection is shut down.

Parameters

in	<i>_subscriber</i>	Function to be called
----	--------------------	-----------------------

Returns

Handle that can be used to unregister the function

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

10.22.4.5 void gazebo::transport::Connection::DisconnectShutdown (event::ConnectionPtr *_subscriber*) [inline]

Unregister a function to be called when the connection is shut down.

Parameters

in	<i>_subscriber</i>	Handle previously returned by ConnectToShutdown() (p. 210)
----	--------------------	---

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

10.22.4.6 void gazebo::transport::Connection::EnqueueMsg (const std::string & *_buffer*, bool *_force* = false)

Write data to the socket.

Parameters

in	<i>_buffer</i>	Data to write
in	<i>_force</i>	If true, block until the data has been written to the socket, otherwise just enqueue the data for asynchronous write

10.22.4.7 unsigned int gazebo::transport::Connection::GetId () const

Get the ID of the connection.

Returns

The connection's unique ID.

10.22.4.8 std::string gazebo::transport::Connection::GetLocalAddress () const

Get the local address of this connection.

Returns

The local address

10.22.4.9 std::string gazebo::transport::Connection::GetLocalHostname () const

Get the local hostname.

Returns

The local hostname

10.22.4.10 unsigned int gazebo::transport::Connection::GetLocalPort () const

Get the port of this connection.

Returns

The local port

10.22.4.11 `std::string gazebo::transport::Connection::GetLocalURI () const`

Get the local URI.

Returns

The local URI

10.22.4.12 `std::string gazebo::transport::Connection::GetRemoteAddress () const`

Get the remote address.

Returns

The remote address

10.22.4.13 `std::string gazebo::transport::Connection::GetRemoteHostname () const`

Get the remote hostname.

Returns

The remote hostname

10.22.4.14 `unsigned int gazebo::transport::Connection::GetRemotePort () const`

Get the remote port number.

Returns

The remote port

10.22.4.15 `std::string gazebo::transport::Connection::GetRemoteURI () const`

Get the remote URI.

Returns

The remote URI

10.22.4.16 `bool gazebo::transport::Connection::IsOpen () const`

Is the connection open?

Returns

true if the connection is open; false otherwise

Referenced by `AsyncRead()`.

10.22.4.17 void gazebo::transport::Connection::Listen (unsigned int *_port*, const AcceptCallback & *_acceptCB*)

Start a server that listens on a port.

Parameters

in	<i>_port</i>	The port to listen on
in	<i>_acceptCB</i>	The callback to invoke when a new connection has been accepted

10.22.4.18 void gazebo::transport::Connection::ProcessWriteQueue ()

Handle on-write callbacks.

10.22.4.19 bool gazebo::transport::Connection::Read (std::string & *_data*)

Read data from the socket.

Parameters

out	<i>_data</i>	Destination for data that is read
-----	--------------	-----------------------------------

Returns

true if data was successfully read, false otherwise

10.22.4.20 void gazebo::transport::Connection::Shutdown ()

Shutdown the socket.

10.22.4.21 void gazebo::transport::Connection::StartRead (const ReadCallback & *_cb*)

Start a thread that reads from the connection and passes new message to the ReadCallback.

Parameters

in	<i>_cb</i>	The callback to invoke when a new message is received
----	------------	---

10.22.4.22 void gazebo::transport::Connection::StopRead ()

Stop the read loop.

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

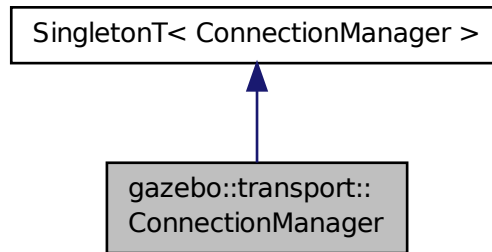
- **Connection.hh**

10.23 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 connecton 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

10.23.1 Detailed Description

Manager of connections.

10.23.2 Member Function Documentation

10.23.2.1 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

10.23.2.2 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)

10.23.2.3 void gazebo::transport::ConnectionManager::Fini ()

Finalize the connection manager.

10.23.2.4 void gazebo::transport::ConnectionManager::GetAllPublishers (std::list< msgs::Publish > & _publishers)

Explicitly update the publisher list.

Parameters

out	<i>_publishers</i>	The updated list of publishers is written here
-----	--------------------	--

10.23.2.5 void gazebo::transport::ConnectionManager::GetTopicNamespaces (std::list< std::string > & *_namespaces*)

Get all the topic namespaces.

Parameters

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

10.23.2.6 bool gazebo::transport::ConnectionManager::Init (const std::string & *_masterHost*, unsigned int *_masterPort*)

Initialize the connection manager.

Parameters

in	<i>_masterHost</i>	Host where the master is running
in	<i>_masterPort</i>	Port where the master is running

Returns

true if initialization succeeded, false otherwise

10.23.2.7 bool gazebo::transport::ConnectionManager::IsRunning () const

Is the manager running?

Returns

true if running, false otherwise

10.23.2.8 void gazebo::transport::ConnectionManager::RegisterTopicNamespace (const std::string & *_name*)

Register a new topic namespace.

Parameters

in	<i>_name</i>	The name of the topic namespace to be registered
----	--------------	--

10.23.2.9 void gazebo::transport::ConnectionManager::RemoveConnection (ConnectionPtr & *_conn*)

Remove a connection from the manager.

Parameters

in	<i>_conn</i>	The connection to be removed
----	--------------	------------------------------

10.23.2.10 void gazebo::transport::ConnectionManager::Run ()

Run the connection manager loop.

Does not return until stopped.

10.23.2.11 void gazebo::transport::ConnectionManager::RunUpdate ()

Run the manager update loop once.

10.23.2.12 void gazebo::transport::ConnectionManager::Stop ()

Stop the connecton manager.

10.23.2.13 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

10.23.2.14 void gazebo::transport::ConnectionManager::Unadvertise (const std::string & *_topic*)

Unadvertise a topic.

Parameters

in	<i>_topic</i>	The topic to unadvertise
----	---------------	--------------------------

10.23.2.15 void gazebo::transport::ConnectionManager::Unsubscribe (const msgs::Subscribe & *_sub*)

Unsubscribe from a topic.

Parameters

in	<i>_sub</i>	A subscription object
----	-------------	-----------------------

10.23.2.16 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

10.23.3 Member Data Documentation

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

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

- **ConnectionManager.hh**

10.24 gazebo::common::Console Class Reference

Message, error, warning functionality.

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

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 Load ()`
Load the message parameters.
- `void SetQuiet (bool _q)`
Set quiet output.

Static Public Member Functions

- `static Console * Instance ()`
Return an instance to this class.

10.24.1 Detailed Description

Message, error, warning functionality.

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

- **Console.hh**

10.25 gazebo::physics::Contact Class Reference

A contact between two collisions.

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

Public Member Functions

- **Contact** ()
Constructor.
- **Contact** (const **Contact** &_contact)
Copy constructor.
- virtual ~**Contact** ()
Destructor.
- **Contact Clone** () const **GAZEBO_DEPRECATED**
Deprecated.
- 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.

10.25.1 Detailed Description

A contact between two collisions.

Each contact can consist of a number of contact points

10.25.2 Constructor & Destructor Documentation

10.25.2.1 gazebo::physics::Contact::Contact ()

Constructor.

10.25.2.2 gazebo::physics::Contact::Contact (const Contact & *_contact*)

Copy constructor.

Parameters

in	<i>_contact</i>	Contact (p. 218) to copy.
----	-----------------	----------------------------------

10.25.2.3 virtual gazebo::physics::Contact::~~Contact () [virtual]

Destructor.

10.25.3 Member Function Documentation

10.25.3.1 Contact gazebo::physics::Contact::Clone () const

Deprecated.

10.25.3.2 std::string gazebo::physics::Contact::DebugString () const

Produce a debug string.

Returns

A string that contains the values of the contact.

10.25.3.3 void gazebo::physics::Contact::FillMsg (msgs::Contact & *_msg*) const

Populate a msgs::Contact with data from this.

Parameters

out	<i>_msg</i>	Contact (p. 218) message the will hold the data.
-----	-------------	---

10.25.3.4 Contact& gazebo::physics::Contact::operator= (const Contact & *_contact*)

Operator =.

Parameters

in	<i>_contact</i>	Contact (p. 218) to copy.
----	-----------------	----------------------------------

Returns

Reference to this contact

10.25.3.5 Contact& gazebo::physics::Contact::operator= (const msgs::Contact & *_contact*)

Operator =.

Parameters

<i>in</i>	<i>_contact</i>	msgs::Contact to copy.
-----------	-----------------	------------------------

Returns

Reference to this contact

10.25.3.6 void gazebo::physics::Contact::Reset ()

Reset to default values.

10.25.4 Member Data Documentation**10.25.4.1 std::string gazebo::physics::Contact::collision1**

Name of the first collision object.

10.25.4.2 std::string gazebo::physics::Contact::collision2

Name of the second collision object.

10.25.4.3 int gazebo::physics::Contact::count

Length of all the arrays.

10.25.4.4 double gazebo::physics::Contact::depths[32]

Array of contact depths.

10.25.4.5 math::Vector3 gazebo::physics::Contact::normals[32]

Array of force normals.

10.25.4.6 math::Vector3 gazebo::physics::Contact::positions[32]

Array of force positions.

10.25.4.7 `common::Time` `gazebo::physics::Contact::time`

Time at which the contact occurred.

10.25.4.8 `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`

10.26 `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. 222).
- `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.

10.26.1 Detailed Description

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

10.26.2 Constructor & Destructor Documentation

10.26.2.1 gazebo::physics::ContactManager::ContactManager ()

Constructor.

10.26.2.2 virtual gazebo::physics::ContactManager::~~ContactManager () [virtual]

Destructor.

10.26.3 Member Function Documentation

10.26.3.1 void gazebo::physics::ContactManager::Clear ()

Clear all stored contacts.

10.26.3.2 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. 223).

Parameters

<i>in</i>	<i>_index</i>	Index of the Contact (p. 218) to return.
-----------	---------------	---

Returns

Pointer to a contact, NULL If index is invalid.

10.26.3.3 unsigned int gazebo::physics::ContactManager::GetContactCount () const

Return the number of valid contacts.

10.26.3.4 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. 223)

Returns

Vector of contact pointers.

10.26.3.5 void gazebo::physics::ContactManager::Init (WorldPtr *_world*)

Initialize the **ContactManager** (p. 222).

This is required in order to publish contact messages via the **ContactManager::PublishContacts** (p. 224) method.

Parameters

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

10.26.3.6 `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.

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

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

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

Set the contact count to zero.

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

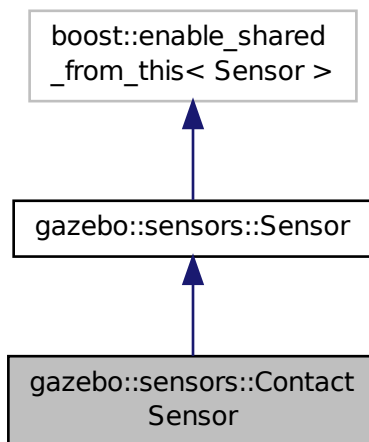
- **ContactManager.hh**

10.27 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.
- **physics::Contact GetCollisionContact** (const std::string &_collisionName, unsigned int _index) const **GAZEBO_DEPRECATED**
*Deprecated. Use **ContactSensor::GetContacts** (p. 227).*
- 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.
- 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

10.27.1 Detailed Description

Contact sensor.

This sensor detects and reports contacts between objects

10.27.2 Constructor & Destructor Documentation

10.27.2.1 gazebo::sensors::ContactSensor::ContactSensor ()

Constructor.

10.27.2.2 virtual gazebo::sensors::ContactSensor::~~ContactSensor () [virtual]

Destructor.

10.27.3 Member Function Documentation

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

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 655).

10.27.3.2 physics::Contact gazebo::sensors::ContactSensor::GetCollisionContact (const std::string & _collisionName, unsigned int _index) const

Deprecated. Use **ContactSensor::GetContacts** (p. 227).

10.27.3.3 unsigned int gazebo::sensors::ContactSensor::GetCollisionContactCount (const std::string & _collisionName) const

Return the number of contacts for an observed collision.

Parameters

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

Returns

The collision contact count.

10.27.3.4 unsigned int gazebo::sensors::ContactSensor::GetCollisionCount () const

Get the number of collisions that the sensor is observing.

Returns

Number of collisions.

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

Get a collision name at index *_index*.

Parameters

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

Returns

name of collision.

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

Gets contacts of a collision.

Parameters

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

Returns

Container of contacts

10.27.3.7 virtual void gazebo::sensors::ContactSensor::Init () [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 657).

10.27.3.8 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. 657).

10.27.3.9 `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 Sensor (p. 652) parameters
<code>in</code>	<code>_worldName</code>	Name of world to load from

Reimplemented from **`gazebo::sensors::Sensor`** (p. 657).

10.27.3.10 `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. 658).

10.27.3.11 `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. 659).

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

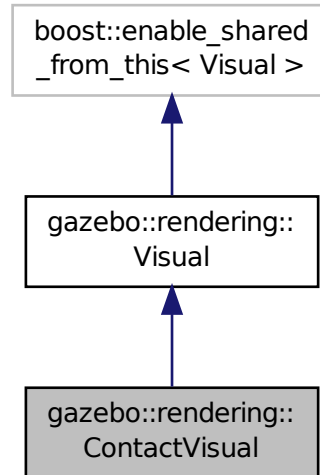
- **`ContactSensor.hh`**

10.28 `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

10.28.1 Detailed Description

Contact visualization.

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

10.28.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	<code>_name</code>	Name of the ContactVisual (p. 228)
in	<code>_vis</code>	Pointer the parent Visual (p. 828)
in	<code>_topicName</code>	Name of the topic which publishes the contact information.

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

Destructor.

10.28.3 Member Function Documentation

10.28.3.1 `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**

10.29 gazebo::rendering::Conversions Class Reference

Conversions (p. 230) **Conversions.hh** (p. 906) **rendering/Conversions.hh** (p. 906).

```
#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. 98) 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. 98) quaternion.*
- static `math::Quaternion Convert (const Ogre::Quaternion &_v)`
***Ogre** (p. 98) quaternion to Gazebo quaternion.*

10.29.1 Detailed Description

Conversions (p. 230) **Conversions.hh** (p. 906) **rendering/Conversions.hh** (p. 906).

A set of utility function to convert between Gazebo and **Ogre** (p. 98) data types

10.29.2 Member Function Documentation

10.29.2.1 `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. 98) color value

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

Return the equivalent gazebo color.

Parameters

in	_clr	Ogre (p. 98) color to convert
----	------	--------------------------------------

Returns

Gazebo color value

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

return **Ogre** (p. 98) Vector from Gazebo Vector3

Parameters

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

Returns

Ogre (p. 98) vector

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

return gazebo Vector from ogre Vector3

Parameters

in	_v	Ogre (p. 98) vector
----	----	----------------------------

Returns

Gazebo vector

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

Gazebo quaternion to **Ogre** (p. 98) quaternion.

Parameters

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

Returns

Ogre (p. 98) quaternion

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

Ogre (p. 98) quaternion to Gazebo quaternion.

Parameters

<code>in</code>	<code>_v</code>	Ogre (p. 98) quaternion return Gazebo quaternion
-----------------	-----------------	---

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

- **Conversions.hh**

10.30 sdf::Converter Class Reference

Convert from one version of **SDF** (p. 649) to another.

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

Static Public Member Functions

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

10.30.1 Detailed Description

Convert from one version of **SDF** (p. 649) to another.

10.30.2 Member Function Documentation

10.30.2.1 `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:

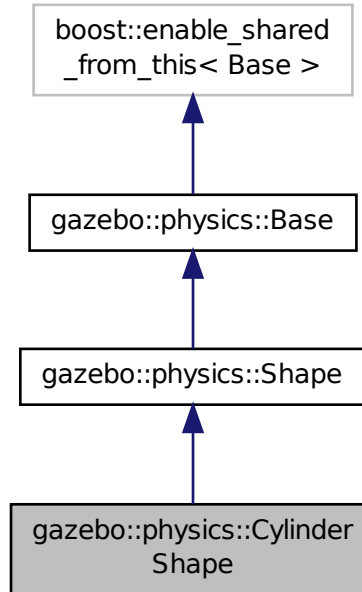
- **Converter.hh**

10.31 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.
- virtual void **GetInertial** (double _mass, **InertialPtr** _inertial) const **GAZEBO_DEPRECATED**
Deprecated.
- double **GetLength** () const
Get length.
- virtual double **GetMass** (double _density) const **GAZEBO_DEPRECATED**
Deprecated.
- 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

10.31.1 Detailed Description

Cylinder collision.

10.31.2 Constructor & Destructor Documentation

10.31.2.1 gazebo::physics::CylinderShape::CylinderShape (CollisionPtr _parent) [explicit]

Constructor.

Parameters

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

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

Destructor.

10.31.3 Member Function Documentation

10.31.3.1 void gazebo::physics::CylinderShape::FillMsg (msgs::Geometry & _msg) [virtual]

Fill in the values for a geometry message.

Parameters

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

Implements **gazebo::physics::Shape** (p. 669).

10.31.3.2 virtual void gazebo::physics::CylinderShape::GetInertial (double _mass, InertialPtr _inertial) const [virtual]

Deprecated.

Reimplemented from **gazebo::physics::Shape** (p. 670).

10.31.3.3 `double gazebo::physics::CylinderShape::GetLength () const`

Get length.

Returns

The cylinder length.

10.31.3.4 `virtual double gazebo::physics::CylinderShape::GetMass (double _density) const` [virtual]

Deprecated.

Reimplemented from **gazebo::physics::Shape** (p. 670).

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

Get radius.

Returns

The cylinder radius.

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

Initialize the cylinder.

Implements **gazebo::physics::Shape** (p. 670).

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

Update values based on a message.

Parameters

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

Implements **gazebo::physics::Shape** (p. 670).

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

Set length.

Parameters

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

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

Set radius.

Parameters

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

10.31.3.10 virtual void gazebo::physics::CylinderShape::SetSize (double *_radius*, double *_length*) [virtual]

Set the size of the cylinder.

Parameters

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

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

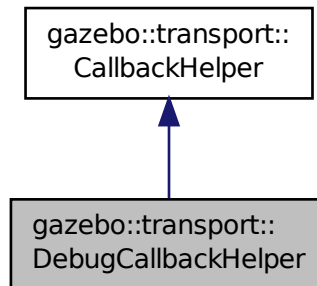
- `CylinderShape.hh`

10.32 gazebo::transport::DebugCallbackHelper Class Reference

CallbackHelper (p. 144) subclass with debug facilities.

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

Inheritance diagram for gazebo::transport::DebugCallbackHelper:



Public Member Functions

- **DebugCallbackHelper** (const boost::function< void(ConstGzStringPtr &)> &*_cb*)
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

10.32.1 Detailed Description

CallbackHelper (p. 144) subclass with debug facilities.

10.32.2 Constructor & Destructor Documentation

10.32.2.1 `gazebo::transport::DebugCallbackHelper::DebugCallbackHelper (const boost::function< void(ConstGzStringPtr &)> & _cb) [inline]`

Constructor.

Parameters

<code>in</code>	<code>_cb</code>	boost function to call on incoming messages
-----------------	------------------	---

10.32.3 Member Function Documentation

10.32.3.1 `std::string gazebo::transport::DebugCallbackHelper::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. 146).

10.32.3.2 `virtual bool gazebo::transport::DebugCallbackHelper::HandleData (const std::string & _newdata) [inline],[virtual]`

Process new incoming data.

Parameters

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

Returns

true if successfully processed; false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 146).

10.32.3.3 `virtual bool gazebo::transport::DebugCallbackHelper::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. 146).

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

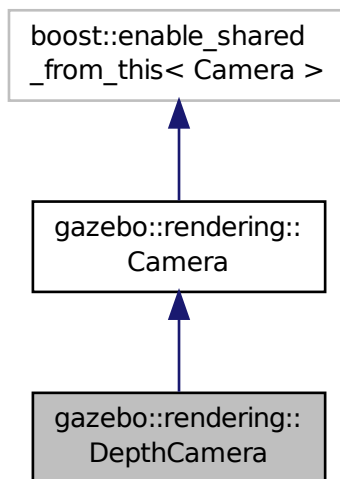
- **CallbackHelper.hh**

10.33 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

10.33.1 Detailed Description

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

10.33.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	<i>_namePrefix</i>	Unique prefix name for the camera.
in	<i>_scene</i>	Scene (p. 632) that will contain the camera
in	<i>_autoRender</i>	Almost everyone should leave this as true.

10.33.2.2 `virtual gazebo::rendering::DepthCamera::~~DepthCamera () [virtual]`

Destructor.

10.33.3 Member Function Documentation

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

Connect a to the new depth image signal.

Parameters

<code>in</code>	<code>_subscriber</code>	Subscriber callback function
-----------------	--------------------------	------------------------------

Returns

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

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

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

Connect a to the new rgb point cloud signal.

Parameters

<code>in</code>	<code>_subscriber</code>	Subscriber callback function
-----------------	--------------------------	------------------------------

Returns

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

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

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

Create a texture which will hold the depth data.

Parameters

<code>in</code>	<code>_textureName</code>	Name of the texture to create
-----------------	---------------------------	-------------------------------

10.33.3.4 `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().

10.33.3.5 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().

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

Finalize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 157).

10.33.3.7 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

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

Initialize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 164).

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

Load the camera with a set of parmeters.

Parameters

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

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

Load the camera with default parmeters.

Reimplemented from **gazebo::rendering::Camera** (p. 165).

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

Render the camera.

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

10.33.3.12 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
----	----------------	------------------------------

10.33.4 Member Data Documentation

10.33.4.1 Ogre::RenderTarget* gazebo::rendering::DepthCamera::depthTarget [protected]

Pointer to the depth target.

10.33.4.2 Ogre::Texture* gazebo::rendering::DepthCamera::depthTexture [protected]

Pointer to the depth texture.

10.33.4.3 Ogre::Viewport* gazebo::rendering::DepthCamera::depthViewport [protected]

Pointer to the depth viewport.

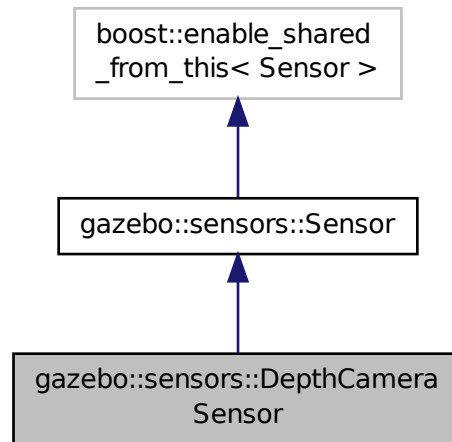
The documentation for this class was generated from the following file:

- **DepthCamera.hh**

10.34 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. 238).*
- 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

10.34.1 Constructor & Destructor Documentation

10.34.1.1 gazebo::sensors::DepthCameraSensor::DepthCameraSensor ()

Constructor.

10.34.1.2 virtual gazebo::sensors::DepthCameraSensor::~~DepthCameraSensor () [virtual]

Destructor.

10.34.2 Member Function Documentation

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

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 655).

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

Returns a pointer to the **rendering::DepthCamera** (p. 238).

Returns

Depth Camera pointer

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

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 657).

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

Load the sensor with SDF parameters.

Parameters

in	<i>_sdf</i>	SDF Sensor (p. 652) parameters
in	<i>_worldName</i>	Name of world to load from

10.34.2.5 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. 658).

10.34.2.6 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

10.34.2.7 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. 658).

10.34.2.8 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. 658).

10.34.2.9 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. 659).

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

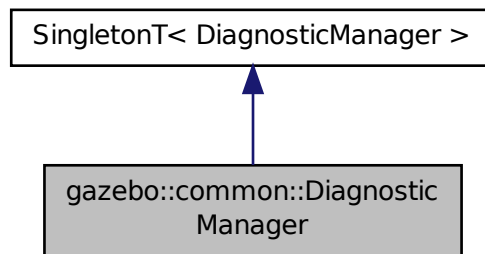
- `DepthCameraSensor.hh`

10.35 gazebo::common::DiagnosticManager Class Reference

A 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 diagnostic timer has started.
- void **TimerStop** (**DiagnosticTimer** *_timer)
A diagnostic timer has stopped.

Additional Inherited Members

10.35.1 Detailed Description

A diagnostic manager class.

10.35.2 Member Function Documentation

10.35.2.1 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 pointer to the new diagnostic timer

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

Get whether the timers are enabled.

Returns

TRue if the timers are enabled

10.35.2.3 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

10.35.2.4 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. 732) of the specified timer

10.35.2.5 Time gazebo::common::DiagnosticManager::GetTime (const std::string & *_label*) const

Get a time based on a label.

Parameters

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

Returns

Time (p. 732) of the specified timer

10.35.2.6 int gazebo::common::DiagnosticManager::GetTimerCount () const

Get the number of timers.

Returns

The number of timers

10.35.2.7 void gazebo::common::DiagnosticManager::SetEnabled (bool *_e*) [inline]

Set whether timers are enabled.

Parameters

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

10.35.2.8 void gazebo::common::DiagnosticManager::TimerStart (DiagnosticTimer * *_timer*)

A diagnostic timer has started.

Parameters

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

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

10.35.2.9 void gazebo::common::DiagnosticManager::TimerStop (DiagnosticTimer * *_timer*)

A diagnostic timer has stopped.

Parameters

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

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

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

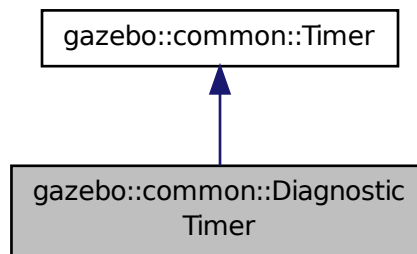
- **Diagnostics.hh**

10.36 gazebo::common::DiagnosticTimer Class Reference

A 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.

10.36.1 Detailed Description

A timer designed for diagnostics.

10.36.2 Constructor & Destructor Documentation

10.36.2.1 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().

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

Destructor.

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

10.36.3 Member Function Documentation

10.36.3.1 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:

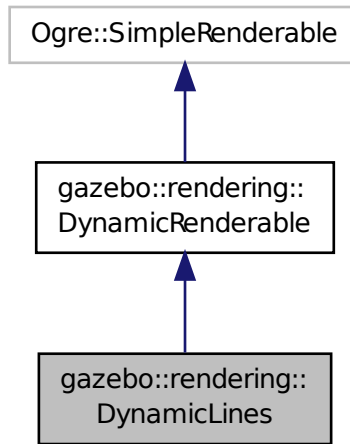
- **Diagnostics.hh**

10.37 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. 98)'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. 254), creates a simple vertex-only decl.*
- virtual void **FillHardwareBuffers** ()
*Implementation **DynamicRenderable** (p. 254), pushes point list out to hardware memory.*

Additional Inherited Members

10.37.1 Detailed Description

Class for drawing lines that can change.

10.37.2 Constructor & Destructor Documentation

10.37.2.1 gazebo::rendering::DynamicLines::DynamicLines (RenderOpType *_opType* = RENDERING_LINE_STRIP)

Constructor.

Parameters

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

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

Destructor.

10.37.3 Member Function Documentation

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

Add a point to the point list.

Parameters

in	<i>pt</i>	math::Vector3 (p. 799) point
----	-----------	-------------------------------------

10.37.3.2 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.

10.37.3.3 void gazebo::rendering::DynamicLines::Clear ()

Remove all points from the point list.

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

Implementation **DynamicRenderable** (p. 254), creates a simple vertex-only decl.

Implements **gazebo::rendering::DynamicRenderable** (p. 256).

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

Implementation **DynamicRenderable** (p. 254), pushes point list out to hardware memory.

Implements **gazebo::rendering::DynamicRenderable** (p. 256).

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

Get type of movable.

Returns

This returns "gazebo::dynamiclines"

10.37.3.7 virtual const Ogre::String& gazebo::rendering::DynamicLines::getMovableType () const [virtual]

Overridden function from **Ogre** (p. 98)'s base class.

Returns

Returns "gazebo::ogredynamiclines"

10.37.3.8 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. 799) value of the point

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

Return the total number of points in the point list.

Returns

Number of points

10.37.3.10 `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. 799) value to set the point to

10.37.3.11 `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:

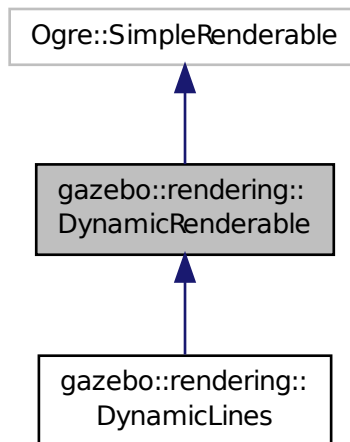
- `DynamicLines.hh`

10.38 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.

10.38.1 Detailed Description

Abstract base class providing mechanisms for dynamically growing hardware buffers.

10.38.2 Constructor & Destructor Documentation

10.38.2.1 `gazebo::rendering::DynamicRenderable::DynamicRenderable ()`

Constructor.

10.38.2.2 `virtual gazebo::rendering::DynamicRenderable::~~DynamicRenderable () [virtual]`

Virtual destructor.

10.38.3 Member Function Documentation

10.38.3.1 `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. 253).

10.38.3.2 `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. 253).

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

Implementation of `Ogre::SimpleRenderable`.

Returns

The bounding radius

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

Get the render operation type.

Returns

The render operation type.

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

Implementation of `Ogre::SimpleRenderable`.

Parameters

<code>in</code>	<code>_cam</code>	Pointer to the Ogre (p. 98) camera that views the renderable.
-----------------	-------------------	--

Returns

The squared depth in the **Camera** (p. 149)'s view

10.38.3.6 `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. 256).

Parameters

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

10.38.3.7 `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

in	<code>_vertexCount</code>	The number of vertices the buffer must hold.
in	<code>_indexCount</code>	The number of indices the buffer must hold. This parameter is ignored if not using indices.

10.38.3.8 void gazebo::rendering::DynamicRenderable::SetOperationType (RenderOpType *_opType*)

Set the render operation type.

Parameters

in	<code>_opType</code>	The type of render operation to perform.
----	----------------------	--

10.38.4 Member Data Documentation

10.38.4.1 size_t gazebo::rendering::DynamicRenderable::indexBufferCapacity [protected]

Maximum capacity of the currently allocated index buffer.

10.38.4.2 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:

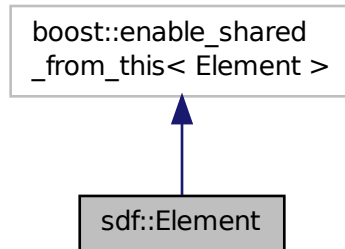
- **DynamicRenderable.hh**

10.39 sdf::Element Class Reference

SDF (p. 649) **Element** (p. 258) 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** ()
- **Element * Clone** () const
- void **Copy** (const **ElementPtr** _elem)
 - Copy values from an **Element** (p. 258).*
- **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. 650).*

- void **PrintDocRightPane** (std::string &_html, int _spacing)

*Helper function for **SDF::PrintDoc** (p. 650).*

- void **PrintValues** (std::string _prefix)
- void **PrintWiki** (std::string _prefix)
- 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** ()

10.39.1 Detailed Description

SDF (p. 649) **Element** (p. 258) class.

10.39.2 Constructor & Destructor Documentation

10.39.2.1 sdf::Element::Element ()

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

10.39.3 Member Function Documentation

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

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

10.39.3.3 void sdf::Element::AddElementDescription (ElementPtr _elem)

Add a new element description.

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

10.39.3.5 void sdf::Element::ClearElements ()

10.39.3.6 Element* sdf::Element::Clone () const

10.39.3.7 void sdf::Element::Copy (const ElementPtr _elem)

Copy values from an **Element** (p. 258).

10.39.3.8 ParamPtr sdf::Element::GetAttribute (const std::string &_key)

Get the param of an attribute.

Parameters

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

10.39.3.9 `ParamPtr sdf::Element::GetAttribute (unsigned int _index) const`

Get an attribute using an index.

10.39.3.10 `unsigned int sdf::Element::GetAttributeCount () const`

Get the number of attributes.

10.39.3.11 `bool sdf::Element::GetAttributeSet (const std::string & _key)`

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

10.39.3.12 `bool sdf::Element::GetCopyChildren () const`

10.39.3.13 `std::string sdf::Element::GetDescription () const`

Get a text description of the element.

10.39.3.14 `ElementPtr sdf::Element::GetElement (const std::string & _name) const`

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

10.39.3.15 `ElementPtr sdf::Element::GetElement (const std::string & _name)`

10.39.3.16 `ElementPtr sdf::Element::GetElementDescription (unsigned int _index) const`

Get an element description using an index.

10.39.3.17 `ElementPtr sdf::Element::GetElementDescription (const std::string & _key) const`

Get an element descriptio using a key.

10.39.3.18 `unsigned int sdf::Element::GetElementDescriptionCount () const`

Get the number of element descriptions.

10.39.3.19 `ElementPtr sdf::Element::GetElementImpl (const std::string & _name) const`

10.39.3.20 `ElementPtr sdf::Element::GetFirstElement () const`

10.39.3.21 `std::string sdf::Element::GetInclude () const`

10.39.3.22 `const std::string& sdf::Element::GetName () const`

10.39.3.23 `ElementPtr sdf::Element::GetNextElement (const std::string & _name = " ") const`

10.39.3.24 `ElementPtr sdf::Element::GetParent () const`

10.39.3.25 `const std::string& sdf::Element::GetRequired () const`

10.39.3.26 `ParamPtr sdf::Element::GetValue ()`

Get the param of the elements value.

10.39.3.27 `bool sdf::Element::GetValueBool (const std::string & _key = " ")`

10.39.3.28 `char sdf::Element::GetValueChar (const std::string & _key = " ")`

10.39.3.29 `gazebo::common::Color sdf::Element::GetValueColor (const std::string & _key = " ")`

10.39.3.30 `double sdf::Element::GetValueDouble (const std::string & _key = " ")`

Referenced by `gazebo::physics::ScrewJoint< T >::Load()`.

10.39.3.31 `float sdf::Element::GetValueFloat (const std::string & _key = " ")`

10.39.3.32 `int sdf::Element::GetValueInt (const std::string & _key = " ")`

10.39.3.33 `gazebo::math::Pose sdf::Element::GetValuePose (const std::string & _key = " ")`

10.39.3.34 `gazebo::math::Quaternion sdf::Element::GetValueQuaternion (const std::string & _key = " ")`

10.39.3.35 `std::string sdf::Element::GetValueString (const std::string & _key = " ")`

10.39.3.36 `gazebo::common::Time sdf::Element::GetValueTime (const std::string & _key = " ")`

10.39.3.37 `unsigned int sdf::Element::GetValueUInt (const std::string & _key = " ")`

10.39.3.38 `gazebo::math::Vector2d sdf::Element::GetValueVector2d (const std::string & _key = " ")`

10.39.3.39 `gazebo::math::Vector3 sdf::Element::GetValueVector3 (const std::string & _key = " ")`

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

10.39.3.40 `bool sdf::Element::HasAttribute (const std::string & _key)`

10.39.3.41 `bool sdf::Element::HasElement (const std::string & _name) const`

Referenced by `gazebo::physics::ScrewJoint< T >::Load()`.

10.39.3.42 `bool sdf::Element::HasElementDescription (const std::string & _name)`

Return true if an element description exists.

10.39.3.43 `void sdf::Element::InsertElement (ElementPtr _elem)`

10.39.3.44 `void sdf::Element::PrintDescription (std::string _prefix)`

10.39.3.45 `void sdf::Element::PrintDocLeftPane (std::string & _html, int _spacing, int & _index)`

Helper function for **SDF::PrintDoc** (p. 650).

This generates the **SDF** (p. 649) 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.

10.39.3.46 `void sdf::Element::PrintDocRightPane (std::string & _html, int _spacing)`

Helper function for **SDF::PrintDoc** (p. 650).

This generates the **SDF** (p. 649) html documentation.

Parameters

out	<code>_html</code>	Accumulated HTML for output
in	<code>_spacing</code>	Amount of spacing for this element.

10.39.3.47 `void sdf::Element::PrintValues (std::string _prefix)`

10.39.3.48 `void sdf::Element::PrintWiki (std::string _prefix)`

10.39.3.49 `void sdf::Element::Reset ()`

10.39.3.50 `bool sdf::Element::Set (const bool & _value)`

10.39.3.51 `bool sdf::Element::Set (const int & _value)`

10.39.3.52 `bool sdf::Element::Set (const unsigned int & _value)`

10.39.3.53 `bool sdf::Element::Set (const float & _value)`

10.39.3.54 `bool sdf::Element::Set (const double & _value)`

10.39.3.55 `bool sdf::Element::Set (const char & _value)`

10.39.3.56 `bool sdf::Element::Set (const std::string & _value)`

- 10.39.3.57 `bool sdf::Element::Set (const char * _value)`
- 10.39.3.58 `bool sdf::Element::Set (const gazebo::math::Vector3 & _value)`
- 10.39.3.59 `bool sdf::Element::Set (const gazebo::math::Vector2i & _value)`
- 10.39.3.60 `bool sdf::Element::Set (const gazebo::math::Vector2d & _value)`
- 10.39.3.61 `bool sdf::Element::Set (const gazebo::math::Quaternion & _value)`
- 10.39.3.62 `bool sdf::Element::Set (const gazebo::math::Pose & _value)`
- 10.39.3.63 `bool sdf::Element::Set (const gazebo::common::Color & _value)`
- 10.39.3.64 `bool sdf::Element::Set (const gazebo::common::Time & _value)`
- 10.39.3.65 `void sdf::Element::SetCopyChildren (bool _value)`
- 10.39.3.66 `void sdf::Element::SetDescription (const std::string & _desc)`

Set a text description for the element.

- 10.39.3.67 `void sdf::Element::SetInclude (const std::string & _filename)`
- 10.39.3.68 `void sdf::Element::SetName (const std::string & _name)`
- 10.39.3.69 `void sdf::Element::SetParent (const ElementPtr _parent)`
- 10.39.3.70 `void sdf::Element::SetRequired (const std::string & _req)`
- 10.39.3.71 `std::string sdf::Element::ToString (const std::string & _prefix) const`
- 10.39.3.72 `void sdf::Element::Update ()`

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

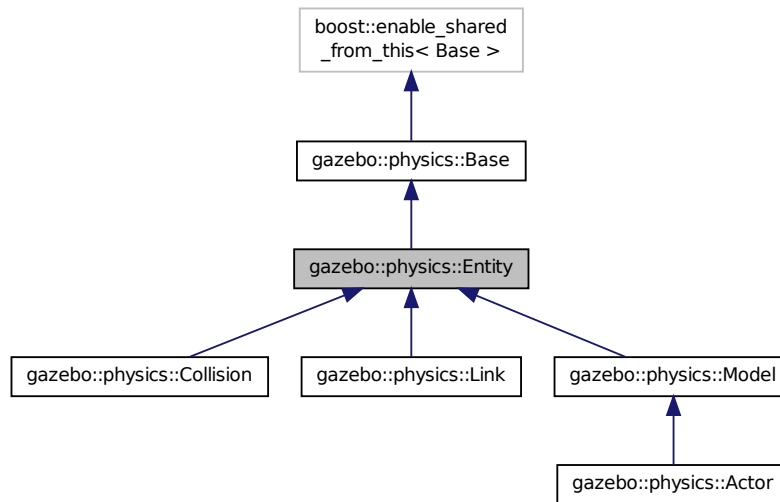
- **SDF.hh**

10.40 gazebo::physics::Entity Class Reference

Base (p. 125) class for all physics objects in Gazebo.

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

Inheritance diagram for gazebo::physics::Entity:



Public Member Functions

- **Entity** (**BasePtr** _parent)
Constructor.
- virtual \sim **Entity** ()
Destructor.
- virtual void **Fini** ()
Finalize the entity.
- virtual **math::Box** **GetBoundingBox** () const
Return the bounding box for the entity.
- **CollisionPtr** **GetChildCollision** (const std::string &_name)
Get a child collision entity, if one exists.
- **LinkPtr** **GetChildLink** (const std::string &_name)
Get a child linke entity, if one exists.
- **math::Box** **GetCollisionBoundingBox** () const
Returns collision bounding box.
- const **math::Pose** & **GetDirtyPose** () const
*Returns **Entity::dirtyPose** (p. 276).*
- void **GetNearestEntityBelow** (double &_distBelow, std::string &_entityName)
Get the distance to the nearest entity below (along the Z-axis) this entity.
- **ModelPtr** **GetParentModel** ()
Get the parent model, if one exists.
- virtual **math::Vector3** **GetRelativeAngularAccel** () const
Get the angular acceleration of the entity.
- virtual **math::Vector3** **GetRelativeAngularVel** () const

- Get the angular velocity of the entity.*

 - virtual **math::Vector3 GetRelativeLinearAccel** () const

Get the linear acceleration of the entity.

 - virtual **math::Vector3 GetRelativeLinearVel** () const

Get the linear velocity of the entity.

 - **math::Pose GetRelativePose** () const

Get the pose of the entity relative to its parent.

 - virtual **math::Vector3 GetWorldAngularAccel** () const

Get the angular acceleration of the entity in the world frame.

 - virtual **math::Vector3 GetWorldAngularVel** () const

Get the angular velocity of the entity in the world frame.

 - virtual **math::Vector3 GetWorldLinearAccel** () const

Get the linear acceleration of the entity in the world frame.

 - virtual **math::Vector3 GetWorldLinearVel** () const

Get the linear velocity of the entity in the world frame.

 - const **math::Pose & GetWorldPose** () const

Get the absolute pose of the entity.

 - bool **IsCanonicalLink** () const

A helper function that checks if this is a canonical body.

 - bool **IsStatic** () const

Return whether this entity is static.

 - virtual void **Load** (sdf::ElementPtr _sdf)

Load the entity.

 - void **PlaceOnEntity** (const std::string &_entityName)

Move this entity to be ontop of another entity by name.

 - void **PlaceOnNearestEntityBelow** ()

Move this entity to be ontop of the nearest entity below.

 - virtual void **Reset** ()

Reset the entity.

 - void **SetAnimation** (const common::PoseAnimationPtr &_anim, boost::function< void()> _onComplete)

Set an animation for this entity.

 - void **SetAnimation** (common::PoseAnimationPtr _anim)

Set an animation for this entity.

 - void **SetCanonicalLink** (bool _value)

Set to true if this entity is a canonical link for a model.

 - void **SetInitialRelativePose** (const math::Pose &_pose)

Set the initial pose.

 - virtual void **SetName** (const std::string &_name)

Set the name of the entity.

 - void **SetRelativePose** (const math::Pose &_pose, bool _notify=true, bool _publish=true)

Set the pose of the entity relative to its parent.

 - void **SetStatic** (const bool &_static)

Set whether this entity is static: immovable.

 - void **SetWorldPose** (const math::Pose &_pose, bool _notify=true, bool _publish=true)

Set the world pose of the entity.

 - void **SetWorldTwist** (const math::Vector3 &_linear, const math::Vector3 &_angular, bool _update-Children=true)

Set angular and linear rates of an **physics::Entity** (p. 265).

- virtual void **StopAnimation** ()
Stop the current animation, if any.
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)
Update the parameters using new sdf values.

Protected Member Functions

- virtual void **OnPoseChange** ()=0
This function is called when the entity's (or one of its parents) pose of the parent has changed.

Protected Attributes

- **common::PoseAnimationPtr** animation
Current pose animation.
- **event::ConnectionPtr** animationConnection
Connection used to update an animation.
- **math::Pose** animationStartPose
Start pose of an animation.
- std::vector< **event::ConnectionPtr** > connections
All our event connections.
- **math::Pose** dirtyPose
The pose set by a physics engine.
- **transport::NodePtr** node
Communication node.
- **EntityPtr** parentEntity
A helper that prevents numerous dynamic_casts.
- msgs::Pose * **poseMsg**
Pose message containr.
- **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

10.40.1 Detailed Description

Base (p. 125) class for all physics objects in Gazebo.

10.40.2 Constructor & Destructor Documentation

10.40.2.1 gazebo::physics::Entity::Entity (BasePtr *_parent*) [explicit]

Constructor.

Parameters

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

10.40.2.2 virtual gazebo::physics::Entity::~~Entity () [virtual]

Destructor.

10.40.3 Member Function Documentation

10.40.3.1 virtual void gazebo::physics::Entity::Fini () [virtual]

Finalize the entity.

Reimplemented from **gazebo::physics::Base** (p. 129).

Reimplemented in **gazebo::physics::Actor** (p. 104), **gazebo::physics::Model** (p. 465), **gazebo::physics::Link** (p. 406), and **gazebo::physics::Collision** (p. 184).

10.40.3.2 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. 406), **gazebo::physics::Model** (p. 465), and **gazebo::physics::Collision** (p. 184).

10.40.3.3 CollisionPtr gazebo::physics::Entity::GetChildCollision (const std::string & *_name*)

Get a child collision entity, if one exists.

Parameters

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

Returns

Pointer to the **Collision** (p. 180) object, or NULL if not found.

10.40.3.4 LinkPtr gazebo::physics::Entity::GetChildLink (const std::string & *_name*)

Get a child link entity, if one exists.

Parameters

in	<code>_name</code>	Name of the child Link (p. 398) object.
----	--------------------	--

Returns

Pointer to the **Link** (p. 398) object, or NULL if not found.

10.40.3.5 `math::Box gazebo::physics::Entity::GetCollisionBoundingBox () const`

Returns collision bounding box.

Returns

Collision bounding box.

10.40.3.6 `const math::Pose& gazebo::physics::Entity::GetDirtyPose () const`

Returns **Entity::dirtyPose** (p. 276).

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.

10.40.3.7 `void gazebo::physics::Entity::GetNearestEntityBelow (double & _distBelow, std::string & _entityName)`

Get the distance to the nearest entity below (along the Z-axis) this entity.

Parameters

out	<code>_distBelow</code>	The distance to the nearest entity below.
out	<code>_entityName</code>	The name of the nearest entity below.

10.40.3.8 `ModelPtr gazebo::physics::Entity::GetParentModel ()`

Get the parent model, if one exists.

Returns

Pointer to a model, or NULL if no parent model exists.

10.40.3.9 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularAccel () const` `[inline], [virtual]`

Get the angular acceleration of the entity.

Returns

A **math::Vector3** (p. 799) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 408), **gazebo::physics::Collision** (p. 185), and **gazebo::physics::Model** (p. 467).

10.40.3.10 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularVel () const [inline],[virtual]`

Get the angular velocity of the entity.

Returns

A **math::Vector3** (p. 799) for the velocity.

Reimplemented in **gazebo::physics::Link** (p. 409), **gazebo::physics::Collision** (p. 185), and **gazebo::physics::Model** (p. 468).

10.40.3.11 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearAccel () const [inline],[virtual]`

Get the linear acceleration of the entity.

Returns

A **math::Vector3** (p. 799) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 409), **gazebo::physics::Collision** (p. 185), and **gazebo::physics::Model** (p. 468).

10.40.3.12 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearVel () const [inline],[virtual]`

Get the linear velocity of the entity.

Returns

A **math::Vector3** (p. 799) for the linear velocity.

Reimplemented in **gazebo::physics::Link** (p. 409), **gazebo::physics::Collision** (p. 186), and **gazebo::physics::Model** (p. 468).

10.40.3.13 `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.

10.40.3.14 `virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularAccel () const [inline],[virtual]`

Get the angular acceleration of the entity in the world frame.

Returns

A `math::Vector3` (p. 799) for the acceleration.

Reimplemented in `gazebo::physics::Link` (p. 410), `gazebo::physics::Collision` (p. 187), and `gazebo::physics::Model` (p. 469).

10.40.3.15 `virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularVel () const [inline],[virtual]`

Get the angular velocity of the entity in the world frame.

Returns

A `math::Vector3` (p. 799) for the velocity.

Reimplemented in `gazebo::physics::Collision` (p. 187), and `gazebo::physics::Model` (p. 469).

10.40.3.16 `virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearAccel () const [inline],[virtual]`

Get the linear acceleration of the entity in the world frame.

Returns

A `math::Vector3` (p. 799) for the acceleration.

Reimplemented in `gazebo::physics::Link` (p. 411), `gazebo::physics::Collision` (p. 187), and `gazebo::physics::Model` (p. 469).

10.40.3.17 `virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearVel () const [inline],[virtual]`

Get the linear velocity of the entity in the world frame.

Returns

A `math::Vector3` (p. 799) for the linear velocity.

Reimplemented in `gazebo::physics::Collision` (p. 187), and `gazebo::physics::Model` (p. 469).

10.40.3.18 `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()`.

10.40.3.19 `bool gazebo::physics::Entity::IsCanonicalLink () const [inline]`

A helper function that checks if this is a canonical body.

Returns

True if the link is canonical.

10.40.3.20 `bool gazebo::physics::Entity::IsStatic () const`

Return whether this entity is static.

Returns

True if static.

10.40.3.21 `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. 132).

Reimplemented in `gazebo::physics::Actor` (p. 104), `gazebo::physics::Model` (p. 469), `gazebo::physics::Link` (p. 411), and `gazebo::physics::Collision` (p. 188).

10.40.3.22 `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. 411), and `gazebo::physics::Model` (p. 470).

10.40.3.23 `void gazebo::physics::Entity::PlaceOnEntity (const std::string & _entityName)`

Move this entity to be ontop of another entity by name.

Parameters

<code>in</code>	<code>_entityName</code>	Name of the Entity (p. 265) this Entity (p. 265) should be ontop of.
-----------------	--------------------------	--

10.40.3.24 `void gazebo::physics::Entity::PlaceOnNearestEntityBelow ()`

Move this entity to be ontop of the nearest entity below.

10.40.3.25 `virtual void gazebo::physics::Entity::Reset () [virtual]`

Reset the entity.

Reimplemented from **gazebo::physics::Base** (p. 134).

Reimplemented in **gazebo::physics::Model** (p. 470), and **gazebo::physics::Link** (p. 412).

10.40.3.26 `void gazebo::physics::Entity::SetAnimation (const common::PoseAnimationPtr & _anim, boost::function< void()> _onComplete)`

Set an animation for this entity.

Parameters

in	<code>_anim</code>	Pose animation.
in	<code>_onComplete</code>	Callback for when the animation completes.

10.40.3.27 `void gazebo::physics::Entity::SetAnimation (common::PoseAnimationPtr _anim)`

Set an animation for this entity.

Parameters

in	<code>_anim</code>	Pose animation.
----	--------------------	-----------------

10.40.3.28 `void gazebo::physics::Entity::SetCanonicalLink (bool _value)`

Set to true if this entity is a canonical link for a model.

Parameters

in	<code>_value</code>	True if the link is canonical.
----	---------------------	--------------------------------

10.40.3.29 `void gazebo::physics::Entity::SetInitialRelativePose (const math::Pose & _pose)`

Set the initial pose.

Parameters

in	<code>_pose</code>	The initial pose.
----	--------------------	-------------------

10.40.3.30 `virtual void gazebo::physics::Entity::SetName (const std::string & _name)` [virtual]

Set the name of the entity.

Parameters

in	<code>_name</code>	The new name.
----	--------------------	---------------

Reimplemented from **gazebo::physics::Base** (p. 134).

10.40.3.31 `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.

10.40.3.32 `void gazebo::physics::Entity::SetStatic (const bool & _static)`

Set whether this entity is static: immovable.

Parameters

in	<i>_static</i>	True = static.
----	----------------	----------------

10.40.3.33 `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	<i>_publish</i>	True to publish the pose.

10.40.3.34 `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. 265).

Parameters

in	<i>_linear</i>	Linear twist.
in	<i>_angular</i>	Angular twist.
in	<i>_updateChildren</i>	True to pass this update to child entities.

10.40.3.35 `virtual void gazebo::physics::Entity::StopAnimation () [virtual]`

Stop the current animation, if any.

Reimplemented in **gazebo::physics::Model** (p. 473).

10.40.3.36 `virtual void gazebo::physics::Entity::UpdateParameters (sdf::ElementPtr _sdf) [virtual]`

Update the parameters using new sdf values.

Parameters

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

Reimplemented from `gazebo::physics::Base` (p. 135).

Reimplemented in `gazebo::physics::Actor` (p. 105), `gazebo::physics::Model` (p. 474), `gazebo::physics::Link` (p. 415), and `gazebo::physics::Collision` (p. 189).

10.40.4 Member Data Documentation

10.40.4.1 `common::PoseAnimationPtr gazebo::physics::Entity::animation [protected]`

Current pose animation.

10.40.4.2 `event::ConnectionPtr gazebo::physics::Entity::animationConnection [protected]`

Connection used to update an animation.

10.40.4.3 `math::Pose gazebo::physics::Entity::animationStartPose [protected]`

Start pose of an animation.

10.40.4.4 `std::vector<event::ConnectionPtr> gazebo::physics::Entity::connections [protected]`

All our event connections.

10.40.4.5 `math::Pose gazebo::physics::Entity::dirtyPose [protected]`

The pose set by a physics engine.

10.40.4.6 `transport::NodePtr gazebo::physics::Entity::node [protected]`

Communication node.

10.40.4.7 `EntityPtr gazebo::physics::Entity::parentEntity [protected]`

A helper that prevents numerous `dynamic_casts`.

10.40.4.8 `msgs::Pose* gazebo::physics::Entity::poseMsg [protected]`

Pose message containr.

10.40.4.9 `common::Time gazebo::physics::Entity::prevAnimationTime` [protected]

Previous time an animation was updated.

10.40.4.10 `transport::PublisherPtr gazebo::physics::Entity::requestPub` [protected]

Request publisher.

10.40.4.11 `transport::PublisherPtr gazebo::physics::Entity::visPub` [protected]

Visual publisher.

10.40.4.12 `msgs::Visual* gazebo::physics::Entity::visualMsg` [protected]

Visual message container.

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

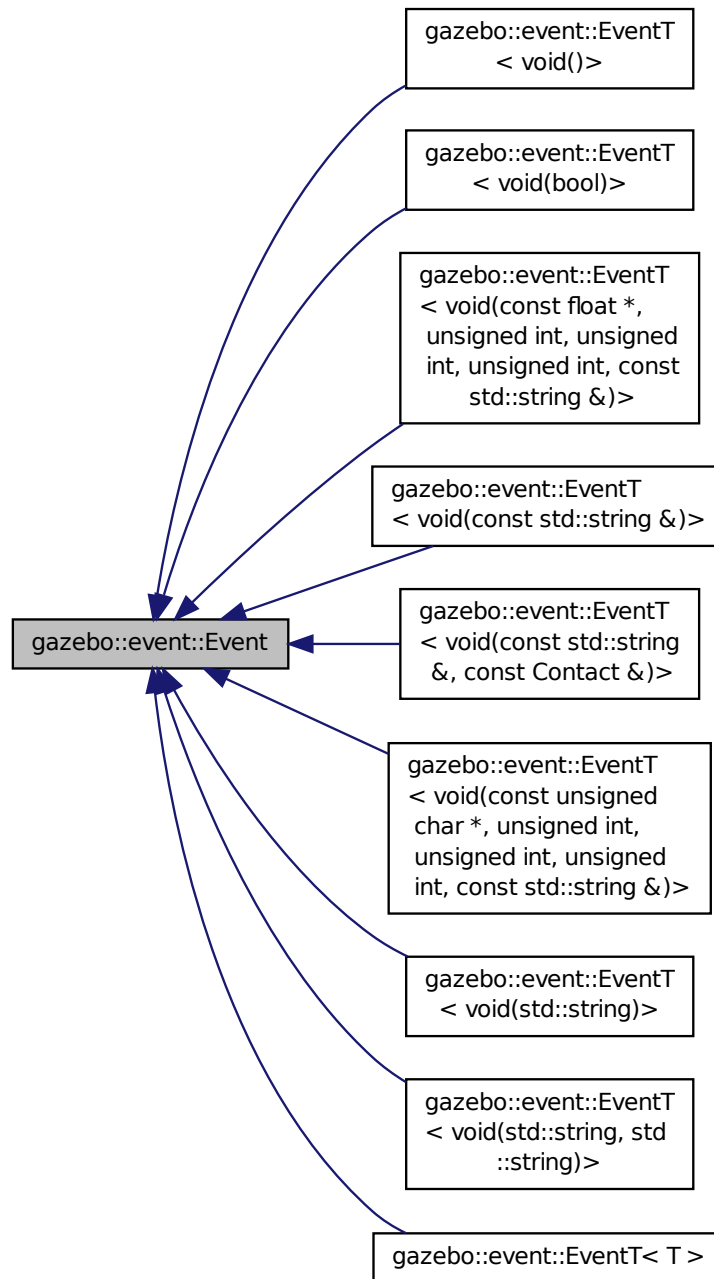
- `Entity.hh`

10.41 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.

10.41.1 Detailed Description

Base class for all events.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 virtual gazebo::event::Event::~Event () [inline],[virtual]

Constructor.

10.41.3 Member Function Documentation

10.41.3.1 virtual void gazebo::event::Event::Disconnect (ConnectionPtr _c) [pure virtual]

Disconnect.

Parameters

in	_c	A pointer to a connection
----	----	---------------------------

Implemented in `gazebo::event::EventT< T >` (p. 37), `gazebo::event::EventT< void(std::string)>` (p. 37), `gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>` (p. 37), `gazebo::event::EventT< void(const std::string &)>` (p. 37), `gazebo::event::EventT< void()>` (p. 37), `gazebo::event::EventT< void(const float *, unsigned int, unsigned int, const std::string &)>` (p. 37), `gazebo::event::EventT< void(const std::string &, const Contact &)>` (p. 37), `gazebo::event::EventT< void(std::string, std::string)>` (p. 37), and `gazebo::event::EventT< void(bool)>` (p. 37).

10.41.3.2 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. 38), `gazebo::event::EventT< void(std::string)>` (p. 38), `gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>` (p. 38), `gazebo::event::EventT< void(const std::string &)>` (p. 38), `gazebo::event::EventT< void()>` (p. 38), `gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>` (p. 38), `gazebo::event::EventT< void(const std::string &, const Contact &)>` (p. 38), `gazebo::event::EventT< void(std::string, std::string)>` (p. 38), and `gazebo::event::EventT< void(bool)>` (p. 38).

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

- **Event.hh**

10.42 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.
- `template<typename T >`
static event::ConnectionPtr ConnectViewContacts (T _subscriber)
Connect to a view contacts 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 void DisconnectViewContacts** (event::ConnectionPtr _connection)
Disconnect from a view contacts 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.
- **static event::EventT** < void(bool)> **viewContacts**
The event used to toggle contact visualization.

10.42.1 Detailed Description

Base class for rendering events.

10.42.2 Member Function Documentation

10.42.2.1 `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.

10.42.2.2 `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.

10.42.2.3 `template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectViewContacts (T
_subscriber) [inline],[static]`

Connect to a view contacts 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 viewContacts.

10.42.2.4 `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().

10.42.2.5 `static void gazebo::rendering::Events::DisconnectRemoveScene (event::ConnectionPtr _connection)
[inline],[static]`

Disconnect from a scene removed event.

Parameters

in	<code>_connection</code>	The connection to disconnect.
----	--------------------------	-------------------------------

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

10.42.2.6 `static void gazebo::rendering::Events::DisconnectViewContacts (event::ConnectionPtr _connection)`
`[inline],[static]`

Disconnect from a view contacts event.

Parameters

in	<code>_connection</code>	The connection to disconnect.
----	--------------------------	-------------------------------

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

10.42.3 Member Data Documentation

10.42.3.1 `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().

10.42.3.2 `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().

10.42.3.3 `event::EventT<void (bool)> gazebo::rendering::Events::viewContacts` `[static]`

The event used to toggle contact visualization.

Referenced by ConnectViewContacts(), and DisconnectViewContacts().

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

- **RenderEvents.hh**

10.43 gazebo::event::Events Class Reference

An **Event** (p. 277) 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 world has been created.

- static **EventT**< void()> **worldUpdateEnd**

World update has ended.

- static **EventT**< void()> **worldUpdateStart**

World update has started.

10.43.1 Detailed Description

An **Event** (p. 277) class to get notifications for simulator events.

10.43.2 Member Function Documentation

10.43.2.1 `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>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

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

10.43.2.2 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectCreateEntity (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the the add entity signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

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

10.43.2.3 `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.

10.43.2.4 `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.

10.43.2.5 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStop (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the diagnostic timer stop signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

Returns

a connection

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

10.43.2.6 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPause (T _subscriber)` `[inline],`
`[static]`

Connect a boost::slot the the pause signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

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

10.43.2.7 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPostRender (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the post render update signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

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

10.43.2.8 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPreRender (T _subscriber)`
`[inline],[static]`

Render start signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

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

10.43.2.9 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectRender (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the render update signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

Returns

a connection

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

10.43.2.10 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectSetSelectedEntity (T _subscriber) [inline], [static]`

Connect a boost::slot the set selected entity.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

10.43.2.11 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStep (T _subscriber) [inline], [static]`

Connect a boost::slot the the step signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

10.43.2.12 `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.

10.43.2.13 `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.

10.43.2.14 `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.

10.43.2.15 `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.

10.43.2.16 `static void gazebo::event::Events::DisconnectAddEntity (ConnectionPtr _subscriber)` `[inline],[static]`

Disconnect a boost::slot the the add entity signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

10.43.2.17 `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.

10.43.2.18 `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().

10.43.2.19 `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().

10.43.2.20 `static void gazebo::event::Events::DisconnectDiagTimerStop (ConnectionPtr _subscriber) [inline], [static]`

Disconnect a boost::slot the diagnostic timer stop signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

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

10.43.2.21 `static void gazebo::event::Events::DisconnectPause (ConnectionPtr _subscriber) [inline], [static]`

Disconnect a boost::slot the the pause signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

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

10.43.2.22 `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.

10.43.2.23 `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.

10.43.2.24 `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.

10.43.2.25 `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.

10.43.2.26 `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.

10.43.2.27 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.

10.43.2.28 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.

10.43.2.29 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.

10.43.2.30 static void gazebo::event::Events::DisconnectWorldUpdateStart (ConnectionPtr *_subscriber*) [inline],[static]

Disconnect a boost::slot the the world update start signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

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

10.43.3 Member Data Documentation

10.43.3.1 EventT<void (std::string)> gazebo::event::Events::addEntity [static]

An entity has been added.

Referenced by ConnectAddEntity(), and DisconnectAddEntity().

10.43.3.2 `EventT<void (std::string)> gazebo::event::Events::deleteEntity` [static]

An entity has been deleted.

Referenced by `ConnectDeleteEntity()`, and `DisconnectDeleteEntity()`.

10.43.3.3 `EventT<void (std::string)> gazebo::event::Events::diagTimerStart` [static]

Diagnostic timer start.

Referenced by `ConnectDiagTimerStart()`, and `DisconnectDiagTimerStart()`.

10.43.3.4 `EventT<void (std::string)> gazebo::event::Events::diagTimerStop` [static]

Diagnostic timer stop.

Referenced by `ConnectDiagTimerStop()`, and `DisconnectDiagTimerStop()`.

10.43.3.5 `EventT<void (std::string)> gazebo::event::Events::entityCreated` [static]

An entity has been created.

Referenced by `ConnectCreateEntity()`, and `DisconnectCreateEntity()`.

10.43.3.6 `EventT<void (bool)> gazebo::event::Events::pause` [static]

Pause signal.

Referenced by `ConnectPause()`, and `DisconnectPause()`.

10.43.3.7 `EventT<void ()> gazebo::event::Events::postRender` [static]

Post-Render.

Referenced by `ConnectPostRender()`, and `DisconnectPostRender()`.

10.43.3.8 `EventT<void ()> gazebo::event::Events::preRender` [static]

Pre-render.

Referenced by `ConnectPreRender()`, and `DisconnectPreRender()`.

10.43.3.9 `EventT<void ()> gazebo::event::Events::render` [static]

Render.

Referenced by `ConnectRender()`, and `DisconnectRender()`.

10.43.3.10 `EventT<void (std::string, std::string)> gazebo::event::Events::setSelectedEntity` [static]

An entity has been selected.

Referenced by `ConnectSetSelectedEntity()`, and `DisconnectSetSelectedEntity()`.

10.43.3.11 `EventT<void ()> gazebo::event::Events::step` [static]

Step the simulation once signal.

Referenced by `ConnectStep()`, and `DisconnectStep()`.

10.43.3.12 `EventT<void ()> gazebo::event::Events::stop` [static]

Simulation stop signal.

Referenced by `ConnectStop()`, and `DisconnectStop()`.

10.43.3.13 `EventT<void (std::string)> gazebo::event::Events::worldCreated` [static]

A world has been created.

Referenced by `ConnectWorldCreated()`, and `DisconnectWorldCreated()`.

10.43.3.14 `EventT<void ()> gazebo::event::Events::worldUpdateEnd` [static]

World update has ended.

Referenced by `ConnectWorldUpdateEnd()`, and `DisconnectWorldUpdateEnd()`.

10.43.3.15 `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:

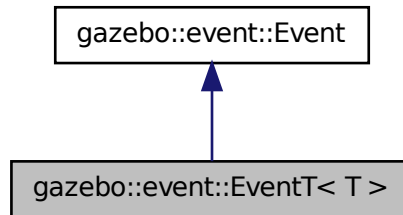
- **Events.hh**

10.44 `gazebo::event::EventT< T >` Class Template Reference

A class for event processing.

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

Inheritance diagram for gazebo::event::EventT< T >:



Public Member Functions

- virtual **~EventT** ()
Destructor.
- **ConnectionPtr Connect** (const boost::function< T > &_subscriber)
Connect a callback to this event.
- unsigned int **ConnectionCount** () const
Get the number of connections.
- virtual void **Disconnect** (**ConnectionPtr** _c)
Disconnect a callback to this event.
- virtual void **Disconnect** (int _id)
Disconnect a callback to this event.
- void **operator**() ()
Access the signal.
- template<typename P >
void **operator**() (const P &_p)
Signal the event with one parameter.
- template<typename P1 , typename P2 >
void **operator**() (const P1 &_p1, const P2 &_p2)
Signal the event with two parameters.
- template<typename P1 , typename P2 , typename P3 >
void **operator**() (const P1 &_p1, const P2 &_p2, const P3 &_p3)
Signal the event with three parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 >
void **operator**() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4)
Signal the event with four parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 >
void **operator**() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5)
Signal the event with five parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 >
void **operator**() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6)
Signal the event with six parameters.

- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 >`
`void operator() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7)`
Signal the event with seven parameters.
- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 >`
`void operator() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8)`
Signal the event with eight parameters.
- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 >`
`void operator() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9)`
Signal the event with nine parameters.
- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 >`
`void operator() (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9, const P10 &_p10)`
Signal the event with ten parameters.
- `void Signal ()`
Signal the event for all subscribers.
- `template<typename P >`
`void Signal (const P &_p)`
Signal the event with one parameter.
- `template<typename P1 , typename P2 >`
`void Signal (const P1 &_p1, const P2 &_p2)`
Signal the event with two parameter.
- `template<typename P1 , typename P2 , typename P3 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3)`
Signal the event with three parameter.
- `template<typename P1 , typename P2 , typename P3 , typename P4 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4)`
Signal the event with four parameter.
- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5)`
Signal the event with five parameter.
- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6)`
Signal the event with six parameter.
- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7)`
Signal the event with seven parameter.
- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8)`
Signal the event with eight parameter.
- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9)`

Signal the event with nine parameter.

- `template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 >`
`void Signal (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9, const P10 &_p10)`

Signal the event with ten parameter.

10.44.1 Detailed Description

```
template<typename T>class gazebo::event::EventT< T >
```

A class for event processing.

10.44.2 Member Function Documentation

10.44.2.1 `template<typename T> void gazebo::event::EventT< T >::operator() () [inline]`

Access the signal.

10.44.2.2 `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
----	----	---------------

10.44.2.3 `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

10.44.2.4 `template<typename T> template<typename P1 , typename P2 , typename P3 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3) [inline]`

Signal the event with three parameters.

Parameters

in	_p1	the first parameter
in	_p2	the second parameter
in	_p3	the second parameter

10.44.2.5 `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

10.44.2.6 `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

10.44.2.7 `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

10.44.2.8 `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

10.44.2.9 `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

10.44.2.10 `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

10.44.2.11 `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

10.44.2.12 `template<typename T> void gazebo::event::EventT< T >::Signal () [inline]`

Signal the event for all subscribers.

Referenced by `gazebo::event::EventT< void(bool)>::operator()()`.

10.44.2.13 `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
----	-----------------	-----------

10.44.2.14 `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

10.44.2.15 `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

10.44.2.16 `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

10.44.2.17 `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

10.44.2.18 `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

10.44.2.19 `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

10.44.2.20 `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

10.44.2.21 `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

```
10.44.2.22 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**

10.45 gazebo::common::Exception Class Reference

Class for generating exceptions.

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

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

10.45.1 Detailed Description

Class for generating exceptions.

10.45.2 Constructor & Destructor Documentation

10.45.2.1 gazebo::common::Exception::Exception ()

Constructor.

10.45.2.2 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

10.45.2.3 virtual gazebo::common::Exception::~~Exception () [virtual]

Destructor.

10.45.3 Member Function Documentation

10.45.3.1 std::string gazebo::common::Exception::GetErrorFile () const

Return the error function.

Returns

The error function name

10.45.3.2 std::string gazebo::common::Exception::GetErrorStr () const

Return the error string.

Returns

The error string

10.45.3.3 void gazebo::common::Exception::Print () const

Print the exception to std out.

10.45.4 Friends And Related Function Documentation

10.45.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::common::Exception & _err)` [friend]

stream insertion operator for Gazebo Error

Parameters

in	<code>_out</code>	the output stream
in	<code>_err</code>	the exception

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

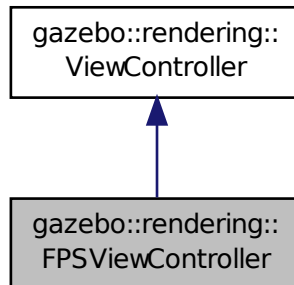
- **Exception.hh**

10.46 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

10.46.1 Detailed Description

First Person Shooter style view controller.

10.46.2 Constructor & Destructor Documentation

10.46.2.1 gazebo::rendering::FPSViewController::FPSViewController (UserCameraPtr *_camera*)

Constructor.

Parameters

in	Camera (p. 149)	to controll
----	------------------------	-------------

10.46.2.2 virtual gazebo::rendering::FPSViewController::~~FPSViewController () [virtual]

Destructor.

10.46.3 Member Function Documentation

10.46.3.1 static std::string gazebo::rendering::FPSViewController::GetTypeString () [static]

Get the type name of this view controller.

Returns

The name of the controller type: "fps"

10.46.3.2 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. 826).

10.46.3.3 `void gazebo::rendering::FPSViewController::HandleKeyReleaseEvent (const std::string & _key) [virtual]`

Handle a key release event.

Parameters

in	_key	The key that was released.
----	------	----------------------------

Implements **gazebo::rendering::ViewController** (p. 827).

10.46.3.4 `virtual void gazebo::rendering::FPSViewController::HandleMouseEvent (const common::MouseEvent & _event) [virtual]`

Handle a mouse event.

Parameters

in	_event	The mouse position.
----	--------	---------------------

Implements **gazebo::rendering::ViewController** (p. 827).

10.46.3.5 `virtual void gazebo::rendering::FPSViewController::Init () [virtual]`

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 827).

10.46.3.6 `virtual void gazebo::rendering::FPSViewController::Update () [virtual]`

Update the camera position.

Implements **gazebo::rendering::ViewController** (p. 828).

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

- **FPSViewController.hh**

10.47 urdf2gazebo::GazeboExtension Class Reference

```
#include <parser_urdf.hh>
```

Public Member Functions

- **GazeboExtension** ()
- **GazeboExtension** (const **GazeboExtension** &ge)

Public Attributes

- `std::vector< TiXmlElement * >` **blobs**
- double **damping_factor**
- `std::string` **fdir1**
- double **fudge_factor**
- bool **gravity**
- double **initial_joint_position**
- bool **is_damping_factor**
- bool **is_fudge_factor**
- bool **is_initial_joint_position**
- bool **is_kd**
- bool **is_kp**
- bool **is_laser_retro**
- bool **is_maxVel**
- bool **is_minDepth**
- bool **is_mu1**
- bool **is_mu2**
- bool **is_stop_cfm**
- bool **is_stop_erp**
- double **kd**
- double **kp**
- double **laser_retro**
- `std::string` **material**
- double **maxVel**
- double **minDepth**
- double **mu1**
- double **mu2**
- `std::string` **old_link_name**
- bool **provideFeedback**
- `gazebo::math::Pose` **reduction_transform**
- bool **self_collide**
- bool **setStaticFlag**
- double **stop_cfm**
- double **stop_erp**

10.47.1 Constructor & Destructor Documentation

10.47.1.1 `urdf2gazebo::GazeboExtension::GazeboExtension ()` `[inline]`

References `blobs`, `damping_factor`, `fdir1`, `fudge_factor`, `gravity`, `initial_joint_position`, `is_damping_factor`, `is_fudge_factor`, `is_initial_joint_position`, `is_kd`, `is_kp`, `is_laser_retro`, `is_maxVel`, `is_minDepth`, `is_mu1`, `is_mu2`, `is_stop_cfm`, `is_stop_erp`, `kd`, `kp`, `laser_retro`, `material`, `maxVel`, `minDepth`, `mu1`, `mu2`, `provideFeedback`, `self_collide`, `setStaticFlag`, `stop_cfm`, and `stop_erp`.

10.47.1.2 `urdf2gazebo::GazeboExtension::GazeboExtension (const GazeboExtension & ge)` `[inline]`

References `blobs`, `damping_factor`, `fdir1`, `fudge_factor`, `gravity`, `initial_joint_position`, `is_damping_factor`, `is_fudge_factor`, `is_initial_joint_position`, `is_kd`, `is_kp`, `is_laser_retro`, `is_maxVel`, `is_minDepth`, `is_mu1`, `is_mu2`, `is_stop_cfm`, `is_stop_erp`, `kd`, `kp`, `laser_retro`, `material`, `maxVel`, `minDepth`, `mu1`, `mu2`, `old_link_name`, `provideFeedback`, `reduction_transform`, `self_collide`, `setStaticFlag`, `stop_cfm`, and `stop_erp`.

10.47.2 Member Data Documentation

10.47.2.1 `std::vector<TiXmlElement*>` urdf2gazebo::GazeboExtension::blobs

Referenced by GazeboExtension().

10.47.2.2 `double` urdf2gazebo::GazeboExtension::damping_factor

Referenced by GazeboExtension().

10.47.2.3 `std::string` urdf2gazebo::GazeboExtension::fdir1

Referenced by GazeboExtension().

10.47.2.4 `double` urdf2gazebo::GazeboExtension::fudge_factor

Referenced by GazeboExtension().

10.47.2.5 `bool` urdf2gazebo::GazeboExtension::gravity

Referenced by GazeboExtension().

10.47.2.6 `double` urdf2gazebo::GazeboExtension::initial_joint_position

Referenced by GazeboExtension().

10.47.2.7 `bool` urdf2gazebo::GazeboExtension::is_damping_factor

Referenced by GazeboExtension().

10.47.2.8 `bool` urdf2gazebo::GazeboExtension::is_fudge_factor

Referenced by GazeboExtension().

10.47.2.9 `bool` urdf2gazebo::GazeboExtension::is_initial_joint_position

Referenced by GazeboExtension().

10.47.2.10 `bool` urdf2gazebo::GazeboExtension::is_kd

Referenced by GazeboExtension().

10.47.2.11 `bool` urdf2gazebo::GazeboExtension::is_kp

Referenced by GazeboExtension().

10.47.2.12 `bool urdf2gazebo::GazeboExtension::is_laser_retro`

Referenced by `GazeboExtension()`.

10.47.2.13 `bool urdf2gazebo::GazeboExtension::is_maxVel`

Referenced by `GazeboExtension()`.

10.47.2.14 `bool urdf2gazebo::GazeboExtension::is_minDepth`

Referenced by `GazeboExtension()`.

10.47.2.15 `bool urdf2gazebo::GazeboExtension::is_mu1`

Referenced by `GazeboExtension()`.

10.47.2.16 `bool urdf2gazebo::GazeboExtension::is_mu2`

Referenced by `GazeboExtension()`.

10.47.2.17 `bool urdf2gazebo::GazeboExtension::is_stop_cfm`

Referenced by `GazeboExtension()`.

10.47.2.18 `bool urdf2gazebo::GazeboExtension::is_stop_erp`

Referenced by `GazeboExtension()`.

10.47.2.19 `double urdf2gazebo::GazeboExtension::kd`

Referenced by `GazeboExtension()`.

10.47.2.20 `double urdf2gazebo::GazeboExtension::kp`

Referenced by `GazeboExtension()`.

10.47.2.21 `double urdf2gazebo::GazeboExtension::laser_retro`

Referenced by `GazeboExtension()`.

10.47.2.22 `std::string urdf2gazebo::GazeboExtension::material`

Referenced by `GazeboExtension()`.

10.47.2.23 double urdf2gazebo::GazeboExtension::maxVel

Referenced by GazeboExtension().

10.47.2.24 double urdf2gazebo::GazeboExtension::minDepth

Referenced by GazeboExtension().

10.47.2.25 double urdf2gazebo::GazeboExtension::mu1

Referenced by GazeboExtension().

10.47.2.26 double urdf2gazebo::GazeboExtension::mu2

Referenced by GazeboExtension().

10.47.2.27 std::string urdf2gazebo::GazeboExtension::old_link_name

Referenced by GazeboExtension().

10.47.2.28 bool urdf2gazebo::GazeboExtension::provideFeedback

Referenced by GazeboExtension().

10.47.2.29 gazebo::math::Pose urdf2gazebo::GazeboExtension::reduction_transform

Referenced by GazeboExtension().

10.47.2.30 bool urdf2gazebo::GazeboExtension::self_collide

Referenced by GazeboExtension().

10.47.2.31 bool urdf2gazebo::GazeboExtension::setStaticFlag

Referenced by GazeboExtension().

10.47.2.32 double urdf2gazebo::GazeboExtension::stop_cfm

Referenced by GazeboExtension().

10.47.2.33 double urdf2gazebo::GazeboExtension::stop_erp

Referenced by GazeboExtension().

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

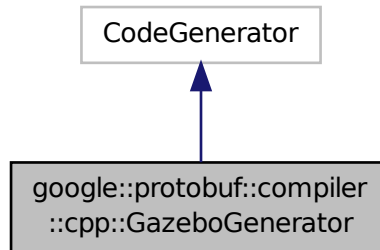
- **parser_urdf.hh**

10.48 google::protobuf::compiler::cpp::GazeboGenerator Class Reference

Google protobuf message generator for **gazebo::msgs** (p. 83).

```
#include <GazeboGenerator.hh>
```

Inheritance diagram for google::protobuf::compiler::cpp::GazeboGenerator:



Public Member Functions

- **GazeboGenerator** (const std::string &_name)
- virtual ~**GazeboGenerator** ()
- virtual bool **Generate** (const FileDescriptor *file, const string ¶meter, OutputDirectory *directory, string *error) const

10.48.1 Detailed Description

Google protobuf message generator for **gazebo::msgs** (p. 83).

10.48.2 Constructor & Destructor Documentation

10.48.2.1 google::protobuf::compiler::cpp::GazeboGenerator::GazeboGenerator (const std::string & *_name*)

10.48.2.2 virtual google::protobuf::compiler::cpp::GazeboGenerator::~~GazeboGenerator () [virtual]

10.48.3 Member Function Documentation

10.48.3.1 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:

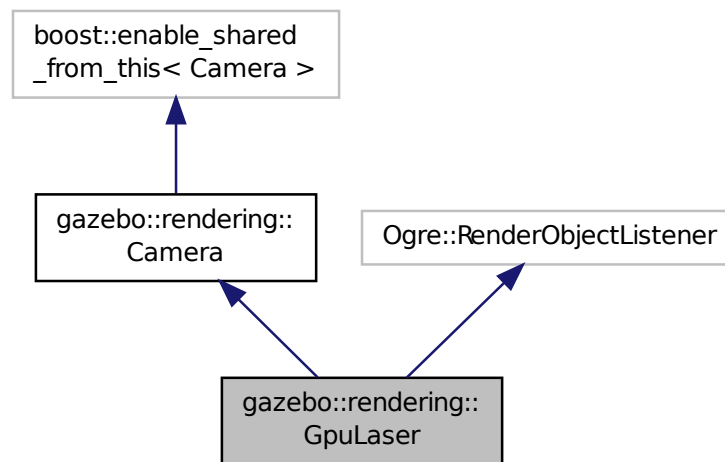
- **GazeboGenerator.hh**

10.49 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

10.49.1 Detailed Description

GPU based laser distance sensor.

10.49.2 Constructor & Destructor Documentation

10.49.2.1 gazebo::rendering::GpuLaser::GpuLaser (const std::string & *_namePrefix*, Scene * *_scene*, bool *_autoRender* = true)

Constructor.

Parameters

in	<i>_namePrefix</i>	Unique prefix name for the camera.
in	<i>_scene</i>	Scene (p. 632) that will contain the camera
in	<i>_autoRender</i>	Almost everyone should leave this as true.

10.49.2.2 virtual gazebo::rendering::GpuLaser::~GpuLaser () [virtual]

Destructor.

10.49.3 Member Function Documentation

10.49.3.1 template<typename T > event::ConnectionPtr gazebo::rendering::GpuLaser::ConnectNewLaserFrame (T *_subscriber*) [inline]

Connect to a laser frame signal.

Parameters

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

Returns

A pointer to the connection. This must be kept in scope.

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

10.49.3.2 void gazebo::rendering::GpuLaser::CreateLaserTexture (const std::string & *_textureName*)

Create the texture which is used to render laser data.

Parameters

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

10.49.3.3 void gazebo::rendering::GpuLaser::DisconnectNewLaserFrame (event::ConnectionPtr & *_c*) [inline]

Disconnect from a laser frame signal.

Parameters

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

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

10.49.3.4 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. 157).

10.49.3.5 const float* gazebo::rendering::GpuLaser::GetLaserData ()

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

Returns

Array of laser data.

10.49.3.6 virtual void gazebo::rendering::GpuLaser::Init () [virtual]

Initialize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 164).

10.49.3.7 virtual void gazebo::rendering::GpuLaser::Load (sdf::ElementPtr & *_sdf*) [virtual]

10.49.3.8 virtual void gazebo::rendering::GpuLaser::Load () [virtual]

Load the camera with default parameters.

Reimplemented from **gazebo::rendering::Camera** (p. 165).

10.49.3.9 virtual void gazebo::rendering::GpuLaser::notifyRenderSingleObject (Ogre::Renderable * *_rend*, const Ogre::Pass * *_p*, const Ogre::AutoParamDataSource * *_s*, const Ogre::LightList * *_ll*, bool *_supp*) [virtual]

10.49.3.10 `virtual void gazebo::rendering::GpuLaser::PostRender () [virtual]`

Post render.

Called after the render signal.

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

10.49.3.11 `void gazebo::rendering::GpuLaser::SetParentSensor (sensors::GpuRaySensor * _parent)`

Set the parent sensor.

Parameters

<code>in</code>	<code><i>_parent</i></code>	Pointer to a <code>sensors::GpuRaySensor</code> (p. 316)
-----------------	-----------------------------	--

10.49.3.12 `void gazebo::rendering::GpuLaser::SetRangeCount (unsigned int _w, unsigned int _h = 1)`

Set the number of laser samples in the width and height.

Parameters

<code>in</code>	<code><i>_w</i></code>	Number of samples in the horizontal sweep
<code>in</code>	<code><i>_h</i></code>	Number of samples in the vertical sweep

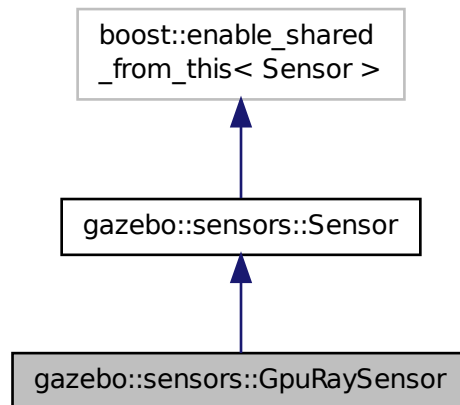
The documentation for this class was generated from the following file:

- `GpuLaser.hh`

10.50 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.
- double **Get1stRatio** () const **GAZEBO_DEPRECATED**
Deprecated.
- double **Get2ndRatio** () const **GAZEBO_DEPRECATED**
Deprecated.
- **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 **GetCHFOV** () const **GAZEBO_DEPRECATED**
Deprecated.
- double **GetCosHorzFOV** () const

- Get Cos Horz field-of-view.*

 - double **GetCosVertFOV** () const

Get Cos Vert field-of-view.
- double **GetCVFOV** () const **GAZEBO_DEPRECATED**

Deprecated.
- int **GetFiducial** (int _index) const

Get detected fiducial value for a ray.
- double **GetHAngle** () const **GAZEBO_DEPRECATED**

Deprecated.
- double **GetHFOV** () const **GAZEBO_DEPRECATED**

Deprecated.
- 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. 313).*
- 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 **GetVAngle** () const **GAZEBO_DEPRECATED**

Deprecated.
- 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.

 - double **GetVFOV** () const **GAZEBO_DEPRECATED**

Deprecated.

 - 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.

10.50.1 Constructor & Destructor Documentation

10.50.1.1 gazebo::sensors::GpuRaySensor::GpuRaySensor ()

Constructor.

10.50.1.2 virtual gazebo::sensors::GpuRaySensor::~~GpuRaySensor () [virtual]

Destructor.

10.50.2 Member Function Documentation

10.50.2.1 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.
----	--------------------------	-----------------

10.50.2.2 void gazebo::sensors::GpuRaySensor::DisconnectNewLaserFrame (event::ConnectionPtr & *_conn*)

Disconnect Laser Frame.

Parameters

in, out	<code>_conn</code>	Connection pointer to disconnect.
---------	--------------------	-----------------------------------

10.50.2.3 virtual void gazebo::sensors::GpuRaySensor::Fini () [protected],[virtual]

Finalize the ray.

Reimplemented from **gazebo::sensors::Sensor** (p. 655).

10.50.2.4 double gazebo::sensors::GpuRaySensor::Get1stRatio () const

Deprecated.

See Also

GetRayCountRatio (p. 325)

10.50.2.5 double gazebo::sensors::GpuRaySensor::Get2ndRatio () const

Deprecated.

See Also

GetRangeCountRatio (p. 324)

10.50.2.6 math::Angle gazebo::sensors::GpuRaySensor::GetAngleMax () const

Get the maximum angle.

Returns

the maximum angle

10.50.2.7 math::Angle gazebo::sensors::GpuRaySensor::GetAngleMin () const

Get the minimum angle.

Returns

The minimum angle

10.50.2.8 `double gazebo::sensors::GpuRaySensor::GetAngleResolution () const`

Get radians between each range.

10.50.2.9 `unsigned int gazebo::sensors::GpuRaySensor::GetCameraCount () const`

Gets the camera count.

Returns

Number of cameras

10.50.2.10 `double gazebo::sensors::GpuRaySensor::GetCHFOV () const`

Deprecated.

10.50.2.11 `double gazebo::sensors::GpuRaySensor::GetCosHorzFOV () const`

Get Cos Horz field-of-view.

Returns

$2 * \text{atan}(\tan(\text{this->hfov}/2) / \cos(\text{this->vfov}/2))$

10.50.2.12 `double gazebo::sensors::GpuRaySensor::GetCosVertFOV () const`

Get Cos Vert field-of-view.

Returns

$2 * \text{atan}(\tan(\text{this->vfov}/2) / \cos(\text{this->hfov}/2))$

10.50.2.13 `double gazebo::sensors::GpuRaySensor::GetCVFOV () const`

Deprecated.

See Also

GetCosVertFOV (p. 322)

10.50.2.14 `int gazebo::sensors::GpuRaySensor::GetFiducial (int _index) const`

Get detected fiducial value for a ray.

Warning: If you are accessing all the ray data in a loop it's possible that the Ray will update in the middle of your access loop. This means some data will come from one scan, and some from another scan. You can solve this problem by using `SetActive(false)` <your accessor loop> `SetActive(true)`.

Parameters

<code>in</code>	<code>_index</code>	Index of specific ray
-----------------	---------------------	-----------------------

Returns

Fiducial value of ray

10.50.2.15 `double gazebo::sensors::GpuRaySensor::GetHAngle () const`

Deprecated.

See Also

GetHorzHalfAngle (p. 323)

10.50.2.16 `double gazebo::sensors::GpuRaySensor::GetHFOV () const`

Deprecated.

See Also

GetHorzFOV (p. 323)

10.50.2.17 `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.

10.50.2.18 `double gazebo::sensors::GpuRaySensor::GetHorzHalfAngle () const`

Get $(\text{horizontal_max_angle} + \text{horizontal_min_angle}) * 0.5$.

Returns

$(\text{horizontal_max_angle} + \text{horizontal_min_angle}) * 0.5$

10.50.2.19 `rendering::GpuLaserPtr gazebo::sensors::GpuRaySensor::GetLaserCamera () const` `[inline]`

Returns a pointer to the internally kept **rendering::GpuLaser** (p. 313).

Returns

Pointer to GpuLaser

10.50.2.20 `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.

10.50.2.21 `int gazebo::sensors::GpuRaySensor::GetRangeCount () const`

Get the range count.

Returns

The number of ranges

10.50.2.22 `double gazebo::sensors::GpuRaySensor::GetRangeCountRatio () const`

Return the ratio of horizontal range count to vertical range count.

A ray count is the number of simulated rays. Whereas a range count is the total number of data points returned. When range count != ray count, then values are interpolated between rays.

10.50.2.23 `double gazebo::sensors::GpuRaySensor::GetRangeMax () const`

Get the maximum range.

Returns

The maximum range

10.50.2.24 `double gazebo::sensors::GpuRaySensor::GetRangeMin () const`

Get the minimum range.

Returns

The minimum range

10.50.2.25 `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

10.50.2.26 `void gazebo::sensors::GpuRaySensor::GetRanges (std::vector< double > & _ranges) const`

Get all the ranges.

Parameters

out	<i>_range</i>	A vector that will contain all the range data
-----	---------------	---

10.50.2.27 `int gazebo::sensors::GpuRaySensor::GetRayCount () const`

Get the ray count.

Returns

The number of rays

10.50.2.28 `double gazebo::sensors::GpuRaySensor::GetRayCountRatio () const`

Return the ratio of horizontal ray count to vertical ray count.

A ray count is the number of simulated rays. Whereas a range count is the total number of data points returned. When range count != ray count, then values are interpolated between rays.

10.50.2.29 `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

10.50.2.30 `double gazebo::sensors::GpuRaySensor::GetVAngle () const`

Deprecated.

See Also

GetVertHalfAngle (p. 326)

10.50.2.31 `double gazebo::sensors::GpuRaySensor::GetVertFOV () const`

Get the vertical field-of-view.

10.50.2.32 `double gazebo::sensors::GpuRaySensor::GetVertHalfAngle () const`

Get $(\text{vertical_max_angle} + \text{vertical_min_angle}) * 0.5$.

Returns

$(\text{vertical_max_angle} + \text{vertical_min_angle}) * 0.5$

10.50.2.33 `math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMax () const`

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.50.2.34 `math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMin () const`

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.50.2.35 `int gazebo::sensors::GpuRaySensor::GetVerticalRangeCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.50.2.36 `int gazebo::sensors::GpuRaySensor::GetVerticalRayCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.50.2.37 `double gazebo::sensors::GpuRaySensor::GetVFOV () const`

Deprecated.

See Also

GetVertFOV (p. 326)

10.50.2.38 `virtual void gazebo::sensors::GpuRaySensor::Init () [virtual]`

Initialize the ray.

Reimplemented from **gazebo::sensors::Sensor** (p. 657).

10.50.2.39 `bool gazebo::sensors::GpuRaySensor::IsHorizontal () const`

Gets if sensor is horizontal.

Returns

True if horizontal, false if not

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

Load the sensor with SDF parameters.

Parameters

<code>in</code>	<code>_sdf</code>	SDF Sensor (p. 652) parameters
<code>in</code>	<code>_worldName</code>	Name of world to load from

10.50.2.41 `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. 658).

10.50.2.42 `void gazebo::sensors::GpuRaySensor::SetAngleMax (double _angle)`

Set the scan maximum angle.

Parameters

<code>in</code>	<code><i>_angle</i></code>	The maximum angle
-----------------	----------------------------	-------------------

10.50.2.43 `void gazebo::sensors::GpuRaySensor::SetAngleMin (double _angle)`

Set the scan minimum angle.

Parameters

<code>in</code>	<code><i>_angle</i></code>	The minimum angle
-----------------	----------------------------	-------------------

10.50.2.44 `void gazebo::sensors::GpuRaySensor::SetVerticalAngleMax (double _angle)`

Set the vertical scan line top angle.

Parameters

<code>in</code>	<code><i>_angle</i></code>	The Maximum angle of the scan block
-----------------	----------------------------	-------------------------------------

10.50.2.45 `void gazebo::sensors::GpuRaySensor::SetVerticalAngleMin (double _angle)`

Set the vertical scan bottom angle.

Parameters

<code>in</code>	<code><i>_angle</i></code>	The minimum angle of the scan block
-----------------	----------------------------	-------------------------------------

10.50.2.46 `virtual void gazebo::sensors::GpuRaySensor::UpdateImpl (bool _force)` `[protected],[virtual]`

Update the sensor information.

Parameters

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

Reimplemented from `gazebo::sensors::Sensor` (p. 659).

10.50.3 Member Data Documentation

10.50.3.1 `unsigned int gazebo::sensors::GpuRaySensor::cameraCount` `[protected]`

Number of cameras.

10.50.3.2 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::cameraElem` [protected]

Camera SDF element.

10.50.3.3 `double gazebo::sensors::GpuRaySensor::chfov` [protected]

Cos horizontal field-of-view.

10.50.3.4 `double gazebo::sensors::GpuRaySensor::cvfov` [protected]

Cos vertical field-of-view.

10.50.3.5 `double gazebo::sensors::GpuRaySensor::far` [protected]

Far clip plane.

10.50.3.6 `double gazebo::sensors::GpuRaySensor::hfov` [protected]

Horizontal field-of-view.

10.50.3.7 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::horzElem` [protected]

Horizontal SDF element.

10.50.3.8 `double gazebo::sensors::GpuRaySensor::horzHalfAngle` [protected]

Horizontal half angle.

10.50.3.9 `unsigned int gazebo::sensors::GpuRaySensor::horzRangeCount` [protected]

Horizontal range count.

10.50.3.10 `unsigned int gazebo::sensors::GpuRaySensor::horzRayCount` [protected]

Horizontal ray count.

10.50.3.11 `bool gazebo::sensors::GpuRaySensor::isHorizontal` [protected]

True if the sensor is horizontal only.

10.50.3.12 `double gazebo::sensors::GpuRaySensor::near` [protected]

Near clip plane.

10.50.3.13 `double gazebo::sensors::GpuRaySensor::rangeCountRatio` [protected]

Range count ratio.

10.50.3.14 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::rangeElem` [protected]

Range SDF element.

10.50.3.15 `double gazebo::sensors::GpuRaySensor::rayCountRatio` [protected]

Ray count ratio.

10.50.3.16 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::scanElem` [protected]

Scan SDF element.

10.50.3.17 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::vertElem` [protected]

Vertical SDF element.

10.50.3.18 `double gazebo::sensors::GpuRaySensor::vertHalfAngle` [protected]

Vertical half angle.

10.50.3.19 `unsigned int gazebo::sensors::GpuRaySensor::vertRangeCount` [protected]

Vertical range count.

10.50.3.20 `unsigned int gazebo::sensors::GpuRaySensor::vertRayCount` [protected]

Vertical ray count.

10.50.3.21 `double gazebo::sensors::GpuRaySensor::vfov` [protected]

Vertical field-of-view.

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

- **GpuRaySensor.hh**

10.51 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. 98) 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.

10.51.1 Detailed Description

Displays a grid of cells, drawn with lines.

Displays a grid of cells, drawn with lines. A grid with an identity orientation is drawn along the XY plane.

10.51.2 Constructor & Destructor Documentation

10.51.2.1 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

10.51.2.2 gazebo::rendering::Grid::~~Grid ()

Destructor.

10.51.3 Member Function Documentation

10.51.3.1 void gazebo::rendering::Grid::Enable (bool *enable*)

Enable or disable the grid.

Parameters

in	<code>_enable</code>	Set to true to view the grid, false to make invisible.
----	----------------------	--

10.51.3.2 uint32_t gazebo::rendering::Grid::GetCellCount () const [inline]

Get the number of cells.

10.51.3.3 float gazebo::rendering::Grid::GetCellLength () const [inline]

Get the cell length.

Returns

The cell length

10.51.3.4 common::Color gazebo::rendering::Grid::GetColor () const [inline]

Return the grid color.

Returns

The grid color

10.51.3.5 uint32_t gazebo::rendering::Grid::GetHeight () const [inline]

Get the height of the grid.

Returns

The height

10.51.3.6 `float gazebo::rendering::Grid::GetLineWidth () const [inline]`

Get the width of the grid line.

Returns

The line width

10.51.3.7 `Ogre::SceneNode* gazebo::rendering::Grid::GetSceneNode () [inline]`

Get the **Ogre** (p. 98) scene node associated with this grid.

Returns

The **Ogre** (p. 98) scene node associated with this grid

10.51.3.8 `void gazebo::rendering::Grid::Init ()`

Initialize the grid.

10.51.3.9 `void gazebo::rendering::Grid::SetCellCount (uint32_t _count)`

Set the number of cells.

Parameters

<code>in</code>	<code>_count</code>	The number of cells
-----------------	---------------------	---------------------

10.51.3.10 `void gazebo::rendering::Grid::SetCellLength (float _len)`

Set the cell length.

Parameters

<code>in</code>	<code>_len</code>	The cell length
-----------------	-------------------	-----------------

10.51.3.11 `void gazebo::rendering::Grid::SetColor (const common::Color & _color)`

Sets the color of the grid.

Parameters

<code>in</code>	<code>_color</code>	The grid color
-----------------	---------------------	----------------

10.51.3.12 `void gazebo::rendering::Grid::SetHeight (uint32_t _count)`

Set the height of the grid.

Parameters

in	<code>_count</code>	Grid (p. 330) height
----	---------------------	-----------------------------

10.51.3.13 void gazebo::rendering::Grid::SetLineWidth (float *_width*)

Set the line width.

Parameters

in	<code>_width</code>	The width of the grid
----	---------------------	-----------------------

10.51.3.14 void gazebo::rendering::Grid::SetUserData (const Ogre::Any & *_data*)

Sets user data on all ogre objects we own.

Parameters

in	<code>_data</code>	The user data
----	--------------------	---------------

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

- **Grid.hh**

10.52 gazebo::physics::Gripper Class Reference

A 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.

10.52.1 Detailed Description

A gripper abstraction.

A gripper is a collection of links that act as a gripper. This class will intelligently generate fixed joints between the gripper and an object within the gripper. This allows the object to be manipulated without falling or behaving poorly.

10.52.2 Constructor & Destructor Documentation

10.52.2.1 gazebo::physics::Gripper::Gripper (ModelPtr _model) [explicit]

Constructor.

Parameters

in	_model	The model which contains the Gripper (p. 334).
----	--------	---

10.52.2.2 virtual gazebo::physics::Gripper::~~Gripper () [virtual]

Destructor.

10.52.3 Member Function Documentation

10.52.3.1 virtual void gazebo::physics::Gripper::Init () [virtual]

Initialize.

10.52.3.2 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**

10.53 gazebo::rendering::GUIOverlay Class Reference

A class that creates a CEGUI overlay on a render window.

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

Public Member Functions

- **GUIOverlay ()**
Constructor.
- virtual **~GUIOverlay ()**
Destructor.
- bool **AttachCameraToImage (CameraPtr &_camera, const std::string &_windowName)**
*Use this function to draw the output from a **rendering::Camera** (p. 149) to and overlay window.*
- bool **AttachCameraToImage (DepthCameraPtr &_camera, const std::string &_windowName)**
*Use this function to draw the output from a **rendering::DepthCamera** (p. 238) 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.

10.53.1 Detailed Description

A class that creates a CEGUI overlay on a render window.

10.53.2 Constructor & Destructor Documentation

10.53.2.1 gazebo::rendering::GUIOverlay::GUIOverlay ()

Constructor.

10.53.2.2 virtual gazebo::rendering::GUIOverlay::~GUIOverlay () [virtual]

Destructor.

10.53.3 Member Function Documentation

10.53.3.1 bool gazebo::rendering::GUIOverlay::AttachCameraToImage (CameraPtr & _camera, const std::string & _windowName)

Use this function to draw the output from a **rendering::Camera** (p. 149) 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

10.53.3.2 `bool gazebo::rendering::GUIOverlay::AttachCameraToImage (DepthCameraPtr & _camera, const std::string & _windowName)`

Use this function to draw the output from a **rendering::DepthCamera** (p. 238) 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

10.53.3.3 `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> .

10.53.3.4 `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.

10.53.3.5 `bool gazebo::rendering::GUIOverlay::HandleKeyPressEvent (const std::string & _key)`

Handle a key press event.

Parameters

<code>in</code>	<code><i>_key</i></code>	The key pressed.
-----------------	--------------------------	------------------

Returns

True if the key press event was handled.

10.53.3.6 `bool gazebo::rendering::GUIOverlay::HandleKeyReleaseEvent (const std::string & _key)`

Handle a key release event.

Parameters

<code>in</code>	<code><i>_key</i></code>	The key released.
-----------------	--------------------------	-------------------

Returns

True if the key release event was handled.

10.53.3.7 `bool gazebo::rendering::GUIOverlay::HandleMouseEvent (const common::MouseEvent & _evt)`

Handle a mouse event.

Parameters

<code>in</code>	<code><i>_evt</i></code>	The mouse event.
-----------------	--------------------------	------------------

Returns

True if the mouse event was handled.

10.53.3.8 `void gazebo::rendering::GUIOverlay::Hide ()`

Make the overlay invisible.

10.53.3.9 `void gazebo::rendering::GUIOverlay::Init (Ogre::RenderTarget * _renderTarget)`

Initialize the overlay.

Parameters

<code>in</code>	<code><i>_renderTarget</i></code>	The render target which will have the overlay.
-----------------	-----------------------------------	--

10.53.3.10 `bool gazebo::rendering::GUIOverlay::IsInitialized ()`

Return true if the overlay has been initialized.

Returns

True if initialized

10.53.3.11 `void gazebo::rendering::GUIOverlay::LoadLayout (const std::string & _filename)`

Load a CEGUI layout file.

Parameters

<code>in</code>	<code>_filename</code>	Name of the layout file.
-----------------	------------------------	--------------------------

10.53.3.12 `void gazebo::rendering::GUIOverlay::Resize (unsigned int _width, unsigned int _height)`

Resize the window.

10.53.3.13 `void gazebo::rendering::GUIOverlay::Show ()`

Make the overlay visible.

10.53.3.14 `void gazebo::rendering::GUIOverlay::Update ()`

Update the overlay's objects.

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

- **GUIOverlay.hh**

10.54 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.
- void **Load** ()

Load the heightmap.

- void **LoadFromMsg** (ConstVisualPtr &_msg)

Load the heightmap from a visual message.

10.54.1 Detailed Description

Rendering a terrain using heightmap information.

10.54.2 Constructor & Destructor Documentation

10.54.2.1 gazebo::rendering::Heightmap::Heightmap (ScenePtr _scene)

Constructor.

Parameters

in	_scene	Pointer to the scene that will contain the heightmap
----	--------	--

10.54.2.2 virtual gazebo::rendering::Heightmap::~~Heightmap () [virtual]

Destructor.

10.54.3 Member Function Documentation

10.54.3.1 double gazebo::rendering::Heightmap::GetHeight (double _x, double _y, double _z = 1000)

Get the height at a location.

Parameters

in	_x	X location
in	_y	Y location
in	_z	Z location

Returns

The height at the specified location

10.54.3.2 void gazebo::rendering::Heightmap::Load ()

Load the heightmap.

10.54.3.3 void gazebo::rendering::Heightmap::LoadFromMsg (ConstVisualPtr & _msg)

Load the heightmap from a visual message.

Parameters

in	<code>_msg</code>	The visual message containing heightmap info
----	-------------------	--

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

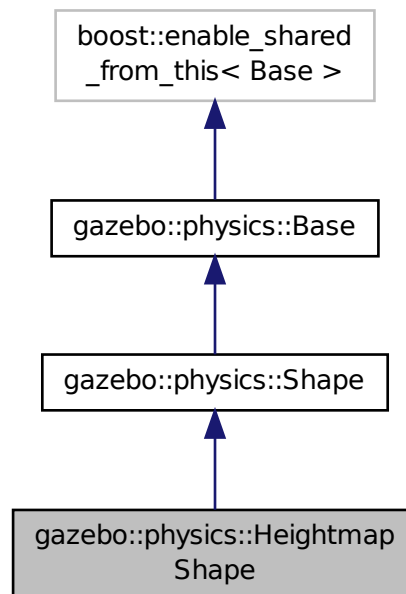
- **Heightmap.hh**

10.55 gazebo::physics::HeightmapShape Class Reference

HeightmapShape (p. 341) 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

10.55.1 Detailed Description

HeightmapShape (p. 341) 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.

10.55.2 Constructor & Destructor Documentation

10.55.2.1 gazebo::physics::HeightmapShape::HeightmapShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	<i>_parent</i>	Parent Collision (p. 180) object.
----	----------------	--

10.55.2.2 virtual gazebo::physics::HeightmapShape::~~HeightmapShape () [virtual]

Destructor.

10.55.3 Member Function Documentation

10.55.3.1 void gazebo::physics::HeightmapShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Fill a geometry message with this shape's data.

Parameters

in	<i>_msg</i>	Message to fill.
----	-------------	------------------

Implements **gazebo::physics::Shape** (p. 669).

10.55.3.2 float gazebo::physics::HeightmapShape::GetHeight (int *_x*, int *_y*)

Get a height at a position.

Parameters

in	<i>_x</i>	X position.
in	<i>_y</i>	Y position.

Returns

The height at a the specified location.

10.55.3.3 float gazebo::physics::HeightmapShape::GetMaxHeight () const

Get the maximum height.

Returns

The maximum height.

10.55.3.4 float gazebo::physics::HeightmapShape::GetMinHeight () const

Get the minimum height.

Returns

The minimum height.

10.55.3.5 `math::Vector3 gazebo::physics::HeightmapShape::GetPos () const`

Get the origin in world coordinate frame.

Returns

The origin in world coordinate frame.

10.55.3.6 `math::Vector3 gazebo::physics::HeightmapShape::GetSize () const`

Get the size in meters.

Returns

The size in meters.

10.55.3.7 `int gazebo::physics::HeightmapShape::GetSubSampling () const`

Get the amount of subsampling.

Returns

Amount of subsampling.

10.55.3.8 `std::string gazebo::physics::HeightmapShape::GetURI () const`

Get the URI of the heightmap image.

Returns

The heightmap image URI.

10.55.3.9 `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. 790), result.x = width, result.y = length/height.

10.55.3.10 `virtual void gazebo::physics::HeightmapShape::Init () [virtual]`

Initialize the heightmap.

Implements `gazebo::physics::Shape` (p. 670).

10.55.3.11 virtual void gazebo::physics::HeightmapShape::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the heightmap.

Parameters

in	<i>_sdf</i>	SDF value to load from.
----	-------------	-------------------------

Reimplemented from **gazebo::physics::Base** (p. 132).

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

Update the heightmap from a message.

Parameters

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

Implements **gazebo::physics::Shape** (p. 670).

10.55.4 Member Data Documentation

10.55.4.1 std::vector<float> gazebo::physics::HeightmapShape::heights [protected]

Lookup table of heights.

10.55.4.2 common::Image gazebo::physics::HeightmapShape::img [protected]

Image used to generate the heights.

10.55.4.3 math::Vector3 gazebo::physics::HeightmapShape::scale [protected]

Scaling factor.

10.55.4.4 int gazebo::physics::HeightmapShape::subSampling [protected]

Level of subsampling.

10.55.4.5 unsigned int gazebo::physics::HeightmapShape::vertSize [protected]

Size of the height lookup table.

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

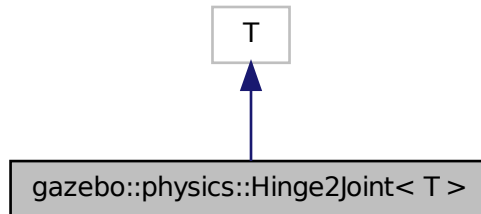
- **HeightmapShape.hh**

10.56 gazebo::physics::Hinge2Joint< T > Class Template Reference

A two axis hinge joint.

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

Inheritance diagram for gazebo::physics::Hinge2Joint< T >:



Public Member Functions

- **Hinge2Joint** (**BasePtr** _parent)
Constructor.
- virtual **~Hinge2Joint** ()
Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (**sdf::ElementPtr** _sdf)
Load the joint.

10.56.1 Detailed Description

```
template<class T>class gazebo::physics::Hinge2Joint< T >
```

A two axis hinge joint.

10.56.2 Constructor & Destructor Documentation

10.56.2.1 `template<class T > gazebo::physics::Hinge2Joint< T >::Hinge2Joint (BasePtr _parent) [inline], [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Parent link.
----	----------------------	--------------

References gazebo::physics::Base::HINGE2_JOINT.

10.56.2.2 `template<class T> virtual gazebo::physics::Hinge2Joint< T >::~~Hinge2Joint () [inline], [virtual]`

Destructor.

10.56.3 Member Function Documentation

10.56.3.1 `template<class T> virtual unsigned int gazebo::physics::Hinge2Joint< T >::GetAngleCount () const [inline], [virtual]`

10.56.3.2 `template<class T> virtual void gazebo::physics::Hinge2Joint< T >::Load (sdf::ElementPtr _sdf) [inline], [virtual]`

Load the joint.

Parameters

in	_sdf	SDF values to load from.
----	------	--------------------------

References `sdf::Element::GetElement()`, and `sdf::Element::GetValueVector3()`.

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

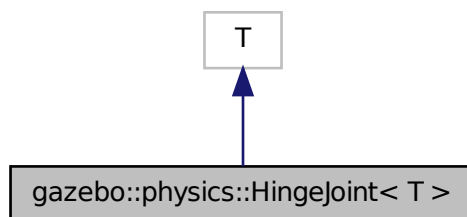
- [Hinge2Joint.hh](#)

10.57 gazebo::physics::HingeJoint< T > Class Template Reference

A single axis hinge joint.

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

Inheritance diagram for `gazebo::physics::HingeJoint< T >`:



Public Member Functions

- **HingeJoint (BasePtr _parent)**
Constructor.

- virtual `~HingeJoint ()`
Destructor.
- virtual unsigned int `GetAngleCount () const`
- virtual void `Load (sdf::ElementPtr _sdf)`
Load joint.

Protected Member Functions

- virtual void `Init ()`
Initialize joint.

10.57.1 Detailed Description

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

A single axis hinge joint.

10.57.2 Constructor & Destructor Documentation

10.57.2.1 `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`.

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

Destructor.

10.57.3 Member Function Documentation

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

10.57.3.2 `template<class T > virtual void gazebo::physics::HingeJoint< T >::Init () [inline], [protected], [virtual]`

Initialize joint.

References `gazebo::msgs::Init()`.

10.57.3.3 `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**

10.58 gazebo::common::Image Class Reference

Encapsulates an image.

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

Public Types

- enum **PixelFormat** {
UNKNOWN, 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 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.

10.58.1 Detailed Description

Encapsulates an image.

10.58.2 Member Enumeration Documentation

10.58.2.1 enum gazebo::common::Image::PixelFormat

Pixel formats enumeration.

Enumerator:

UNKNOWN
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_GRGB8***

10.58.3 Constructor & Destructor Documentation

10.58.3.1 `gazebo::common::Image::Image (const std::string & _filename = " ") [explicit]`

Constructor.

Parameters

<i>in</i>	<i>_filename</i>	the path to the image
-----------	------------------	-----------------------

10.58.3.2 `virtual gazebo::common::Image::~~Image () [virtual]`

Destructor.

10.58.4 Member Function Documentation

10.58.4.1 `Color gazebo::common::Image::GetAvgColor ()`

Get the average color.

Returns

The average color

10.58.4.2 `unsigned int gazebo::common::Image::GetBPP () const`

Get the size of one pixel in bits.

Returns

The BPP of the image

10.58.4.3 `void gazebo::common::Image::GetData (unsigned char ** _data, unsigned int & _count) const`

Get the image as a data array.

Parameters

<i>out</i>	<i>_data</i>	Pointer to a NULL array of char.
<i>out</i>	<i>_count</i>	The resulting data array size

10.58.4.4 `std::string gazebo::common::Image::GetFilename () const`

Get the full filename of the image.

Returns

The filename used to load the image

10.58.4.5 `unsigned int gazebo::common::Image::GetHeight () const`

Get the height.

Returns

The image height

10.58.4.6 `Color gazebo::common::Image::GetMaxColor ()`

Get the max color.

Returns

The max color

10.58.4.7 `int gazebo::common::Image::GetPitch () const`

Returns

The pitch of the image

10.58.4.8 `Color gazebo::common::Image::GetPixel (unsigned int _x, unsigned int _y)`

Get a pixel color value.

Parameters

<code>in</code>	<code>_x</code>	Column location in the image
<code>in</code>	<code>_y</code>	Row location in the image

10.58.4.9 `PixelFormat gazebo::common::Image::GetPixelFormat () const`

Get the pixel format.

Returns

PixelFormat

10.58.4.10 void gazebo::common::Image::GetRGBData (unsigned char ** *_data*, unsigned int & *_count*) const

Get only the RGB data from the image.

This will drop the alpha channel if one is present.

Parameters

out	<i>_data</i>	Pointer to a NULL array of char.
out	<i>_count</i>	The resulting data array size

10.58.4.11 unsigned int gazebo::common::Image::GetWidth () const

Get the width.

Returns

The image width

10.58.4.12 int gazebo::common::Image::Load (const std::string & *_filename*)

Load an image.

Return 0 on success

Parameters

in	<i>_filename</i>	the path to the image file
----	------------------	----------------------------

10.58.4.13 void gazebo::common::Image::Rescale (int *_width*, int *_height*)

Rescale the image.

Parameters

in	<i>_width</i>	New image width
in	<i>_height</i>	New image height

10.58.4.14 void gazebo::common::Image::SavePNG (const std::string & *_filename*)

Save the image in PNG format.

Parameters

in	<i>_filename</i>	The name of the saved image
----	------------------	-----------------------------

10.58.4.15 void gazebo::common::Image::SetFromData (const unsigned char * *_data*, unsigned int *_width*, unsigned int *_height*, Image::PixelFormat *_format*)

Set the image from raw data.

Parameters

in	<code>_data</code>	Pointer to the raw image data
in	<code>_width</code>	Width in pixels
in	<code>_height</code>	Height in pixels
in	<code>_format</code>	Pixel format of the provided data

10.58.4.16 `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:

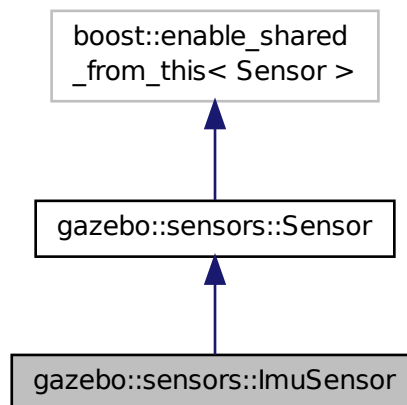
- **Image.hh**

10.59 gazebo::sensors::ImuSensor Class Reference

An IMU sensor.

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

Inheritance diagram for gazebo::sensors::ImuSensor:



Public Member Functions

- **ImuSensor ()**

Constructor.

- virtual `~ImuSensor ()`

Destructor.

- `math::Vector3 GetAngularVelocity ()` const

Returns the angular velocity.

- `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

10.59.1 Detailed Description

An IMU sensor.

10.59.2 Constructor & Destructor Documentation

10.59.2.1 gazebo::sensors::ImuSensor::ImuSensor ()

Constructor.

10.59.2.2 virtual gazebo::sensors::ImuSensor::~~ImuSensor () [virtual]

Destructor.

10.59.3 Member Function Documentation

10.59.3.1 virtual void gazebo::sensors::ImuSensor::Fini () [protected],[virtual]

Finalize the sensor.

Reimplemented from `gazebo::sensors::Sensor` (p. 655).

10.59.3.2 `math::Vector3 gazebo::sensors::ImuSensor::GetAngularVelocity () const`

Returns the angular velocity.

Returns

Angular velocity.

10.59.3.3 `math::Vector3 gazebo::sensors::ImuSensor::GetLinearAcceleration () const`

Returns the linear acceleration.

Returns

Linear acceleration.

10.59.3.4 `virtual void gazebo::sensors::ImuSensor::Init () [protected],[virtual]`

Initialize the IMU.

Reimplemented from `gazebo::sensors::Sensor` (p. 657).

10.59.3.5 `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 Sensor (p. 652) parameters.
in	<code>_worldName</code>	Name of world to load from.

Reimplemented from `gazebo::sensors::Sensor` (p. 657).

10.59.3.6 `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. 658).

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

Reimplemented from **gazebo::sensors::Sensor** (p. 659).

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

- **ImuSensor.hh**

10.60 gazebo::physics::Inertial Class Reference

A class for inertial information about a link.

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

Public Member Functions

- **Inertial** ()
Default Constructor.
- **Inertial** (double *_mass*)
Constructor.
- **Inertial** (const **Inertial** &*_inertial*)
Copy constructor.
- virtual **~Inertial** ()
Destructor.
- const **math::Vector3** & **GetCoG** () const
Get the center of gravity.
- 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. 398) 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.

10.60.1 Detailed Description

A class for inertial information about a link.

10.60.2 Constructor & Destructor Documentation

10.60.2.1 gazebo::physics::Inertial::Inertial ()

Default Constructor.

10.60.2.2 gazebo::physics::Inertial::Inertial (double *_mass*) [explicit]

Constructor.

Parameters

in	<i>_mass</i>	Mass value in kg if using metric.
----	--------------	-----------------------------------

10.60.2.3 gazebo::physics::Inertial::Inertial (const Inertial & *_inertial*)

Copy constructor.

Parameters

in	<i>_inertial</i>	Inertial (p. 357) element to copy
----	------------------	--

10.60.2.4 virtual gazebo::physics::Inertial::~~Inertial () [virtual]

Destructor.

10.60.3 Member Function Documentation

10.60.3.1 const math::Vector3& gazebo::physics::Inertial::GetCoG () const [inline]

Get the center of gravity.

Returns

The center of gravity.

10.60.3.2 double gazebo::physics::Inertial::GetIXX () const

Get IXX.

Returns

IXX value

10.60.3.3 double gazebo::physics::Inertial::GetIXY () const

Get IXY.

Returns

IXY value

10.60.3.4 `double gazebo::physics::Inertial::GetIXZ () const`

Get IXZ.

Returns

IXZ value

10.60.3.5 `double gazebo::physics::Inertial::GetIYY () const`

Get IYY.

Returns

IYY value

10.60.3.6 `double gazebo::physics::Inertial::GetIYZ () const`

Get IYZ.

Returns

IYZ value

10.60.3.7 `double gazebo::physics::Inertial::GetIZZ () const`

Get IZZ.

Returns

IZZ value

10.60.3.8 `double gazebo::physics::Inertial::GetMass () const`

Get the mass.

10.60.3.9 `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. 398) frame.

Returns

The inertial pose.

10.60.3.10 `math::Vector3 gazebo::physics::Inertial::GetPrincipalMoments () const`

Get the principal moments of inertia (Ixx, Iyy, Izz).

Returns

The principal moments.

10.60.3.11 `math::Vector3 gazebo::physics::Inertial::GetProductsOfInertia () const`

Get the products of inertia (Ixy, Iyx, Iyz).

Returns

The products of inertia.

10.60.3.12 `void gazebo::physics::Inertial::Load (sdf::ElementPtr _sdf)`

Load from SDF values.

Parameters

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

10.60.3.13 `Inertial gazebo::physics::Inertial::operator+ (const Inertial & _inertial) const`

Addition operator.

Parameters

<code>in</code>	<code>_inertial</code>	Inertial (p. 357) to add.
-----------------	------------------------	----------------------------------

Returns

The result of the addition.

10.60.3.14 `const Inertial& gazebo::physics::Inertial::operator+= (const Inertial & _inertial)`

Addition equal operator.

Parameters

<code>in</code>	<code>_inertial</code>	Inertial (p. 357) to add.
-----------------	------------------------	----------------------------------

Returns

Reference to this object.

10.60.3.15 `Inertial& gazebo::physics::Inertial::operator= (const Inertial & _inertial)`

Equal operator.

Parameters

<code>in</code>	<code><i>_inertial</i></code>	<code>Inertial</code> (p. 357) to copy.
-----------------	-------------------------------	---

Returns

Reference to this object.

10.60.3.16 `void gazebo::physics::Inertial::ProcessMsg (const msgs::Inertial & _msg)`

Update parameters from a message.

Parameters

<code>in</code>	<code><i>_msg</i></code>	Message to read
-----------------	--------------------------	-----------------

10.60.3.17 `void gazebo::physics::Inertial::Reset ()`

Reset all the mass properties.

10.60.3.18 `void gazebo::physics::Inertial::Rotate (const math::Quaternion & _rot)`

Rotate this mass.

Parameters

<code>in</code>	<code><i>_rot</i></code>	Rotation amount.
-----------------	--------------------------	------------------

10.60.3.19 `void gazebo::physics::Inertial::SetCoG (double _cx, double _cy, double _cz)`

Set the center of gravity.

Parameters

<code>in</code>	<code><i>_cx</i></code>	X position.
<code>in</code>	<code><i>_cy</i></code>	Y position.
<code>in</code>	<code><i>_cz</i></code>	Z position.

10.60.3.20 `void gazebo::physics::Inertial::SetCoG (const math::Vector3 & _center)`

Set the center of gravity.

Parameters

in	<code>_center</code>	Center of the gravity.
----	----------------------	------------------------

10.60.3.21 void gazebo::physics::Inertial::SetInertiaMatrix (double *.ixx*, double *.iyy*, double *.izz*, double *.ixy*, double *.ixz*, double *iyz*)

Set the mass matrix.

Parameters

in	<code>_ixx</code>	X second moment of inertia about x axis.
in	<code>_iyy</code>	Y second moment of inertia about y axis.
in	<code>_izz</code>	Z second moment of inertia about z axis.
in	<code>_ixy</code>	XY inertia.
in	<code>_ixz</code>	XZ inertia.
in	<code>_iyz</code>	YZ inertia.

10.60.3.22 void gazebo::physics::Inertial::SetIXX (double *.v*)

Set IXX.

Parameters

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

10.60.3.23 void gazebo::physics::Inertial::SetIXY (double *.v*)

Set IXY.

Parameters

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

10.60.3.24 void gazebo::physics::Inertial::SetIXZ (double *.v*)

Set IXZ.

Parameters

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

10.60.3.25 void gazebo::physics::Inertial::SetIYY (double *.v*)

Set IYY.

Parameters

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

10.60.3.26 void gazebo::physics::Inertial::SetIYZ (double *_v*)

Set IYZ.

Parameters

<i>in</i>	<i>_v</i>	IXX value
-----------	-----------	-----------

10.60.3.27 void gazebo::physics::Inertial::SetIZZ (double *_v*)

Set IZZ.

Parameters

<i>in</i>	<i>_v</i>	IZZ value
-----------	-----------	-----------

10.60.3.28 void gazebo::physics::Inertial::SetMass (double *m*)

Set the mass.

10.60.3.29 void gazebo::physics::Inertial::UpdateParameters (sdf::ElementPtr *_sdf*)

update the parameters using new sdf values.

Parameters

<i>in</i>	<i>_sdf</i>	Update values from.
-----------	-------------	---------------------

10.60.4 Friends And Related Function Documentation

10.60.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::physics::Inertial & *_inertial*) [friend]

Output operator.

Parameters

<i>in</i>	<i>_out</i>	Output stream.
<i>in</i>	<i>_inertial</i>	Inertial (p. 357) object to output.

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

- **Inertial.hh**

10.61 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.

10.61.1 Detailed Description

Manages boost::asio IO.

10.61.2 Constructor & Destructor Documentation

10.61.2.1 gazebo::transport::IOManager::IOManager ()

Constructor.

10.61.2.2 gazebo::transport::IOManager::~~IOManager ()

Destructor.

10.61.3 Member Function Documentation

10.61.3.1 void gazebo::transport::IOManager::DecCount ()

Decrement the event count by 1.

10.61.3.2 unsigned int gazebo::transport::IOManager::GetCount () const

Get the event count.

Returns

The event count

10.61.3.3 `boost::asio::io_service& gazebo::transport::IOManager::GetIO ()`

Get handle to boost::asio IO service.

Returns

Handle to boost::asio IO service

10.61.3.4 `void gazebo::transport::IOManager::IncCount ()`

Increment the event count by 1.

10.61.3.5 `void gazebo::transport::IOManager::Stop ()`

Stop the IO service.

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

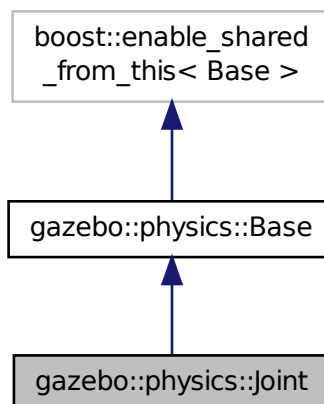
- **IOManager.hh**

10.62 `gazebo::physics::Joint` Class Reference

Base (p. 125) 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. 366) attribute types.

Public Member Functions

- **Joint** (**BasePtr** _parent)
Constructor.
- virtual \sim **Joint** ()
Destructor.
- 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 **FillJointMsg** (msgs::Joint &_msg) **GAZEBO_DEPRECATED**
DEPRECATED.
- 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 **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. 398) due to the existence of this **Joint** (p. 366).*
- virtual **math::Vector3 GetLinkTorque** (unsigned int _index) const =0
*Get the torque applied to the center of mass of a **physics::Link** (p. 398) due to the existence of this **Joint** (p. 366).*

- **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. 366).*
- virtual void **Load** (**sdf::ElementPtr** _sdf)
*Load **physics::Joint** (p. 366) from a SDF **sdf::Element** (p. 258).*
- 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. 366) is static, Gazebo stores the state of this **Joint** (p. 366) as a scalar inside the **Joint** (p. 366) class, so this call will NOT move the joint dynamically for a static **Model** (p. 460).*
- virtual void **SetAttribute** (**Attribute** _attr, int _index, double _value) **GAZEBO_DEPRECATED=0**
Set a parameter for the joint.
- 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. 366).*
- 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.
- **LinkPtr childLink**
The first link this joint connects to.
- double **damping_coefficient**
joint damping_coefficient
- **ModelPtr model**
Pointer to the parent model.
- **LinkPtr parentLink**
The second link this joint connects to.

10.62.1 Detailed Description

Base (p. 125) class for all joints.

10.62.2 Member Enumeration Documentation

10.62.2.1 enum gazebo::physics::Joint::Attribute

Joint (p. 366) 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.

10.62.3 Constructor & Destructor Documentation

10.62.3.1 `gazebo::physics::Joint::Joint (BasePtr _parent) [explicit]`

Constructor.

Parameters

in	Joint (p. 366)	parent
----	-----------------------	--------

10.62.3.2 `virtual gazebo::physics::Joint::~~Joint () [virtual]`

Destructor.

10.62.4 Member Function Documentation

10.62.4.1 `virtual bool gazebo::physics::Joint::AreConnected (LinkPtr _one, LinkPtr _two) const [pure virtual]`

Determines if the two bodies are connected by a joint.

Parameters

in	<code>_one</code>	First link.
in	<code>_two</code>	Second link.

Returns

True if the two links are connected by a joint.

10.62.4.2 `virtual void gazebo::physics::Joint::Attach (LinkPtr _parent, LinkPtr _child) [virtual]`

Attach the two bodies with this joint.

Parameters

in	<code>_parent</code>	Parent link.
in	<code>_child</code>	Child link.

10.62.4.3 `template<typename T > event::ConnectionPtr gazebo::physics::Joint::ConnectJointUpdate (T _subscriber) [inline]`

Connect a boost::slot to 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().

10.62.4.4 virtual void gazebo::physics::Joint::Detach () [virtual]

Detach this joint from all links.

10.62.4.5 void gazebo::physics::Joint::DisconnectJointUpdate (event::ConnectionPtr & _conn) [inline]

Disconnect a boost::slot the the joint update signal.

Parameters

in	_conn	Connection to disconnect.
----	-------	---------------------------

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

10.62.4.6 void gazebo::physics::Joint::FillJointMsg (msgs::Joint & _msg)

DEPRECATED.

Parameters

out	_msg	Message to fill with joint's properties
-----	------	---

See Also

Joint::FillMsg (p. 371)

10.62.4.7 void gazebo::physics::Joint::FillMsg (msgs::Joint & _msg)

Fill a joint message.

Parameters

out	_msg	Message to fill with this joint's properties.
-----	------	---

10.62.4.8 virtual math::Vector3 gazebo::physics::Joint::GetAnchor (int _index) const [pure virtual]

Get the anchor point.

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

Anchor value for the axis.

10.62.4.9 `math::Angle gazebo::physics::Joint::GetAngle (int _index) const`

Get the angle of rotation of an axis(index)

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Angle of the axis.

10.62.4.10 `virtual unsigned int gazebo::physics::Joint::GetAngleCount () const [pure virtual]`

Get the angle count.

Returns

The number of DOF for the joint.

10.62.4.11 `virtual math::Angle gazebo::physics::Joint::GetAngleImpl (int _index) const [protected], [pure virtual]`

Get the angle of an axis helper function.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Angle of the axis.

10.62.4.12 `LinkPtr gazebo::physics::Joint::GetChild () const`

Get the child link.

Returns

Pointer to the child link.

10.62.4.13 `virtual double gazebo::physics::Joint::GetForce (int _index) [virtual]`

Todo : not yet implemented. Get the internal forces at a this **Joint** (p.366). 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	_index	Index of the axis.
----	--------	--------------------

Returns

The force applied to an axis.

10.62.4.14 `virtual math::Vector3 gazebo::physics::Joint::GetGlobalAxis (int _index) const` [pure virtual]

Get the axis of rotation in global coordinate frame.

Parameters

in	_index	Index of the axis to get.
----	--------	---------------------------

Returns

Axis value for the provided index.

10.62.4.15 `virtual math::Angle gazebo::physics::Joint::GetHighStop (int _index)` [pure virtual]

Get the high stop of an axis(index).

Parameters

in	_index	Index of the axis.
----	--------	--------------------

Returns

Angle of the high stop value.

10.62.4.16 `virtual LinkPtr gazebo::physics::Joint::GetJointLink (int _index) const` [pure virtual]

Get the link to which the joint is attached according the _index.

Parameters

in	_index	Index of the link to retrieve.
----	--------	--------------------------------

Returns

Pointer to the request link. NULL if the index was invalid.

10.62.4.17 `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. 398) due to the existence of this **Joint** (p. 366).

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>	The index of the link(0 or 1).
-----------------	--------------------	--------------------------------

Returns

Force applied to the link.

10.62.4.18 `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. 398) due to the existence of this **Joint** (p. 366).

Note that the unit of torque should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons-Meters. If using imperial units (sorry), then unit of force is lb-force-inches not (lb-mass-inches), etc.

Parameters

<code>in</code>	<code>index</code>	The index of the link(0 or 1)
-----------------	--------------------	-------------------------------

Returns

Torque applied to the link.

10.62.4.19 `math::Vector3 gazebo::physics::Joint::GetLocalAxis (int index) const`

Get the axis of rotation.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to get.
-----------------	---------------------	---------------------------

Returns

Axis value for the provided index.

10.62.4.20 `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.

10.62.4.21 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

<i>in</i>	<i>_index</i>	Index of the axis.
-----------	---------------	--------------------

Returns

The maximum force.

10.62.4.22 LinkPtr gazebo::physics::Joint::GetParent () const

Get the parent link.

Returns

Pointer to the parent link.

10.62.4.23 virtual double gazebo::physics::Joint::GetVelocity (int *_index*) const [pure virtual]

Get the rotation rate of an axis(index)

Parameters

<i>in</i>	<i>_index</i>	Index of the axis.
-----------	---------------	--------------------

Returns

The rotaional velocity of the joint axis.

10.62.4.24 virtual void gazebo::physics::Joint::Init () [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Base** (p. 132).

10.62.4.25 void gazebo::physics::Joint::Load (LinkPtr *_parent*, LinkPtr *_child*, const math::Pose & *_pose*)

Set pose, parent and child links of a **physics::Joint** (p. 366).

Parameters

in	<code>_parent</code>	Parent link.
in	<code>_child</code>	Child link.
in	<code>_pose</code>	Pose of the link.

10.62.4.26 virtual void gazebo::physics::Joint::Load (sdf::ElementPtr *sdf*) [virtual]

Load **physics::Joint** (p. 366) from a SDF **sdf::Element** (p. 258).

Parameters

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

Reimplemented from **gazebo::physics::Base** (p. 132).

10.62.4.27 virtual void gazebo::physics::Joint::Reset () [virtual]

Reset the joint.

Reimplemented from **gazebo::physics::Base** (p. 134).

10.62.4.28 virtual void gazebo::physics::Joint::SetAnchor (int *_index*, const math::Vector3 & *_anchor*) [pure virtual]

Set the anchor point.

Parameters

in	<code>_index</code>	Indx of the axis.
in	<code>_anchor</code>	Anchor value.

10.62.4.29 void gazebo::physics::Joint::SetAngle (int *_index*, math::Angle *_angle*)

If the **Joint** (p. 366) is static, Gazebo stores the state of this **Joint** (p. 366) as a scalar inside the **Joint** (p. 366) class, so this call will NOT move the joint dynamically for a static **Model** (p. 460).

But if this **Model** (p. 460) is not static, then it is updated dynamically, all the conected children **Link** (p. 398)'s are moved as a result of the **Joint** (p. 366) angle setting. Dynamic **Joint** (p. 366) angle update is accomplished by calling **JointController::SetJointPosition** (p. 381).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_angle</code>	Angle to set the joint to.

10.62.4.30 virtual void gazebo::physics::Joint::SetAttribute (Attribute *_attr*, int *_index*, double *_value*) [pure virtual]

Set a parameter for the joint.

Parameters

in	<code>_attr</code>	Attribute to set.
in	<code>_index</code>	Index of the axis.
in	<code>_value</code>	Value of the attribute.

10.62.4.31 `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)** (p. 376)

Parameters

in	<code>_key</code>	String key.
in	<code>_index</code>	Index of the axis.
in	<code>_value</code>	Value of the attribute.

10.62.4.32 `virtual void gazebo::physics::Joint::SetAxis (int _index, const math::Vector3 & _axis)` `[pure virtual]`

Set the axis of rotation.

Parameters

in	<code>_index</code>	Index of the axis to set.
in	<code>_axis</code>	Axis value.

10.62.4.33 `virtual void gazebo::physics::Joint::SetDamping (int _index, double _damping)` `[pure virtual]`

Set the joint damping.

Parameters

in	<code>_index</code>	Index of the axis to set.
in	<code>_damping</code>	Damping value for the axis.

10.62.4.34 `virtual void gazebo::physics::Joint::SetForce (int _index, double _force)` `[virtual]`

Set the force applied to this **physics::Joint** (p. 366).

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	<code>_index</code>	Index of the axis.
in	<code>_force</code>	Force value.

10.62.4.35 `virtual void gazebo::physics::Joint::SetHighStop (int _index, const math::Angle & _angle) [pure virtual]`

Set the high stop of an axis(index).

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_angle</i></code>	High stop angle.

10.62.4.36 `virtual void gazebo::physics::Joint::SetLowStop (int _index, const math::Angle & _angle) [pure virtual]`

Set the low stop of an axis(index).

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_angle</i></code>	Low stop angle.

10.62.4.37 `virtual void gazebo::physics::Joint::SetMaxForce (int _index, double _force) [pure virtual]`

Set the max allowed force of an axis(index).

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

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_force</i></code>	Maximum force that can be applied to the axis.

10.62.4.38 `void gazebo::physics::Joint::SetModel (ModelPtr _model)`

Set the model this joint belongs too.

Parameters

<code>in</code>	<code><i>_model</i></code>	Pointer to a model.
-----------------	----------------------------	---------------------

10.62.4.39 `void gazebo::physics::Joint::SetState (const JointState & _state)`

Set the joint state.

Parameters

<code>in</code>	<code><i>_state</i></code>	Joint (p. 366) state
-----------------	----------------------------	-----------------------------

10.62.4.40 `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.

10.62.4.41 `void gazebo::physics::Joint::Update () [virtual]`

Update the joint.

Reimplemented from `gazebo::physics::Base` (p. 135).

10.62.4.42 `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. 135).

10.62.5 Member Data Documentation

10.62.5.1 `LinkPtr gazebo::physics::Joint::anchorLink [protected]`

Anchor link.

10.62.5.2 `math::Vector3 gazebo::physics::Joint::anchorPos [protected]`

Anchor pose.

10.62.5.3 `LinkPtr gazebo::physics::Joint::childLink [protected]`

The first link this joint connects to.

10.62.5.4 `double gazebo::physics::Joint::damping_coefficient [protected]`

joint damping_coefficient

10.62.5.5 `ModelPtr gazebo::physics::Joint::model [protected]`

Pointer to the parent model.

10.62.5.6 `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**

10.63 gazebo::physics::JointController Class Reference

A class for manipulating **physics::Joint** (p. 366).

```
#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. 366) by name.*
- void **SetJointPosition** (**JointPtr** _joint, double _position)
*Set the positions of a **Joint** (p. 366) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 692) and radians for **HingeJoint** (p. 347), etc.*
- void **SetJointPositions** (const std::map< std::string, double > &_jointPositions)
*Set the positions of a set of **Joint** (p. 366)'s.*
- void **Update** ()
Update the joint control.

10.63.1 Detailed Description

A class for manipulating **physics::Joint** (p. 366).

10.63.2 Constructor & Destructor Documentation

10.63.2.1 gazebo::physics::JointController::JointController (**ModelPtr** _model) [explicit]

Constructor.

Parameters

in	_model	Model (p. 460) that uses this joint controller.
----	--------	--

10.63.3 Member Function Documentation

10.63.3.1 void gazebo::physics::JointController::AddJoint (**JointPtr** _joint)

Add a joint to control.

Parameters

in	_joint	Joint (p. 366) to control.
----	--------	-----------------------------------

10.63.3.2 void gazebo::physics::JointController::Reset ()

Reset all commands.

10.63.3.3 void gazebo::physics::JointController::SetJointPosition (const std::string & *_name*, double *_position*)

Set the positions of a **Joint** (p. 366) by name.

See Also

JointController::SetJointPosition(JointPtr, double) (p. 381)

10.63.3.4 void gazebo::physics::JointController::SetJointPosition (JointPtr *_joint*, double *_position*)

Set the positions of a **Joint** (p. 366) by name. The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 692) and radians for **HingeJoint** (p. 347), etc.

Implementation: In order to change the position of a **Joint** (p. 366) inside a **Model** (p. 460), this call must recursively crawl through all the connected children **Link** (p. 398)'s in this **Model** (p. 460), and update each **Link** (p. 398) Pose affected by this **Joint** (p. 366) angle update. Warning: There is no constraint satisfaction being done here, traversal through the kinematic graph has unexpected behavior if you try to set the joint position of a link inside a loop structure.

Parameters

in	<i>_joint</i>	Joint (p. 366) to set.
in	<i>_position</i>	Position of the joint.

10.63.3.5 void gazebo::physics::JointController::SetJointPositions (const std::map< std::string, double > & *_jointPositions*)

Set the positions of a set of **Joint** (p. 366)'s.

See Also

JointController::SetJointPosition(JointPtr, double) (p. 381)

10.63.3.6 void gazebo::physics::JointController::Update ()

Update the joint control.

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

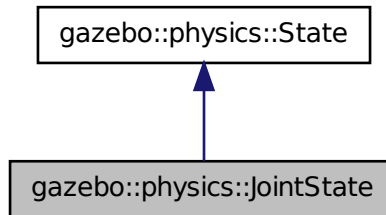
- **JointController.hh**

10.64 gazebo::physics::JointState Class Reference

keeps track of state of a **physics::Joint** (p. 366)

```
#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.
- **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

10.64.1 Detailed Description

keeps track of state of a **physics::Joint** (p. 366)

10.64.2 Constructor & Destructor Documentation

10.64.2.1 gazebo::physics::JointState::JointState ()

Default constructor.

10.64.2.2 gazebo::physics::JointState::JointState (JointPtr *joint*) [explicit]

Constructor.

Parameters

in	<i>_joint</i>	Joint (p. 366) to get the state of.
----	---------------	--

10.64.2.3 gazebo::physics::JointState::JointState (const sdf::ElementPtr *sdf*) [explicit]

Constructor.

Build a **JointState** (p. 381) from SDF data

Parameters

in	<i>_sdf</i>	SDF data to load a joint state from.
----	-------------	--------------------------------------

10.64.2.4 virtual gazebo::physics::JointState::~~JointState () [virtual]

Destructor.

10.64.3 Member Function Documentation

10.64.3.1 math::Angle gazebo::physics::JointState::GetAngle (unsigned int *_axis*) const

Get the joint angle.

Parameters

in	<i>_axis</i>	The axis index.
----	--------------	-----------------

Returns

Angle of the axis.

Exceptions

<i>common::Exception</i> (p. 303)	When <code>_axis</code> is invalid.
---	-------------------------------------

10.64.3.2 `unsigned int gazebo::physics::JointState::GetAngleCount () const`

Get the number of angles.

Returns

The number of angles.

10.64.3.3 `const std::vector<math::Angle>& gazebo::physics::JointState::GetAngles () const`

Get the angles.

Returns

Vector of angles.

10.64.3.4 `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.

10.64.3.5 `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. 704).

10.64.3.6 `JointState gazebo::physics::JointState::operator+ (const JointState & state) const`

Addition operator.

Parameters

<code>in</code>	<code>_pt</code>	A state to add.
-----------------	------------------	-----------------

Returns

The resulting state.

10.64.3.7 JointState gazebo::physics::JointState::operator- (const JointState & _state) const

Subtraction operator.

Parameters

<i>in</i>	<i>_pt</i>	A state to subtract.
-----------	------------	----------------------

Returns

The resulting state.

10.64.3.8 JointState& gazebo::physics::JointState::operator= (const JointState & _state)

Assignment operator.

Parameters

<i>in</i>	<i>_state</i>	State (p. 702) value
-----------	---------------	-----------------------------

Returns

this

10.64.4 Friends And Related Function Documentation**10.64.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::physics::JointState & _state) [friend]**

Stream insertion operator.

Parameters

<i>in</i>	<i>_out</i>	output stream.
<i>in</i>	<i>_state</i>	Joint (p. 366) state to output.

Returns

The stream.

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

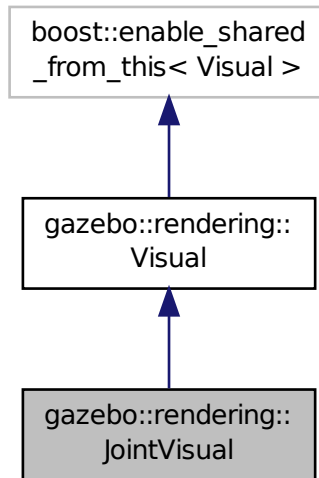
- **JointState.hh**

10.65 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

10.65.1 Detailed Description

Visualization for joints.

10.65.2 Constructor & Destructor Documentation

10.65.2.1 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

10.65.2.2 `virtual gazebo::rendering::JointVisual::~JointVisual() [virtual]`

Destructor.

10.65.3 Member Function Documentation

10.65.3.1 `void gazebo::rendering::JointVisual::Load(ConstJointPtr & _msg)`

Load the visual based on a message.

Parameters

in	<code>_msg</code>	Joint message
----	-------------------	---------------

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

- **JointVisual.hh**

10.66 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.

10.66.1 Detailed Description

Wrench information from a joint.

These are forces and torques on parent and child Links, relative to the **Link** (p. 398)'s center of mass.

10.66.2 Member Function Documentation

10.66.2.1 `JointWrench& gazebo::physics::JointWrench::operator= (const JointWrench & _wrench) [inline]`

Operator =.

Parameters

<code>in</code>	<code>_wrench</code>	Joint (p. 366) wrench to set from.
-----------------	----------------------	---

Returns

`*this`

References `body1Force`, `body1Torque`, `body2Force`, and `body2Torque`.

10.66.3 Member Data Documentation

10.66.3.1 `math::Vector3 gazebo::physics::JointWrench::body1Force`

Force on the first link.

Referenced by `operator=()`.

10.66.3.2 `math::Vector3 gazebo::physics::JointWrench::body1Torque`

Torque on the first link.

Referenced by `operator=()`.

10.66.3.3 `math::Vector3 gazebo::physics::JointWrench::body2Force`

Force on the second link.

Referenced by `operator=()`.

10.66.3.4 `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:

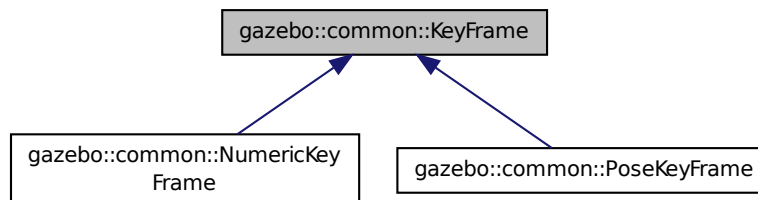
- **JointWrench.hh**

10.67 gazebo::common::KeyFrame Class Reference

A key frame in an animation.

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

Inheritance diagram for gazebo::common::KeyFrame:



Public Member Functions

- **KeyFrame** (double *_time*)
Constructor.
- virtual **~KeyFrame** ()
Destructor.
- double **GetTime** () const
Get the time of the keyframe.

Protected Attributes

- double **time**
time of key frame

10.67.1 Detailed Description

A key frame in an animation.

10.67.2 Constructor & Destructor Documentation

10.67.2.1 gazebo::common::KeyFrame::KeyFrame (double *_time*)

Constructor.

Parameters

in	<i>_time</i>	Time (p. 732) of the keyframe in seconds
----	--------------	---

10.67.2.2 `virtual gazebo::common::KeyFrame::~~KeyFrame () [virtual]`

Destructor.

10.67.3 Member Function Documentation

10.67.3.1 `double gazebo::common::KeyFrame::GetTime () const`

Get the time of the keyframe.

Returns

the time

10.67.4 Member Data Documentation

10.67.4.1 `double gazebo::common::KeyFrame::time [protected]`

time of key frame

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

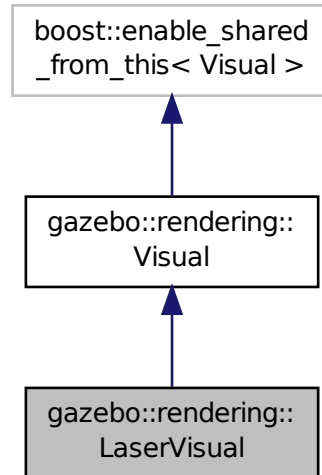
- **KeyFrame.hh**

10.68 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

10.68.1 Detailed Description

Visualization for laser data.

10.68.2 Constructor & Destructor Documentation

10.68.2.1 gazebo::rendering::LaserVisual::LaserVisual (const std::string & _name, **VisualPtr** _vis, const std::string & _topicName)

Constructor.

Parameters

in	<code>_name</code>	Name of the visual.
in	<code>_vis</code>	Pointer to the parent Visual (p. 828).
in	<code>_topicName</code>	Name of the topic that has laser data.

10.68.2.2 virtual gazebo::rendering::LaserVisual::~~LaserVisual () [virtual]

Destructor.

10.68.3 Member Function Documentation

10.68.3.1 virtual void gazebo::rendering::LaserVisual::SetEmissive (const common::Color & _color) [virtual]

Documentation inherited from parent.

Reimplemented from **gazebo::rendering::Visual** (p. 842).

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

- **LaserVisual.hh**

10.69 gazebo::rendering::Light Class Reference

A light source.

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

Public Member Functions

- **Light (ScenePtr _scene)**
Constructor.
- virtual **~Light ()**
Destructor.
- void **FillMsg** (msgs::Light &_msg) const
Fill the contents of a light message.
- **common::Color GetDiffuseColor ()** const
Get the diffuse color.
- **math::Vector3 GetDirection ()** const
Get the direction.
- std::string **GetName ()** const
Get the name of the visual.
- **math::Vector3 GetPosition ()** const
Get the position of the light.
- **common::Color GetSpecularColor ()** const
Get the specular color.
- std::string **GetType ()** const
Get the type of the light.

- void **Load** (sdf::ElementPtr _sdf)
Load the light using a set of SDF parameters.
- void **Load** ()
Load the light using default parameters.
- void **LoadFromMsg** (ConstLightPtr &_msg)
Load from a light message.
- void **SetAttenuation** (double _constant, double _linear, double _quadratic)
Set the attenuation.
- void **SetCastShadows** (const bool &_cast)
Set cast shadows.
- void **SetDiffuseColor** (const common::Color &_color)
Set the diffuse color.
- void **SetDirection** (const math::Vector3 &_dir)
Set the direction.
- void **SetLightType** (const std::string &_type)
Set the light type.
- void **SetName** (const std::string &_name)
Set the name of the visual.
- void **SetPosition** (const math::Vector3 &_p)
Set the position of the light.
- void **SetRange** (const double &_range)
Set the range.
- virtual bool **SetSelected** (bool _s)
Set whether this entity has been selected by the user through the gui.
- void **SetSpecularColor** (const common::Color &_color)
Set the specular color.
- void **SetSpotFalloff** (const double &_value)
Set the spot light falloff.
- void **SetSpotInnerAngle** (const double &_angle)
Set the spot light inner angle.
- void **SetSpotOuterAngle** (const double &_angle)
Set the spot light outer angle.
- void **ShowVisual** (bool _s)
Set whether to show the visual.
- void **ToggleShowVisual** ()
- void **UpdateFromMsg** (ConstLightPtr &_msg)
Update a light source from a message.

Protected Member Functions

- virtual void **OnPoseChange** ()
On pose change callback.

10.69.1 Detailed Description

A light source.

There are three types of lights: Point, Spot, and Directional. This class encapsulates all three. Point lights are light light bulbs, spot lights project a cone of light, and directional lights are light sun light.

10.69.2 Constructor & Destructor Documentation

10.69.2.1 gazebo::rendering::Light::Light (ScenePtr *_scene*)

Constructor.

Parameters

in	<i>_scene</i>	Pointer to the scene that contains the Light (p. 392).
----	---------------	---

10.69.2.2 virtual gazebo::rendering::Light::~~Light () [virtual]

Destructor.

10.69.3 Member Function Documentation

10.69.3.1 void gazebo::rendering::Light::FillMsg (msgs::Light & *_msg*) const

Fill the contents of a light message.

Parameters

out	<i>_msg</i>	Message to fill.
-----	-------------	------------------

10.69.3.2 common::Color gazebo::rendering::Light::GetDiffuseColor () const

Get the diffuse color.

Returns

The light's diffuse color.

10.69.3.3 math::Vector3 gazebo::rendering::Light::GetDirection () const

Get the direction.

Returns

The light's direction.

10.69.3.4 std::string gazebo::rendering::Light::GetName () const

Get the name of the visual.

Returns

The light's name.

10.69.3.5 `math::Vector3 gazebo::rendering::Light::GetPosition () const`

Get the position of the light.

Returns

The position of the light

10.69.3.6 `common::Color gazebo::rendering::Light::GetSpecularColor () const`

Get the specular color.

Returns

The specular color

10.69.3.7 `std::string gazebo::rendering::Light::GetType () const`

Get the type of the light.

Returns

The light type: "point", "spot", "directional".

10.69.3.8 `void gazebo::rendering::Light::Load (sdf::ElementPtr _sdf)`

Load the light using a set of SDF parameters.

Parameters

in	_sdf	Pointer to the SDF containing the Light (p. 392) description.
----	------	--

10.69.3.9 `void gazebo::rendering::Light::Load ()`

Load the light using default parameters.

10.69.3.10 `void gazebo::rendering::Light::LoadFromMsg (ConstLightPtr & _msg)`

Load from a light message.

Parameters

in	_msg	Containing the light information.
----	------	-----------------------------------

10.69.3.11 `virtual void gazebo::rendering::Light::OnPoseChange () [inline], [protected], [virtual]`

On pose change callback.

10.69.3.12 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

10.69.3.13 void gazebo::rendering::Light::SetCastShadows (const bool & *_cast*)

Set cast shadows.

Parameters

in	<i>_cast</i>	Set to true to cast shadows.
----	--------------	------------------------------

10.69.3.14 void gazebo::rendering::Light::SetDiffuseColor (const common::Color & *_color*)

Set the diffuse color.

Parameters

in	<i>_color</i>	Light (p. 392) diffuse color.
----	---------------	--------------------------------------

10.69.3.15 void gazebo::rendering::Light::SetDirection (const math::Vector3 & *_dir*)

Set the direction.

Parameters

in	<i>_dir</i>	Set the light's direction. Only applicable to spot and directional lights.
----	-------------	--

10.69.3.16 void gazebo::rendering::Light::SetLightType (const std::string & *_type*)

Set the light type.

Parameters

in	<i>_type</i>	The light type: "point", "spot", "directional"
----	--------------	--

10.69.3.17 void gazebo::rendering::Light::SetName (const std::string & *_name*)

Set the name of the visual.

Parameters

in	<i>_name</i>	Name of the light source.
----	--------------	---------------------------

10.69.3.18 void gazebo::rendering::Light::SetPosition (const math::Vector3 & _p)

Set the position of the light.

Parameters

in	_p	New position for the light
----	----	----------------------------

10.69.3.19 void gazebo::rendering::Light::SetRange (const double & _range)

Set the range.

Parameters

in	_range	Rage of the light in meters.
----	--------	------------------------------

10.69.3.20 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.
----	----	---

10.69.3.21 void gazebo::rendering::Light::SetSpecularColor (const common::Color & _color)

Set the specular color.

Parameters

in	_color	The specular color
----	--------	--------------------

10.69.3.22 void gazebo::rendering::Light::SetSpotFalloff (const double & _value)

Set the spot light falloff.

Parameters

in	_value	Falloff value
----	--------	---------------

10.69.3.23 void gazebo::rendering::Light::SetSpotInnerAngle (const double & _angle)

Set the spot light inner angle.

Parameters

in	_angle	Inner angle in radians
----	--------	------------------------

10.69.3.24 void gazebo::rendering::Light::SetSpotOuterAngle (const double & *_angle*)

Set the spot light outer angle.

Parameters

in	<i>_angle</i>	Outer angle in radians
----	---------------	------------------------

10.69.3.25 void gazebo::rendering::Light::ShowVisual (bool *_s*)

Set whether to show the visual.

Parameters

in	<i>_s</i>	Set to true to draw a representation of the light.
----	-----------	--

10.69.3.26 void gazebo::rendering::Light::ToggleShowVisual ()

10.69.3.27 void gazebo::rendering::Light::UpdateFromMsg (ConstLightPtr & *_msg*)

Update a light source from a message.

Parameters

in	<i>_msg</i>	Light (p. 392) message to update from
----	-------------	--

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

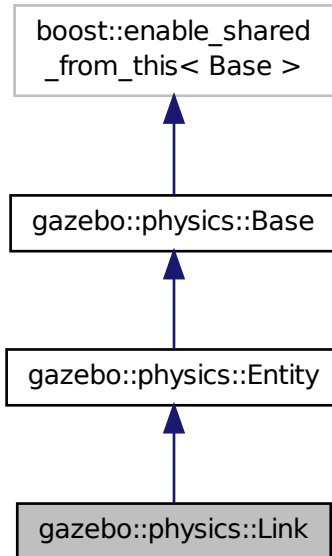
- **Light.hh**

10.70 gazebo::physics::Link Class Reference

Link (p. 398) 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. 398) as a parent **Link** (p. 398).*
- 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. 398) as a child **Link** (p. 398).*
- 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 **FillLinkMsg** (msgs::Link &_msg) **GAZEBO_DEPRECATED**
DEPRECATED.
- 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. 398) as a parent **Link** (p. 398).*
- void **RemoveParentJoint** (JointPtr _joint)
*Remove Joints that have this **Link** (p. 398) as a child **Link** (p. 398).*
- 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. 357) properties.*
- **math::Vector3** **linearAccel**

Linear acceleration.

- `std::vector< std::string > visuals`

Link (p. 398) visual elements.

Additional Inherited Members

10.70.1 Detailed Description

Link (p. 398) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

10.70.2 Constructor & Destructor Documentation

10.70.2.1 gazebo::physics::Link::Link (EntityPtr _parent) [explicit]

Constructor.

Parameters

in	<code>_parent</code>	Parent of this link.
----	----------------------	----------------------

10.70.2.2 virtual gazebo::physics::Link::~~Link () [virtual]

Destructor.

10.70.3 Member Function Documentation

10.70.3.1 void gazebo::physics::Link::AddChildJoint (JointPtr _joint)

Joints that have this **Link** (p. 398) as a parent **Link** (p. 398).

Parameters

in	<code>_joint</code>	Joint (p. 366) that is a child of this link.
----	---------------------	---

10.70.3.2 virtual void gazebo::physics::Link::AddForce (const math::Vector3 & _force) [pure virtual]

Add a force to the body.

Parameters

in	<code>_force</code>	Force to add.
----	---------------------	---------------

10.70.3.3 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	<code>_force</code>	Force to add.
in	<code>_relPos</code>	Position on the link to add the force.

10.70.3.4 `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	<code>_force</code>	Force to add.
in	<code>_pos</code>	Position in global coord frame to add the force.

10.70.3.5 `void gazebo::physics::Link::AddParentJoint (JointPtr _joint)`

Joints that have this **Link** (p. 398) as a child **Link** (p. 398).

Parameters

in	<code>_joint</code>	Joint (p. 366) that is a parent of this link.
----	---------------------	--

10.70.3.6 `virtual void gazebo::physics::Link::AddRelativeForce (const math::Vector3 & _force) [pure virtual]`

Add a force to the body, components are relative to the body's own frame of reference.

Parameters

in	<code>_force</code>	Force to add.
----	---------------------	---------------

10.70.3.7 `virtual void gazebo::physics::Link::AddRelativeTorque (const math::Vector3 & _torque) [pure virtual]`

Add a torque to the body, components are relative to the body's own frame of reference.

Parameters

in	<code>_torque</code>	Torque value to add.
----	----------------------	----------------------

10.70.3.8 `virtual void gazebo::physics::Link::AddTorque (const math::Vector3 & _torque) [pure virtual]`

Add a torque to the body.

Parameters

in	<code>_torque</code>	Torque value to add to the link.
----	----------------------	----------------------------------

10.70.3.9 void gazebo::physics::Link::AttachStaticModel (ModelIPtr & *_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.

10.70.3.10 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().

10.70.3.11 void gazebo::physics::Link::DetachAllStaticModels ()

Detach all static models from this link.

10.70.3.12 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.
----	-------------------	--------------------------------------

10.70.3.13 void gazebo::physics::Link::DisconnectEnabled (event::ConnectionPtr & *_conn*) [inline]

Disconnect to the add entity signal.

Parameters

in	<i>_conn</i>	Connection pointer to disconnect.
----	--------------	-----------------------------------

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

10.70.3.14 void gazebo::physics::Link::FillLinkMsg (msgs::Link & *_msg*)

DEPRECATED.

10.70.3.15 `void gazebo::physics::Link::FillMsg (msgs::Link & _msg)`

Fill a link message.

Parameters

out	<code>_msg</code>	Message to fill
-----	-------------------	-----------------

10.70.3.16 `void gazebo::physics::Link::Fini () [virtual]`

Finalize the body.

Reimplemented from **`gazebo::physics::Entity`** (p. 269).

10.70.3.17 `double gazebo::physics::Link::GetAngularDamping () const`

Get the angular damping factor.

Returns

Angular damping.

10.70.3.18 `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. 269).

10.70.3.19 `Link_V gazebo::physics::Link::GetChildJointsLinks () const`

Returns a vector of children Links connected by joints.

Returns

A vector of children Links connected by joints.

10.70.3.20 `CollisionPtr gazebo::physics::Link::GetCollision (const std::string & _name)`

Get a child collision by name.

Parameters

in	<code>_name</code>	Name of the collision object.
----	--------------------	-------------------------------

Returns

Pointer to the collision, NULL if the name was not found.

10.70.3.21 CollisionPtr gazebo::physics::Link::GetCollision (unsigned int *_index*) const

Get a child collision by index.

Parameters

<i>in</i>	<i>_index</i>	Index of the collision object.
-----------	---------------	--------------------------------

Returns

Pointer to the collision, NULL if the name was not found.

10.70.3.22 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.

10.70.3.23 Collision_V gazebo::physics::Link::GetCollisions () const

Get all the child collisions.

Returns

A std::vector of all the child collisions.

10.70.3.24 virtual bool gazebo::physics::Link::GetEnabled () const [pure virtual]

Get whether this body is enabled in the physics engine.

Returns

True if the link is enabled.

10.70.3.25 virtual bool gazebo::physics::Link::GetGravityMode () [pure virtual]

Get the gravity mode.

Returns

True if gravity is enabled.

10.70.3.26 `InertiaPtr gazebo::physics::Link::GetInertia () const [inline]`

Get the inertia of the link.

Returns

Inertia of the link.

References inertial.

10.70.3.27 `virtual bool gazebo::physics::Link::GetKinematic () const [inline],[virtual]`

Implement this function.

Get whether this body is in the kinematic state.

Returns

True if the link is kinematic only.

10.70.3.28 `double gazebo::physics::Link::GetLinearDamping () const`

Get the linear damping factor.

Returns

Linear damping.

10.70.3.29 `ModelPtr gazebo::physics::Link::GetModel () const`

Get the model that this body belongs to.

Returns

Model (p. 460) that this body belongs to.

10.70.3.30 `Link_V gazebo::physics::Link::GetParentJointsLinks () const`

Returns a vector of parent Links connected by joints.

Returns

Vector of parent Links connected by joints.

10.70.3.31 `math::Vector3 gazebo::physics::Link::GetRelativeAngularAccel () const [virtual]`

Get the angular acceleration of the body.

Returns

Angular acceleration of the body.

Reimplemented from `gazebo::physics::Entity` (p. 270).

10.70.3.32 `math::Vector3 gazebo::physics::Link::GetRelativeAngularVel () const [virtual]`

Get the angular velocity of the body.

Returns

Angular velocity of the body.

Reimplemented from `gazebo::physics::Entity` (p. 271).

10.70.3.33 `math::Vector3 gazebo::physics::Link::GetRelativeForce () const`

Get the force applied to the body.

Returns

Force applied to the body.

10.70.3.34 `math::Vector3 gazebo::physics::Link::GetRelativeLinearAccel () const [virtual]`

Get the linear acceleration of the body.

Returns

Linear acceleration of the body.

Reimplemented from `gazebo::physics::Entity` (p. 271).

10.70.3.35 `math::Vector3 gazebo::physics::Link::GetRelativeLinearVel () const [virtual]`

Get the linear velocity of the body.

Returns

Linear velocity of the body.

Reimplemented from `gazebo::physics::Entity` (p. 271).

10.70.3.36 `math::Vector3 gazebo::physics::Link::GetRelativeTorque () const`

Get the torque applied to the body.

Returns

Torque applied to the body.

10.70.3.37 `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.

10.70.3.38 `unsigned int gazebo::physics::Link::GetSensorCount () const`

Get sensor count.

This will return the number of sensors created by the link when it was loaded. This function is commonly used with **Link::GetSensorName** (p. 410).

Returns

The number of sensors created by the link.

10.70.3.39 `std::string gazebo::physics::Link::GetSensorName (unsigned int _index) const`

Get sensor name.

Get the name of a sensor based on an index. The index should be in the range of 0...**Link::GetSensorCount()** (p. 410).

Note

A **Link** (p. 398) does not manage or maintain a pointer to a **sensors::Sensor** (p. 652). Access to a Sensor object is accomplished through the **sensors::SensorManager** (p. 662). This was done to separate the physics engine from the sensor engine.

Parameters

in	<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.

10.70.3.40 `math::Vector3 gazebo::physics::Link::GetWorldAngularAccel () const` [virtual]

Get the angular acceleration of the body in the world frame.

Returns

Angular acceleration of the body in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 272).

10.70.3.41 `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.

10.70.3.42 `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. 272).

10.70.3.43 `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.

10.70.3.44 `virtual void gazebo::physics::Link::Init ()` [virtual]

Initialize the body.

Reimplemented from `gazebo::physics::Base` (p. 132).

10.70.3.45 `virtual void gazebo::physics::Link::Load (sdf::ElementPtr _sdf)` [virtual]

Load the body based on an SDF element.

Parameters

in	_sdf	SDF parameters.
----	------	-----------------

Reimplemented from `gazebo::physics::Entity` (p. 273).

10.70.3.46 `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. 273).

10.70.3.47 `void gazebo::physics::Link::ProcessMsg (const msgs::Link & _msg)`

Update parameters from a message.

Parameters

in	_msg	Message to read.
----	------	------------------

10.70.3.48 `void gazebo::physics::Link::RemoveChildJoint (JointPtr _joint)`

Remove Joints that have this **Link** (p. 398) as a parent **Link** (p. 398).

Parameters

in	<i>_joint</i>	Joint (p. 366) that is a child of this link.
----	---------------	---

10.70.3.49 `void gazebo::physics::Link::RemoveParentJoint (JointPtr _joint)`

Remove Joints that have this **Link** (p. 398) as a child **Link** (p. 398).

Parameters

in	<i>_joint</i>	Joint (p. 366) that is a parent of this link.
----	---------------	--

10.70.3.50 `void gazebo::physics::Link::Reset () [virtual]`

Reset the link.

Reimplemented from **gazebo::physics::Entity** (p. 273).

10.70.3.51 `void gazebo::physics::Link::SetAngularAccel (const math::Vector3 & _accel)`

Set the angular acceleration of the body.

Parameters

in	<i>_accel</i>	Angular acceleration.
----	---------------	-----------------------

10.70.3.52 `virtual void gazebo::physics::Link::SetAngularDamping (double _damping) [pure virtual]`

Set the angular damping factor.

Parameters

in	<i>_damping</i>	Angular damping factor.
----	-----------------	-------------------------

10.70.3.53 `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.
----	-------------	-------------------

10.70.3.54 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.
----	-----------------	---

10.70.3.55 void gazebo::physics::Link::SetCollideMode (const std::string & *_mode*)

Set the collide mode of the body.

Parameters

in	<i>_mode</i>	Collision (p. 180) 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
----	--------------	--

10.70.3.56 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.
----	----------------	--

10.70.3.57 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.
----	---------------	--------------

10.70.3.58 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.
----	--------------	-------------------------

10.70.3.59 void gazebo::physics::Link::SetInertial (const InertialPtr & *_inertial*)

Set the mass of the link.

[in] *_inertial* **Inertial** (p. 357) value for the link.

10.70.3.60 `virtual void gazebo::physics::Link::SetKinematic (const bool & _kinematic) [virtual]`

Implement this function.

Set whether this body is in the kinematic state.

Parameters

<code>in</code>	<code><i>_kinematic</i></code>	True to make the link kinematic only.
-----------------	--------------------------------	---------------------------------------

10.70.3.61 `void gazebo::physics::Link::SetLaserRetro (float _retro)`

Set the laser retro reflectiveness.

Parameters

<code>in</code>	<code><i>_retro</i></code>	Retro value for all child collisions.
-----------------	----------------------------	---------------------------------------

10.70.3.62 `void gazebo::physics::Link::SetLinearAccel (const math::Vector3 & _accel)`

Set the linear acceleration of the body.

Parameters

<code>in</code>	<code><i>_accel</i></code>	Linear acceleration.
-----------------	----------------------------	----------------------

10.70.3.63 `virtual void gazebo::physics::Link::SetLinearDamping (double _damping) [pure virtual]`

Set the linear damping factor.

Parameters

<code>in</code>	<code><i>_damping</i></code>	Linear damping factor.
-----------------	------------------------------	------------------------

10.70.3.64 `virtual void gazebo::physics::Link::SetLinearVel (const math::Vector3 & _vel) [pure virtual]`

Set the linear velocity of the body.

Parameters

<code>in</code>	<code><i>_vel</i></code>	Linear velocity.
-----------------	--------------------------	------------------

10.70.3.65 `virtual bool gazebo::physics::Link::SetSelected (bool _set) [virtual]`

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

Parameters

<code>in</code>	<code><i>_set</i></code>	True to set the link as selected.
-----------------	--------------------------	-----------------------------------

Reimplemented from **gazebo::physics::Base** (p. 134).

10.70.3.66 virtual void gazebo::physics::Link::SetSelfCollide (bool *_collide*) [pure virtual]

Set whether this body will collide with others in the model.

Parameters

in	<i>_collide</i>	True to enable collisions.
----	-----------------	----------------------------

10.70.3.67 void gazebo::physics::Link::SetState (const LinkState & *_state*)

Set the current link state.

Parameters

in	<i>_state</i>	The state to set the link to.
----	---------------	-------------------------------

10.70.3.68 virtual void gazebo::physics::Link::SetTorque (const math::Vector3 & *_torque*) [pure virtual]

Set the torque applied to the body.

Parameters

in	<i>_torque</i>	Torque value.
----	----------------	---------------

10.70.3.69 virtual void gazebo::physics::Link::Update () [virtual]

Update the body.

Reimplemented from **gazebo::physics::Base** (p. 135).

10.70.3.70 virtual void gazebo::physics::Link::UpdateMass () [inline],[virtual]

Update the mass matrix.

10.70.3.71 virtual void gazebo::physics::Link::UpdateParameters (sdf::ElementPtr *_sdf*) [virtual]

Update the parameters using new sdf values.

Parameters

in	<i>_sdf</i>	SDF values to load from.
----	-------------	--------------------------

Reimplemented from **gazebo::physics::Entity** (p. 276).

10.70.3.72 `virtual void gazebo::physics::Link::UpdateSurface () [inline],[virtual]`

Update surface parameters.

10.70.4 Member Data Documentation

10.70.4.1 `math::Vector3 gazebo::physics::Link::angularAccel [protected]`

Angular acceleration.

10.70.4.2 `std::vector<math::Pose> gazebo::physics::Link::attachedModelsOffset [protected]`

Offsets for the attached models.

10.70.4.3 `std::vector<std::string> gazebo::physics::Link::cgVisuals [protected]`

Center of gravity visual elements.

10.70.4.4 `InertialPtr gazebo::physics::Link::inertial [protected]`

Inertial (p. 357) properties.

Referenced by `GetInertial()`.

10.70.4.5 `math::Vector3 gazebo::physics::Link::linearAccel [protected]`

Linear acceleration.

10.70.4.6 `std::vector<std::string> gazebo::physics::Link::visuals [protected]`

Link (p. 398) visual elements.

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

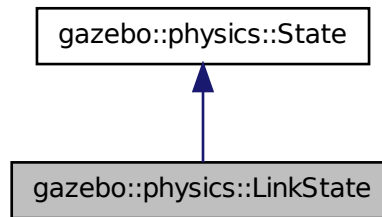
- **Link.hh**

10.71 gazebo::physics::LinkState Class Reference

Store state information of a **physics::Link** (p. 398) 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 \sim **LinkState** ()
Destructor.
- 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. 398).*
- 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

10.71.1 Detailed Description

Store state information of a **physics::Link** (p. 398) object.

This class captures the entire state of a **Link** (p. 398) at one specific time during a simulation run.

State (p. 702) of a **Link** (p. 398) includes the state of itself all its child **Collision** (p. 180) entities.

10.71.2 Constructor & Destructor Documentation

10.71.2.1 gazebo::physics::LinkState::LinkState ()

Default constructor.

10.71.2.2 gazebo::physics::LinkState::LinkState (const LinkPtr _link) [explicit]

Constructor.

Build a **LinkState** (p. 416) from an existing **Link** (p. 398).

Parameters

in	<code>_model</code>	Pointer to the Link (p. 398) from which to gather state info.
----	---------------------	--

10.71.2.3 gazebo::physics::LinkState::LinkState (const sdf::ElementPtr _sdf) [explicit]

Constructor.

Build a **LinkState** (p. 416) from SDF data

Parameters

in	<code>_sdf</code>	SDF data to load a link state from.
----	-------------------	-------------------------------------

10.71.2.4 virtual gazebo::physics::LinkState::~~LinkState () [virtual]

Destructor.

10.71.3 Member Function Documentation

10.71.3.1 const math::Pose& gazebo::physics::LinkState::GetAcceleration () const

Get the link acceleration.

Returns

The acceleration represented as a **math::Pose** (p. 556).

10.71.3.2 CollisionState gazebo::physics::LinkState::GetCollisionState (unsigned int _index) const

Get a collision state.

Get a **Collision** (p. 180) **State** (p. 702) based on an index, where index is in the range of 0...**LinkState::GetCollisionStateCount** (p. 420).

Parameters

in	<i>_index</i>	Index of the CollisionState (p. 190).
----	---------------	--

Returns

State (p. 702) of the **Collision** (p. 180).

Exceptions

common::Exception (p. 303)	When <i>_index</i> is invalid.
--------------------------------------	--------------------------------

10.71.3.3 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. 190) with the matching name, if any.

Parameters

in	<i>_collisionName</i>	Name of the CollisionState (p. 190)
----	-----------------------	--

Returns

State (p. 702) of the **Collision** (p. 180).

Exceptions

common::Exception (p. 303)	When <i>_collisionName</i> is invalid
--------------------------------------	---------------------------------------

10.71.3.4 `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. 190) recorded.

10.71.3.5 `const std::vector<CollisionState>& gazebo::physics::LinkState::GetCollisionStates () const`

Get the collision states.

Returns

A vector of collision states.

10.71.3.6 `const math::Pose& gazebo::physics::LinkState::GetPose () const`

Get the link pose.

Returns

The **math::Pose** (p. 556) of the **Link** (p. 398).

10.71.3.7 `const math::Pose& gazebo::physics::LinkState::GetVelocity () const`

Get the link velocity.

Returns

The velocity represented as a **math::Pose** (p. 556).

10.71.3.8 `const math::Pose& gazebo::physics::LinkState::GetWrench () const`

Get the force applied to the **Link** (p. 398).

Returns

Magnitude of the force.

10.71.3.9 `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.

10.71.3.10 virtual void gazebo::physics::LinkState::Load (const sdf::ElementPtr *_elem*) [virtual]

Load state from SDF element.

Load **LinkState** (p. 416) information from stored data in and SDF::Element.

Parameters

in	<i>_elem</i>	Pointer to the SDF::Element containing state info.
----	--------------	--

Reimplemented from **gazebo::physics::State** (p. 704).

10.71.3.11 **LinkState** gazebo::physics::LinkState::operator+ (const LinkState & *_state*) const

Addition operator.

Parameters

in	<i>_pt</i>	A state to add.
----	------------	-----------------

Returns

The resulting state.

10.71.3.12 **LinkState** gazebo::physics::LinkState::operator- (const LinkState & *_state*) const

Subtraction operator.

Parameters

in	<i>_pt</i>	A state to subtract.
----	------------	----------------------

Returns

The resulting state.

10.71.3.13 **LinkState&** gazebo::physics::LinkState::operator= (const LinkState & *_state*)

Assignment operator.

Parameters

in	<i>_state</i>	State (p. 702) value
----	---------------	-----------------------------

Returns

this

10.71.4 Friends And Related Function Documentation

10.71.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::physics::LinkState & _state)` [friend]

Stream insertion operator.

Parameters

in	<code>_out</code>	output stream
in	<code>_state</code>	Link (p. 398) state to output

Returns

the stream

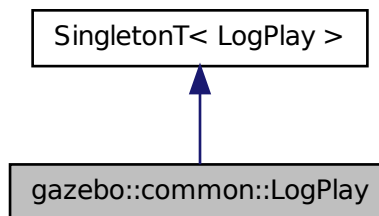
The documentation for this class was generated from the following file:

- [LinkState.hh](#)

10.72 gazebo::common::LogPlay Class Reference

```
#include <LogPlay.hh>
```

Inheritance diagram for gazebo::common::LogPlay:



Public Member Functions

- 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

10.72.1 Member Function Documentation

10.72.1.1 bool gazebo::common::LogPlay::IsOpen () const

Return true if a file is open.

Returns

True if a log file is open.

10.72.1.2 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	_logFile	The file to load
----	----------	------------------

Exceptions

Exception (p. 303)

10.72.1.3 bool gazebo::common::LogPlay::Step (std::string & _data)

Step through the open log file.

Parameters

out	_data	Data from next entry in the log file.
-----	-------	---------------------------------------

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

- **LogPlay.hh**

10.73 Logplay Class Reference

Open and playback log files that were recorded using LogRecord.

10.73.1 Detailed Description

Open and playback log files that were recorded using LogRecord.

Use **Logplay** (p. 423) 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:

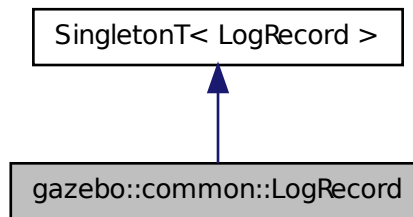
- **LogPlay.hh**

10.74 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 **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

10.74.1 Detailed Description

addtogroup gazebo_common

Handles logging of data to disk

The **LogRecord** (p. 424) 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. 422) for playback functionality.

State information for an entity may be logged through the **LogRecord::Add** (p. 425) function, and stopped through the **LogRecord::Remove** (p. 426) function. Data may be logged into a single file, or split into many separate files by specifying different filenames for the **LogRecord::Add** (p. 425) function.

The **LogRecord** (p. 424) is updated at the start of each simulation step. This guarantees that all data is stored.

See Also

Logplay (p. 423), State

10.74.2 Member Function Documentation

10.74.2.1 `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

Exception (p. 303)

10.74.2.2 `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.

10.74.2.3 `bool gazebo::common::LogRecord::Init (const std::string & _subdir)`

Initialize logging into a subdirectory.

Init may only be called once, False will be returned if called multiple times.

Parameters

<code>in</code>	<code>_subdir</code>	Directory to record to
-----------------	----------------------	------------------------

Returns

True if successful.

10.74.2.4 `bool gazebo::common::LogRecord::Remove (const std::string & _name)`

Remove an entity from a log.

Removes an entity from the logger. The stops data recording for the entity and all its children. For example, specifying a world will stop all data logging.

Parameters

<code>in</code>	<code>_name</code>	Name of the log
-----------------	--------------------	-----------------

Returns

True if the entity existed and was removed. False if the entity was not registered with the logger.

10.74.2.5 `void gazebo::common::LogRecord::Start (const std::string & _encoding = "bz2")`

Start the logger.

Parameters

<code>in</code>	<code>_encoding</code>	The type of encoding (txt, or bz2).
-----------------	------------------------	-------------------------------------

10.74.2.6 `void gazebo::common::LogRecord::Stop ()`

Stop the logger.

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

- **LogRecord.hh**

10.75 gazebo::Master Class Reference

A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

```
#include <gazebo_core.hh>
```

Public Member Functions

- **Master** ()
Constructor.
- virtual **~Master** ()
Destructor.
- void **Fini** ()
Finalize the master.
- void **Init** (uint16_t _port)
Initialize.
- void **Run** ()
Run the master.
- void **RunOnce** ()
Run the master one iteration.
- void **RunThread** ()
Run the master in a new thread.
- void **Stop** ()
Stop the master.

10.75.1 Detailed Description

A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

Base class for simulation server that handles commandline options, starts a **Master** (p.426), runs World update and sensor generation loops.

10.75.2 Constructor & Destructor Documentation

10.75.2.1 gazebo::Master::Master ()

Constructor.

10.75.2.2 virtual gazebo::Master::~~Master () [virtual]

Destructor.

10.75.3 Member Function Documentation

10.75.3.1 void gazebo::Master::Fini ()

Finalize the master.

10.75.3.2 void gazebo::Master::Init (uint16_t _port)

Initialize.

Parameters

in	_port	The master's port
----	-------	-------------------

10.75.3.3 void gazebo::Master::Run ()

Run the master.

10.75.3.4 void gazebo::Master::RunOnce ()

Run the master one iteration.

10.75.3.5 void gazebo::Master::RunThread ()

Run the master in a new thread.

10.75.3.6 void gazebo::Master::Stop ()

Stop the master.

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

- **Master.hh**

10.76 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.

10.76.1 Detailed Description

Encapsulates description of a material.

10.76.2 Member Enumeration Documentation

10.76.2.1 enum gazebo::common::Material::BlendMode

Enumerator:

ADD
MODULATE
REPLACE
BLEND_COUNT

10.76.2.2 enum gazebo::common::Material::ShadeMode

Enumerator:

FLAT
GOURAUD
PHONG
BLINN
SHADE_COUNT

10.76.3 Constructor & Destructor Documentation

10.76.3.1 gazebo::common::Material::Material ()

Constructor.

10.76.3.2 virtual gazebo::common::Material::~~Material () [virtual]

Destructor.

10.76.3.3 gazebo::common::Material::Material (const Color & _clr)

Create a material with a default color.

Parameters

in	_clr	Color (p. 193) of the material
----	------	---------------------------------------

10.76.4 Member Function Documentation

10.76.4.1 Color gazebo::common::Material::GetAmbient () const

Get the ambient color.

Returns

The ambient color

10.76.4.2 `void gazebo::common::Material::GetBlendFactors (double & _srcFactor, double & _dstFactor)`

Get the blend factors.

Parameters

<code>in</code>	<code><i>_srcFactor</i></code>	Source factor is returned in this variable
<code>in</code>	<code><i>_dstFactor</i></code>	Destination factor is returned in this variable

10.76.4.3 **BlendMode** `gazebo::common::Material::GetBlendMode () const`

Get the blending mode.

Returns

the blend mode

10.76.4.4 `bool gazebo::common::Material::GetDepthWrite () const`

Get depth write.

Returns

the depth write enabled state

10.76.4.5 **Color** `gazebo::common::Material::GetDiffuse () const`

Get the diffuse color.

Returns

The diffuse color

10.76.4.6 **Color** `gazebo::common::Material::GetEmissive () const`

Get the emissive color.

Returns

The emissive color

10.76.4.7 `bool gazebo::common::Material::GetLighting () const`

Get lighting enabled.

Returns

the lighting enabled state

10.76.4.8 `std::string gazebo::common::Material::GetName () const`

Get the name of the material.

Returns

The name of the material

10.76.4.9 `double gazebo::common::Material::GetPointSize () const`

Get the point size.

Returns

the point size

10.76.4.10 `ShadeMode gazebo::common::Material::GetShadeMode () const`

Get the shading mode.

Returns

the shading mode

10.76.4.11 `double gazebo::common::Material::GetShininess () const`

Get the shininess.

Returns

The shininess value

10.76.4.12 `Color gazebo::common::Material::GetSpecular () const`

Get the specular color.

Returns

The specular color

10.76.4.13 `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

10.76.4.14 `double gazebo::common::Material::GetTransparency () const`

Get the transparency percentage (0..1)

Returns

The transparency percentage

10.76.4.15 `void gazebo::common::Material::SetAmbient (const Color & _clr)`

Set the ambient color.

Parameters

in	_clr	The ambient color
----	------	-------------------

10.76.4.16 `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

10.76.4.17 `void gazebo::common::Material::SetBlendMode (BlendMode _b)`

Set the blending mode.

Parameters

in	_b	the blend mode
----	----	----------------

10.76.4.18 `void gazebo::common::Material::SetDepthWrite (bool _value)`

Set depth write.

Parameters

in	_value	the depth write enabled state
----	--------	-------------------------------

10.76.4.19 `void gazebo::common::Material::SetDiffuse (const Color & _clr)`

Set the diffuse color.

Parameters

in	<code>_clr</code>	The diffuse color
----	-------------------	-------------------

10.76.4.20 void gazebo::common::Material::SetEmissive (const Color & `_clr`)

Set the emissive color.

Parameters

in	<code>_clr</code>	The emissive color
----	-------------------	--------------------

10.76.4.21 void gazebo::common::Material::SetLighting (bool `_value`)

Set lighting enabled.

Parameters

in	<code>_value</code>	the lighting enabled state
----	---------------------	----------------------------

10.76.4.22 void gazebo::common::Material::SetPointSize (double `_size`)

Set the point size.

Parameters

in	<code>_size</code>	the size
----	--------------------	----------

10.76.4.23 void gazebo::common::Material::SetShadeMode (ShadeMode `_b`)

Set the shading mode param[in] the shading mode.

10.76.4.24 void gazebo::common::Material::SetShininess (double `_t`)

Set the shininess.

Parameters

in	<code>_t</code>	The shininess value
----	-----------------	---------------------

10.76.4.25 void gazebo::common::Material::SetSpecular (const Color & `_clr`)

Set the specular color.

Parameters

in	<code>_clr</code>	The specular color
----	-------------------	--------------------

10.76.4.26 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
----	-------------	--

10.76.4.27 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>

10.76.4.28 void gazebo::common::Material::SetTransparency (double *_t*)

Set the transparency percentage (0..1)

Parameters

in	<i>_t</i>	The amount of transparency (0..1)
----	-----------	-----------------------------------

10.76.5 Friends And Related Function Documentation

10.76.5.1 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.

10.76.6 Member Data Documentation

10.76.6.1 Color gazebo::common::Material::ambient [protected]

the ambient light color

10.76.6.2 BlendMode gazebo::common::Material::blendMode [protected]

blend mode

10.76.6.3 std::string gazebo::common::Material::BlendModeStr[BLEND_COUNT] [static]

10.76.6.4 Color gazebo::common::Material::diffuse [protected]

the diffuse lighth color

10.76.6.5 **Color** gazebo::common::Material::emissive [protected]

the emissive light color

10.76.6.6 **std::string** gazebo::common::Material::name [protected]

the name of the material

10.76.6.7 **double** gazebo::common::Material::pointSize [protected]

point size

10.76.6.8 **ShadeMode** gazebo::common::Material::shadeMode [protected]

the shade mode

10.76.6.9 **std::string** gazebo::common::Material::ShadeModeStr[SHADE_COUNT] [static]

10.76.6.10 **double** gazebo::common::Material::shininess [protected]

shininess value (0 to 1)

10.76.6.11 **Color** gazebo::common::Material::specular [protected]

the specular light color

10.76.6.12 **std::string** gazebo::common::Material::texImage [protected]

the texture image file name

10.76.6.13 **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**

10.77 gazebo::math::Matrix3 Class Reference

A 3x3 matrix class.

```
#include <Matrix3.hh>
```

Public Member Functions

- **Matrix3** ()
Constructor.
- **Matrix3** (const **Matrix3** &_m)
Copy constructor.
- **Matrix3** (double _v00, double _v01, double _v02, double _v10, double _v11, double _v12, double _v20, double _v21, double _v22)
Constructor.
- virtual ~**Matrix3** ()
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.

10.77.1 Detailed Description

A 3x3 matrix class.

10.77.2 Constructor & Destructor Documentation

10.77.2.1 gazebo::math::Matrix3::Matrix3 ()

Constructor.

10.77.2.2 gazebo::math::Matrix3::Matrix3 (const Matrix3 & _m)

Copy constructor.

Parameters

_m	Matrix to copy
----	----------------

10.77.2.3 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

10.77.2.4 virtual gazebo::math::Matrix3::~~Matrix3 () [virtual]

Destructor.

10.77.3 Member Function Documentation

10.77.3.1 bool gazebo::math::Matrix3::operator==(const Matrix3 & _m) const

Equality test operator.

Parameters

in	_m	Matrix3 (p. 437) to test
----	----	---------------------------------

Returns

True if equal (using the default tolerance of 1e-6)

10.77.3.2 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.

10.77.3.3 `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.

10.77.3.4 `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

10.77.3.5 `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

10.77.3.6 `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

10.77.4 Friends And Related Function Documentation

10.77.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Matrix3 & _m)` [friend]

Stream insertion operator.

Parameters

<code>in</code>	<code>_out</code>	Output stream
<code>in</code>	<code>_m</code>	Matrix to output

Returns

the stream

10.77.5 Member Data Documentation

10.77.5.1 `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**

10.78 gazebo::math::Matrix4 Class Reference

A 3x3 matrix class.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Matrix4 ()**
Constructor.
- **Matrix4 (const Matrix4 &_m)**
Copy constructor.
- **Matrix4 (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)**
Constructor.
- virtual **~Matrix4 ()**
Destructor.
- **math::Pose GetAsPose () const**
*Get the transformation as **math::Pose** (p. 556).*
- **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. 799).

- **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.

10.78.1 Detailed Description

A 3x3 matrix class.

10.78.2 Constructor & Destructor Documentation

10.78.2.1 gazebo::math::Matrix4::Matrix4 ()

Constructor.

10.78.2.2 gazebo::math::Matrix4::Matrix4 (const Matrix4 & _m)

Copy constructor.

Parameters

<code>_m</code>	Matrix to copy
-----------------	----------------

10.78.2.3 gazebo::math::Matrix4::Matrix4 (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

Constructor.

Parameters

<code>in</code>	<code>_v00</code>	Row 0, Col 0 value
<code>in</code>	<code>_v01</code>	Row 0, Col 1 value
<code>in</code>	<code>_v02</code>	Row 0, Col 2 value
<code>in</code>	<code>_v03</code>	Row 0, Col 3 value
<code>in</code>	<code>_v10</code>	Row 1, Col 0 value
<code>in</code>	<code>_v11</code>	Row 1, Col 1 value
<code>in</code>	<code>_v12</code>	Row 1, Col 2 value
<code>in</code>	<code>_v13</code>	Row 1, Col 3 value
<code>in</code>	<code>_v20</code>	Row 2, Col 0 value
<code>in</code>	<code>_v21</code>	Row 2, Col 1 value
<code>in</code>	<code>_v22</code>	Row 2, Col 2 value
<code>in</code>	<code>_v23</code>	Row 2, Col 3 value
<code>in</code>	<code>_v30</code>	Row 3, Col 0 value
<code>in</code>	<code>_v31</code>	Row 3, Col 1 value
<code>in</code>	<code>_v32</code>	Row 3, Col 2 value
<code>in</code>	<code>_v33</code>	Row 3, Col 3 value

10.78.2.4 virtual gazebo::math::Matrix4::~~Matrix4 () [virtual]

Destructor.

10.78.3 Member Function Documentation

10.78.3.1 **math::Pose** gazebo::math::Matrix4::GetAsPose () const

Get the transformation as **math::Pose** (p. 556).

Returns

the pose

10.78.3.2 **Vector3** gazebo::math::Matrix4::GetEulerRotation (unsigned int *solution_number* = 1) const

Get the rotation as a Euler angles.

Returns

the rotation

10.78.3.3 **Quaternion** gazebo::math::Matrix4::GetRotation () const

Get the rotation as a quaternion.

Returns

the rotation

10.78.3.4 **Vector3** gazebo::math::Matrix4::GetTranslation () const

Get the translational values as a **Vector3** (p. 799).

Returns

x,y,z

10.78.3.5 **Matrix4** gazebo::math::Matrix4::Inverse () const

Return the inverse matrix.

10.78.3.6 **bool** gazebo::math::Matrix4::IsAffine () const

Return true if the matrix is affine.

Returns

true if the matrix is affine, false otherwise

10.78.3.7 **Matrix4** gazebo::math::Matrix4::operator* (const **Matrix4** & *_mat*) const

Multiplication operator.

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

Returns

This matrix * `_mat`

10.78.3.8 Matrix4 gazebo::math::Matrix4::operator* (const Matrix3 & `_mat`) const

Multiplication operator.

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

Returns

This matrix * `_mat`

10.78.3.9 Vector3 gazebo::math::Matrix4::operator* (const Vector3 & `_vec`) const

Multiplication operator.

Parameters

<code>_vec</code>	Vector3 (p. 799)
-------------------	-------------------------

Returns

Resulting vector from multiplication

10.78.3.10 Matrix4& gazebo::math::Matrix4::operator= (const Matrix4 & `_mat`)

Equal operator.

this = `_mat`

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

Returns

itself

10.78.3.11 const Matrix4& gazebo::math::Matrix4::operator= (const Matrix3 & `_mat`)

Equal operator for 3x3 matrix.

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

Returns

itself

10.78.3.12 `bool gazebo::math::Matrix4::operator==(const Matrix4 & _m) const`

Equality operator.

Parameters

<code>in</code>	<code>_m</code>	Matrix3 (p. 437) to test
-----------------	-----------------	---------------------------------

Returns

true if the 2 matrices are equal (using the tolerance 1e-6), false otherwise

10.78.3.13 `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.

10.78.3.14 `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.

10.78.3.15 `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

10.78.3.16 void gazebo::math::Matrix4::SetScale (const Vector3 & *s*)

Set the scale.

Parameters

in	<code>_s</code>	scale
----	-----------------	-------

10.78.3.17 void gazebo::math::Matrix4::SetTranslate (const Vector3 & *t*)

Set the translational values [(0, 3) (1, 3) (2, 3)].

Parameters

in	<code>_t</code>	Values to set
----	-----------------	---------------

10.78.3.18 Vector3 gazebo::math::Matrix4::TransformAffine (const Vector3 & *v*) const

Perform an affine transformation.

Parameters

<code>_v</code>	Vector3 (p. 799) value for the transformation
-----------------	--

Returns

The result of the transformation

10.78.4 Friends And Related Function Documentation

10.78.4.1 `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

10.78.5 Member Data Documentation

10.78.5.1 `const Matrix4 gazebo::math::Matrix4::IDENTITY` [static]

Identity matrix.

10.78.5.2 `double gazebo::math::Matrix4::m[4][4]` [protected]

The 4x4 matrix.

Referenced by `operator[]()`.

10.78.5.3 `const Matrix4 gazebo::math::Matrix4::ZERO` [static]

Zero matrix.

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

- **Matrix4.hh**

10.79 gazebo::common::Mesh Class Reference

A 3D mesh.

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

Public Member Functions

- **Mesh** ()
Constructor.
- virtual **~Mesh** ()
Destructor.
- int **AddMaterial** (**Material** *_mat)
Add a material to the mesh.
- void **AddSubMesh** (**SubMesh** *_child)
Add a submesh mesh.

- void **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 **SetSkeleton** (**Skeleton** * _skel)
Set the mesh skeleton.

10.79.1 Detailed Description

A 3D mesh.

10.79.2 Constructor & Destructor Documentation

10.79.2.1 gazebo::common::Mesh::Mesh ()

Constructor.

10.79.2.2 virtual gazebo::common::Mesh::~~Mesh () [virtual]

Destructor.

10.79.3 Member Function Documentation

10.79.3.1 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

10.79.3.2 void gazebo::common::Mesh::AddSubMesh (SubMesh * *_child*)

Add a submesh mesh.

The **Mesh** (p. 448) object takes ownership of the submesh.

Parameters

in	<i>_child</i>	the submesh
----	---------------	-------------

10.79.3.3 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

10.79.3.4 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
----	----------------	------------------------------

10.79.3.5 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

10.79.3.6 unsigned int gazebo::common::Mesh::GetIndexCount () const

Return the number of indices.

Returns

the count

10.79.3.7 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

10.79.3.8 unsigned int gazebo::common::Mesh::GetMaterialCount () const

Get the number of materials.

Returns

the count

10.79.3.9 math::Vector3 gazebo::common::Mesh::GetMax () const

Get the maximum X, Y, Z values.

Returns

the upper bounds of the bounding box

10.79.3.10 `math::Vector3 gazebo::common::Mesh::GetMin () const`

Get the minimum X, Y, Z values.

Returns

the lower bounds of the bounding box

10.79.3.11 `std::string gazebo::common::Mesh::GetName () const`

Get the name of this mesh.

Returns

the name

10.79.3.12 `unsigned int gazebo::common::Mesh::GetNormalCount () const`

Return the number of normals.

Returns

the count

10.79.3.13 `std::string gazebo::common::Mesh::GetPath () const`

Get the path which contains the mesh resource.

Returns

the path to the mesh resource

10.79.3.14 `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.

10.79.3.15 `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

10.79.3.16 `unsigned int gazebo::common::Mesh::GetSubMeshCount () const`

Get the number of children.

Returns

the count

10.79.3.17 `unsigned int gazebo::common::Mesh::GetTexCoordCount () const`

Return the number of texture coordinates.

Returns

the count

10.79.3.18 `unsigned int gazebo::common::Mesh::GetVertexCount () const`

Return the number of vertices.

Returns

the count

10.79.3.19 `bool gazebo::common::Mesh::HasSkeleton () const`

Return true if mesh is attached to a skeleton.

10.79.3.20 `void gazebo::common::Mesh::RecalculateNormals ()`

Recalculate all the normals of each face defined by three indices.

10.79.3.21 `void gazebo::common::Mesh::Scale (double _factor)`

Scale all vertices by `_factor`.

Parameters

<code><i>_factor</i></code>	Scaling factor
-----------------------------	----------------

10.79.3.22 `void gazebo::common::Mesh::SetName (const std::string & _n)`

Set the name of this mesh.

Parameters

in	<code>_n</code>	the name to set
----	-----------------	-----------------

10.79.3.23 void gazebo::common::Mesh::SetPath (const std::string & *_path*)

Set the path which contains the mesh resource.

Parameters

in	<code>_path</code>	the file path
----	--------------------	---------------

10.79.3.24 void gazebo::common::Mesh::SetSkeleton (Skeleton * *_skel*)

Set the mesh skeleton.

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

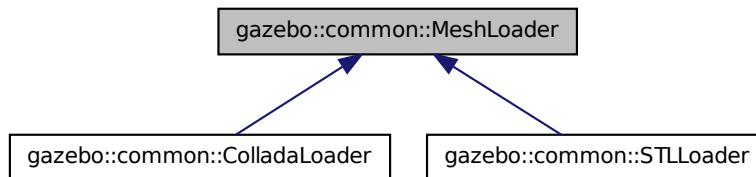
- **Mesh.hh**

10.80 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.

10.80.1 Detailed Description

Base class for loading meshes.

10.80.2 Constructor & Destructor Documentation

10.80.2.1 gazebo::common::MeshLoader::MeshLoader ()

Constructor.

10.80.2.2 virtual gazebo::common::MeshLoader::~~MeshLoader () [virtual]

Destructor.

10.80.3 Member Function Documentation

10.80.3.1 virtual Mesh* gazebo::common::MeshLoader::Load (const std::string & *filename*) [pure virtual]

Load a 3D mesh.

Parameters

in	<i>_filename</i>	the path to the mesh
----	------------------	----------------------

Returns

a pointer to the created mesh

Implemented in **gazebo::common::ColladaLoader** (p. 180), and **gazebo::common::STLLoader** (p. 706).

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

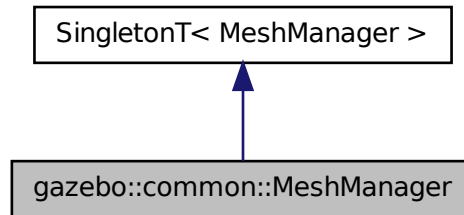
- **MeshLoader.hh**

10.81 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

10.81.1 Detailed Description

Maintains and manages all meshes.

10.81.2 Member Function Documentation

10.81.2.1 void gazebo::common::MeshManager::AddMesh (Mesh * _mesh)

Add a mesh to the manager.

This **MeshManager** (p. 455) takes ownership of the mesh and will destroy it. See ~MeshManager.

Parameters

in	<i>the</i>	mesh to add.
----	------------	--------------

10.81.2.2 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

10.81.2.3 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

10.81.2.4 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

10.81.2.5 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

10.81.2.6 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

10.81.2.7 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

10.81.2.8 void gazebo::common::MeshManager::CreateSphere (const std::string & *_name*, float *_radius*, int *_rings*, int *_segments*)

Create a sphere mesh.

Parameters

in	<code>_name</code>	the name of the mesh
in	<code>_radius</code>	radius of the sphere in meter
in	<code>_rings</code>	number of circles on th y axis
in	<code>_segments</code>	number of segment per circle

10.81.2.9 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	<code>_name</code>	the name of the new mesh
in	<code>_innerRadius</code>	the inner radius of the tube in the x y plane
in	<code>_outterRadius</code>	the outer radius of the tube in the x y plane
in	<code>_height</code>	the height along z
in	<code>_rings</code>	the number of circles along the height
in	<code>_segments</code>	the number of segment per circle

10.81.2.10 void gazebo::common::MeshManager::GenSphericalTexCoord (const Mesh * `_mesh`, math::Vector3 `_center`)

generate spherical texture coordinates

10.81.2.11 const Mesh* gazebo::common::MeshManager::GetMesh (const std::string & `_name`) const

Get a mesh by name.

Parameters

in	<code>_name</code>	the name of the mesh to look for
----	--------------------	----------------------------------

Returns

the mesh or NULL if not found

10.81.2.12 void gazebo::common::MeshManager::GetMeshAABB (const Mesh * `_mesh`, math::Vector3 & `_center`, math::Vector3 & `_min_xyz`, math::Vector3 & `_max_xyz`)

Get mesh aabb and center.

Parameters

in	<code>_mesh</code>	the mesh
out	<code>_center</code>	the AAB center position
out	<code>_min_xyz</code>	the bounding box minimum
out	<code>_max_xyz</code>	the bounding box maximum

10.81.2.13 `bool gazebo::common::MeshManager::HasMesh (const std::string & _name) const`

Return true if the mesh exists.

Parameters

<code>in</code>	<code><i>_name</i></code>	the name of the mesh
-----------------	---------------------------	----------------------

10.81.2.14 `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

10.81.2.15 `const Mesh* gazebo::common::MeshManager::Load (const std::string & _filename)`

Load a mesh from a file.

Parameters

<code>in</code>	<code><i>_filename</i></code>	the path to the mesh
-----------------	-------------------------------	----------------------

Returns

a pointer to the created mesh

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

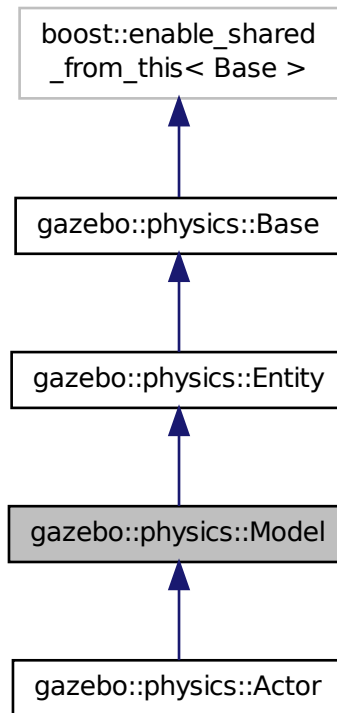
- **MeshManager.hh**

10.82 gazebo::physics::Model Class Reference

A model is a collection of links, joints, and plugins.

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


Inheritance diagram for gazebo::physics::Model:



Public Member Functions

- **Model** (BasePtr _parent)
Constructor.
- virtual **~Model** ()
Destructor.
- void **AttachStaticModel** (ModelPtr &_model, math::Pose _offset)
Attach a static model to this model.
- void **DetachStaticModel** (const std::string &_model)
Detach a static model from this model.
- void **FillModelMsg** (msgs::Model &_msg) **GAZEBO_DEPRECATED**
DEPRECATED.
- void **FillMsg** (msgs::Model &_msg)
Fill a model message.
- virtual void **Fini** ()
Finalize the model.
- **Link_V GetAllLinks** () const **GAZEBO_DEPRECATED**
Deprecated.

- 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** (unsigned int index) const **GAZEBO_DEPRECATED**
Get a joint by index.
- **JointPtr GetJoint** (const std::string &name)
Get a joint.
- 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 GetLink** (unsigned int _index) const **GAZEBO_DEPRECATED**
This function is dangerous. Do not use.
- **LinkPtr GetLinkById** (unsigned int _id) const
Get a link by id.
- **Link_V GetLinks** () const
*Construct and return a vector of **Link** (p. 398)'s in this model Note this constructs the vector of **Link** (p. 398)'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 **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. 398), 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. 366) Animation.*
- void **SetJointPosition** (const std::string &_jointName, double _position)
*Set the positions of a **Joint** (p. 366) 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. 460) by specifying desired Pose of a **Link** (p. 398) within the **Model** (p. 460).*
- void **SetLinkWorldPose** (const math::Pose &_pose, const LinkPtr &_link)
*Set the Pose of the entire **Model** (p. 460) by specifying desired Pose of a **Link** (p. 398) within the **Model** (p. 460).*
- 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. 464)*
- std::vector< **math::Pose** > **attachedModelsOffset**
*used by **Model::AttachStaticModel** (p. 464)*

Additional Inherited Members

10.82.1 Detailed Description

A model is a collection of links, joints, and plugins.

10.82.2 Constructor & Destructor Documentation

10.82.2.1 gazebo::physics::Model::Model (**BasePtr** *_parent*) [explicit]

Constructor.

Parameters

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

10.82.2.2 virtual gazebo::physics::Model::~~Model () [virtual]

Destructor.

10.82.3 Member Function Documentation

10.82.3.1 void gazebo::physics::Model::AttachStaticModel (**ModelPtr** & *_model*, **math::Pose** *_offset*)

Attach a static model to this model.

This function takes as input a static **Model** (p. 460), which is a **Model** (p. 460) that has been marked as static (no physics simulation), and attaches it to this **Model** (p. 460) 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

<i>in</i>	<i>_model</i>	Pointer to the static model.
<i>in</i>	<i>_offset</i>	Offset, relative to this Model (p. 460), to place <i>_model</i> .

10.82.3.2 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. 464).

10.82.3.3 void gazebo::physics::Model::FillModelMsg (msgs::Model & *_msg*)

DEPRECATED.

10.82.3.4 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.
----	-------------	--

10.82.3.5 virtual void gazebo::physics::Model::Fini () [virtual]

Finalize the model.

Reimplemented from **gazebo::physics::Entity** (p. 269).

Reimplemented in **gazebo::physics::Actor** (p. 104).

10.82.3.6 Link_V gazebo::physics::Model::GetAllLinks () const

Deprecated.

10.82.3.7 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.

10.82.3.8 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.269).

10.82.3.9 JointPtr gazebo::physics::Model::GetJoint (unsigned int *index*) const

Get a joint by index.

Parameters

<i>index</i>	Index of the joint
--------------	--------------------

Returns

A pointer to the joint

10.82.3.10 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

10.82.3.11 unsigned int gazebo::physics::Model::GetJointCount () const

Get the number of joints.

Returns

Get the number of joints.

10.82.3.12 const Joint_V& gazebo::physics::Model::GetJoints () const

Get the joints.

Returns

Vector of joints.

10.82.3.13 LinkPtr gazebo::physics::Model::GetLink (const std::string & *_name* = "canonical") const

Get a link by name.

Parameters

in	_name	Name of the link to get.
----	-------	--------------------------

Returns

Pointer to the link, NULL if the name is invalid.

10.82.3.14 **LinkPtr** gazebo::physics::Model::GetLink (unsigned int *_index*) const

This function is dangerous. Do not use.

10.82.3.15 **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.

10.82.3.16 **Link_V** gazebo::physics::Model::GetLinks () const

Construct and return a vector of **Link** (p. 398)'s in this model Note this constructs the vector of **Link** (p. 398)'s on the fly, could be costly.

Returns

a vector of **Link** (p. 398)'s in this model

10.82.3.17 unsigned int gazebo::physics::Model::GetPluginCount () const

Get the number of plugins this model has.

Returns

Number of plugins associated with this model.

10.82.3.18 virtual **math::Vector3** gazebo::physics::Model::GetRelativeAngularAccel () const [virtual]

Get the angular acceleration of the entity.

Returns

math::Vector3 (p. 799), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 270).

10.82.3.19 virtual **math::Vector3** gazebo::physics::Model::GetRelativeAngularVel () const [virtual]

Get the angular velocity of the entity.

Returns

math::Vector3 (p. 799), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 271).

10.82.3.20 virtual **math::Vector3** gazebo::physics::Model::GetRelativeLinearAccel () const [virtual]

Get the linear acceleration of the entity.

Returns

math::Vector3 (p. 799), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 271).

10.82.3.21 virtual **math::Vector3** gazebo::physics::Model::GetRelativeLinearVel () const [virtual]

Get the linear velocity of the entity.

Returns

math::Vector3 (p. 799), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 271).

10.82.3.22 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. 131).

Reimplemented in **gazebo::physics::Actor** (p. 104).

10.82.3.23 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.

10.82.3.24 virtual **math::Vector3** gazebo::physics::Model::GetWorldAngularAccel () const [virtual]

Get the angular acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 799), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 272).

10.82.3.25 virtual **math::Vector3** gazebo::physics::Model::GetWorldAngularVel () const [virtual]

Get the angular velocity of the entity in the world frame.

Returns

math::Vector3 (p. 799), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 272).

10.82.3.26 virtual **math::Vector3** gazebo::physics::Model::GetWorldLinearAccel () const [virtual]

Get the linear acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 799), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 272).

10.82.3.27 virtual **math::Vector3** gazebo::physics::Model::GetWorldLinearVel () const [virtual]

Get the linear velocity of the entity in the world frame.

Returns

math::Vector3 (p. 799), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 272).

10.82.3.28 virtual void gazebo::physics::Model::Init () [virtual]

Initialize the model.

Reimplemented from **gazebo::physics::Base** (p. 132).

Reimplemented in **gazebo::physics::Actor** (p. 104).

10.82.3.29 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. 273).

10.82.3.30 void gazebo::physics::Model::LoadPlugins ()

Load all plugins.

Load all plugins specified in the SDF for the model.

10.82.3.31 virtual void gazebo::physics::Model::OnPoseChange () [protected],[virtual]

Callback when the pose of the model has been changed.

Implements **gazebo::physics::Entity** (p. 273).

10.82.3.32 void gazebo::physics::Model::ProcessMsg (const msgs::Model & _msg)

Update parameters from a model message.

Parameters

in	<code>_msg</code>	Message to process.
----	-------------------	---------------------

10.82.3.33 virtual void gazebo::physics::Model::RemoveChild (EntityPtr _child) [virtual]

Remove a child.

Parameters

in	<code>_child</code>	Remove a child entity.
----	---------------------	------------------------

10.82.3.34 void gazebo::physics::Model::Reset () [virtual]

Reset the model.

Reimplemented from **gazebo::physics::Entity** (p. 273).

10.82.3.35 void gazebo::physics::Model::SetAngularAccel (const math::Vector3 & _vel)

Set the angular acceleration of the model, and all its links.

Parameters

in	<code>_vel</code>	The new angular acceleration
----	-------------------	------------------------------

10.82.3.36 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.
----	-------------	---------------------------

10.82.3.37 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.
----	-----------------	--

10.82.3.38 void gazebo::physics::Model::SetCollideMode (const std::string & *_mode*)

This is not implemented in **Link** (p. 398), which means this function doesn't do anything.

Set the collide mode of the model.

Parameters

in	<i>_mode</i>	The collision mode
----	--------------	--------------------

10.82.3.39 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.
----	-----------------	-------------------------------

10.82.3.40 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.
----	---------------	---

10.82.3.41 void gazebo::physics::Model::SetJointAnimation (const std::map< std::string, common::NumericAnimationPtr > *_anim*, boost::function< void()> *_onComplete* = NULL)

Joint (p. 366) 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.

10.82.3.42 `void gazebo::physics::Model::SetJointPosition (const std::string & jointName, double position)`

Set the positions of a **Joint** (p. 366) by name.

See Also

JointController::SetJointPosition (p. 381)

Parameters

in	<code>_jointName</code>	Name of the joint to set.
in	<code>_position</code>	Position to set the joint to.

10.82.3.43 `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. 381).

Parameters

in	<code>_jointPositions</code>	Map of joint names to their positions.
----	------------------------------	--

10.82.3.44 `void gazebo::physics::Model::SetLaserRetro (const float retro)`

Set the laser retro reflectiveness of the model.

Parameters

in	<code>_retro</code>	Retro reflectance value.
----	---------------------	--------------------------

10.82.3.45 `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.
----	-------------------	------------------------------

10.82.3.46 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.
----	-------------------	--------------------------

10.82.3.47 void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & _pose, std::string _linkName)

Set the Pose of the entire **Model** (p. 460) by specifying desired Pose of a **Link** (p. 398) within the **Model** (p. 460).

Doing so, keeps the configuration of the **Model** (p. 460) unchanged, i.e. all **Joint** (p. 366) angles are unchanged.

Parameters

in	<code>_pose</code>	Pose to set the link to.
in	<code>_linkName</code>	Name of the link to set.

10.82.3.48 void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & _pose, const LinkPtr & _link)

Set the Pose of the entire **Model** (p. 460) by specifying desired Pose of a **Link** (p. 398) within the **Model** (p. 460).

Doing so, keeps the configuration of the **Model** (p. 460) unchanged, i.e. all **Joint** (p. 366) angles are unchanged.

Parameters

in	<code>_pose</code>	Pose to set the link to.
in	<code>_link</code>	Pointer to the link to set.

10.82.3.49 void gazebo::physics::Model::SetState (const ModelState & _state)

Set the current model state.

Parameters

in	<code>_state</code>	State (p. 702) to set the model to.
----	---------------------	--

10.82.3.50 virtual void gazebo::physics::Model::StopAnimation () [virtual]

Stop the current animations.

Reimplemented from **gazebo::physics::Entity** (p. 275).

10.82.3.51 void gazebo::physics::Model::Update () [virtual]

Update the model.

Reimplemented from **gazebo::physics::Base** (p. 135).

10.82.3.52 `virtual void gazebo::physics::Model::UpdateParameters (sdf::ElementPtr _sdf) [virtual]`

Update the parameters using new sdf values.

Parameters

in	_sdf	SDF values to update from.
----	------	----------------------------

Reimplemented from `gazebo::physics::Entity` (p. 276).

Reimplemented in `gazebo::physics::Actor` (p. 105).

10.82.4 Member Data Documentation

10.82.4.1 `std::vector<ModelPtr> gazebo::physics::Model::attachedModels [protected]`

used by `Model::AttachStaticModel` (p. 464)

10.82.4.2 `std::vector<math::Pose> gazebo::physics::Model::attachedModelsOffset [protected]`

used by `Model::AttachStaticModel` (p. 464)

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

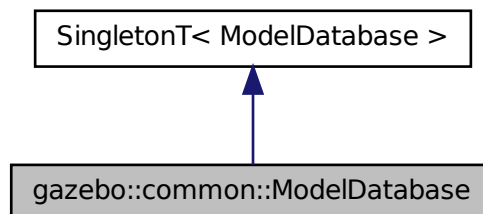
- `Model.hh`

10.83 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 **GetManifest** (const std::string &_uri)
Return the manifest.xml 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

10.83.1 Detailed Description

Connects to model database, and has utility functions to find models.

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

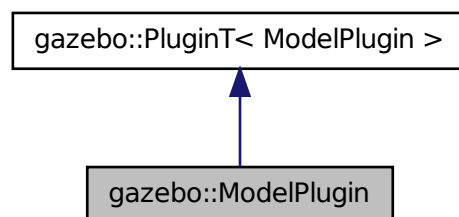
- **ModelDatabase.hh**

10.84 gazebo::ModelPlugin Class Reference

A plugin with access to **physics::Model** (p. 460).

```
#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

10.84.1 Detailed Description

A plugin with access to **physics::Model** (p. 460).

See [reference](#).

10.84.2 Constructor & Destructor Documentation

10.84.2.1 gazebo::ModelPlugin::ModelPlugin () [inline]

Constructor.

References gazebo::MODEL_PLUGIN, and gazebo::PluginT< ModelPlugin >::type.

10.84.2.2 virtual gazebo::ModelPlugin::~~ModelPlugin () [inline],[virtual]

Destructor.

10.84.3 Member Function Documentation

10.84.3.1 virtual void gazebo::ModelPlugin::Init () [inline],[virtual]

Override this method for custom plugin initialization behavior.

10.84.3.2 virtual void gazebo::ModelPlugin::Load (**physics::ModelPtr** _model, **sdf::ElementPtr** _sdf) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

in	<code>_model</code>	Pointer to the Model
in	<code>_sdf</code>	Pointer to the SDF element of the plugin.

10.84.3.3 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:

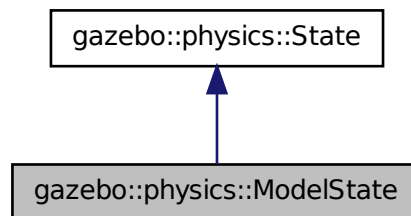
- **common/Plugin.hh**

10.85 gazebo::physics::ModelState Class Reference

Store state information of a **physics::Model** (p. 460) 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.
- **JointState GetJointState** (unsigned int _index) const
*Get a **Joint** (p. 366) state.*
- **JointState GetJointState** (const std::string &_jointName) const
*Get a **Joint** (p. 366) state by **Joint** (p. 366) 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. 398) 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 **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

10.85.1 Detailed Description

Store state information of a **physics::Model** (p. 460) object.

This class captures the entire state of a **Model** (p. 460) at one specific time during a simulation run.

State (p. 702) of a **Model** (p. 460) includes the state of all its child Links and Joints.

10.85.2 Constructor & Destructor Documentation

10.85.2.1 gazebo::physics::ModelState::ModelState ()

Default constructor.

10.85.2.2 gazebo::physics::ModelState::ModelState (const **ModelPtr** _model) [explicit]

Constructor.

Build a **ModelState** (p. 477) from an existing **Model** (p. 460).

Parameters

in	_model	Pointer to the model from which to gather state info.
----	--------	---

10.85.2.3 gazebo::physics::ModelState::ModelState (const sdf::ElementPtr _sdf) [explicit]

Constructor.

Build a **ModelState** (p. 477) from SDF data

Parameters

in	<code>_sdf</code>	SDF data to load a model state from.
----	-------------------	--------------------------------------

10.85.2.4 virtual gazebo::physics::ModelState::~~ModelState () [virtual]

Destructor.

10.85.3 Member Function Documentation

10.85.3.1 JointState gazebo::physics::ModelState::GetJointState (unsigned int _index) const

Get a **Joint** (p. 366) state.

Return a **JointState** (p. 381) based on a index, where index is between 0...**ModelState::GetJointStateCount()** (p. 480).

Parameters

in	<code>_index</code>	Index of a JointState (p. 381).
----	---------------------	--

Returns

State (p. 702) of a **Joint** (p. 366).

Exceptions

<i>common::Exception</i> (p. 303)	When <code>_index</code> is out of range.
---	---

10.85.3.2 JointState gazebo::physics::ModelState::GetJointState (const std::string & _jointName) const

Get a **Joint** (p. 366) state by **Joint** (p. 366) name.

Searches through all JointStates. Returns the **JointState** (p. 381) with the matching name, if any.

Parameters

in	<code>_jointName</code>	Name of the JointState (p. 381).
----	-------------------------	---

Returns

State (p. 702) of the **Joint** (p. 366).

Exceptions

<i>common::Exception</i> (p. 303)	When <code>_jointName</code> is invalid.
---	--

10.85.3.3 unsigned int gazebo::physics::ModelState::GetJointStateCount () const

Get the number of joint states.

Returns the number of JointStates recorded.

Returns

Number of JointStates.

10.85.3.4 const std::vector<JointState>& gazebo::physics::ModelState::GetJointStates () const

Get the joint states.

Returns

A vector of joint states.

10.85.3.5 LinkState gazebo::physics::ModelState::GetLinkState (unsigned int *_index*) const

Get a link state.

Get a **Link** (p. 398) **State** (p. 702) based on an index, where index is in the range of 0...**ModelState::GetLinkStateCount** (p. 481)

Parameters

<i>in</i>	<i>_index</i>	Index of the LinkState (p. 416)
-----------	---------------	--

Returns

State (p. 702) of the **Link** (p. 398).

Exceptions

<i>common::Exception</i> (p. 303)	When <i>_index</i> is out of range.
---	-------------------------------------

10.85.3.6 LinkState gazebo::physics::ModelState::GetLinkState (const std::string & *_linkName*) const

Get a link state by **Link** (p. 398) name.

Searches through all LinkStates. Returns the **LinkState** (p. 416) with the matching name, if any.

Parameters

<i>in</i>	<i>_linkName</i>	Name of the LinkState (p. 416)
-----------	------------------	---------------------------------------

Returns

State (p. 702) of the **Link** (p. 398).

Exceptions

<i>common::Exception</i> (p. 303)	When <code>_linkName</code> is invalid.
---	---

10.85.3.7 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. 416) recorded.

10.85.3.8 const std::vector<LinkState>& gazebo::physics::ModelState::GetLinkStates () const

Get the link states.

Returns

A vector of link states.

10.85.3.9 const math::Pose& gazebo::physics::ModelState::GetPose () const

Get the stored model pose.

Returns

The **math::Pose** (p. 556) of the **Model** (p. 460).

10.85.3.10 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.

10.85.3.11 virtual void gazebo::physics::ModelState::Load (const sdf::ElementPtr *_elem*) [virtual]

Load state from SDF element.

Load **ModelState** (p. 477) 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. 704).

10.85.3.12 ModelState gazebo::physics::ModelState::operator+ (const ModelState & *_state*) const

Addition operator.

Parameters

<i>in</i>	<i>_pt</i>	A state to subtract.
-----------	------------	----------------------

Returns

The resulting state.

10.85.3.13 ModelState gazebo::physics::ModelState::operator- (const ModelState & *_state*) const

Subtraction operator.

Parameters

<i>in</i>	<i>_pt</i>	A state to subtract.
-----------	------------	----------------------

Returns

The resulting state.

10.85.3.14 ModelState& gazebo::physics::ModelState::operator= (const ModelState & *_state*)

Assignment operator.

Parameters

<i>in</i>	<i>_state</i>	State (p. 702) value
-----------	---------------	-----------------------------

Returns

this

10.85.4 Friends And Related Function Documentation

10.85.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::physics::ModelState & *_state*) [friend]

Stream insertion operator.

Parameters

<i>in</i>	<i>_out</i>	output stream.
<i>in</i>	<i>_state</i>	Model (p. 460) state to output.

Returns

The stream.

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

- **ModelState.hh**

10.86 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.

10.86.1 Detailed Description

Generic description of a mouse event.

10.86.2 Member Enumeration Documentation

10.86.2.1 enum gazebo::common::MouseEvent::Buttons

Standard mouse buttons enumeration.

Enumerator:

NO_BUTTON

LEFT

MIDDLE

RIGHT

10.86.2.2 enum gazebo::common::MouseEvent::EventType

Mouse event types enumeration.

Enumerator:

NO_EVENT

MOVE

PRESS

RELEASE

SCROLL

10.86.3 Constructor & Destructor Documentation

10.86.3.1 gazebo::common::MouseEvent::MouseEvent () [inline]

Constructor.

10.86.4 Member Data Documentation

10.86.4.1 bool gazebo::common::MouseEvent::alt

Alt key press flag.

10.86.4.2 unsigned int gazebo::common::MouseEvent::button

The button which caused the event.

10.86.4.3 unsigned int gazebo::common::MouseEvent::buttons

State of the buttons when the event was generated.

10.86.4.4 bool gazebo::common::MouseEvent::control

Control key press flag.

10.86.4.5 bool gazebo::common::MouseEvent::dragging

Flag for mouse drag motion.

10.86.4.6 float gazebo::common::MouseEvent::moveScale

Scaling factor.

10.86.4.7 math::Vector2i gazebo::common::MouseEvent::pos

Mouse pointer position on the screen.

10.86.4.8 math::Vector2i gazebo::common::MouseEvent::pressPos

Position of button press.

10.86.4.9 math::Vector2i gazebo::common::MouseEvent::prevPos

Previous position.

10.86.4.10 math::Vector2i gazebo::common::MouseEvent::scroll

Scroll position.

10.86.4.11 bool gazebo::common::MouseEvent::shift

Shift key press flag.

10.86.4.12 EventType gazebo::common::MouseEvent::type

Event type.

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

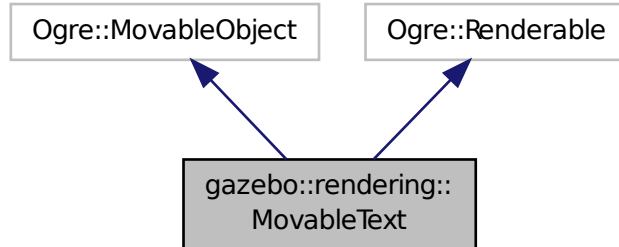
- **MouseEvent.hh**

10.87 gazebo::rendering::MovableText Class Reference

Movable text.

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

Inheritance diagram for gazebo::rendering::MovableText:



Public Types

- enum **HorizAlign** { **H_LEFT**, **H_CENTER** }
Horizontal alignment.
- enum **VertAlign** { **V_BELOW**, **V_ABOVE** }
vertical alignment

Public Member Functions

- **MovableText** ()
Constructor.
- virtual **~MovableText** ()
Destructor.
- **math::Box GetAABB** ()
Get the axis aligned bounding box of the text.
- float **GetBaseline** () const
Get the baseline height.
- float **GetCharHeight** () const
Set the height of a characters return Height of the characters.
- const **common::Color** & **GetColor** () const
Get the text color.
- const std::string & **GetFont** () const
Get the font.
- bool **GetShowOnTop** () const
True = text is displayed on top.
- float **GetSpaceWidth** () const

Get the width of a space.

- const std::string & **GetText** () const

Get the displayed text.

- void **Load** (const std::string &_name, const std::string &_text, const std::string &_fontName="Arial", float _charHeight=1.0, const **common::Color** &_color=**common::Color::White**)

Loads text and font info.

- void **SetBaseline** (float _height)

Set the baseline height of the text.

- void **SetCharHeight** (float _height)

Set the height of a character.

- void **SetColor** (const **common::Color** &_color)

Set the text color.

- void **SetFontName** (const std::string &_font)

Set the font.

- void **SetShowOnTop** (bool _show)

True = text always is displayed on top.

- void **SetSpaceWidth** (float _width)

Set the width of a space.

- void **SetText** (const std::string &_text)

Set the text to display.

- void **SetTextAlignment** (const **HorizAlign** &_hAlign, const **VertAlign** &_vAlign)

Set the alignment of the text.

- void **Update** ()

Update the text.

- virtual void **visitRenderables** (Ogre::Renderable::Visitor *_visitor, bool _debug=false)

Protected Member Functions

- void **_setupGeometry** ()
- void **_updateColors** ()
- float **getBoundingRadius** () const
- const Ogre::LightList & **getLights** (void) const
- const Ogre::MaterialPtr & **getMaterial** (void) const
- void **getRenderOperation** (Ogre::RenderOperation &op)
- float **getSquaredViewDepth** (const Ogre::Camera *cam) const
- void **getWorldTransforms** (Ogre::Matrix4 *xform) const

10.87.1 Detailed Description

Movable text.

10.87.2 Member Enumeration Documentation

10.87.2.1 enum gazebo::rendering::MovableText::HorizAlign

Horizontal alignment.

Enumerator:

H_LEFT Left alignment.

H_CENTER Center alignment.

10.87.2.2 enum gazebo::rendering::MovableText::VertAlign

vertical alignment

Enumerator:

V_BELOW Align below.

V_ABOVE Align above.

10.87.3 Constructor & Destructor Documentation

10.87.3.1 gazebo::rendering::MovableText::MovableText ()

Constructor.

10.87.3.2 virtual gazebo::rendering::MovableText::~MovableText () [virtual]

Destructor.

10.87.4 Member Function Documentation

10.87.4.1 void gazebo::rendering::MovableText::_setupGeometry () [protected]

10.87.4.2 void gazebo::rendering::MovableText::_updateColors () [protected]

10.87.4.3 math::Box gazebo::rendering::MovableText::GetAABB ()

Get the axis aligned bounding box of the text.

Returns

The axis aligned bounding box.

10.87.4.4 float gazebo::rendering::MovableText::GetBaseline () const

Get the baseline height.

Returns

Baseline height

10.87.4.5 float gazebo::rendering::MovableText::getBoundingRadius () const [protected]

10.87.4.6 float gazebo::rendering::MovableText::GetCharHeight () const

Set the height of a characters return Height of the characters.

10.87.4.7 const common::Color& gazebo::rendering::MovableText::GetColor () const

Get the text color.

Returns

Texture color.

10.87.4.8 const std::string& gazebo::rendering::MovableText::GetFont () const

Get the font.

Returns

The font name

10.87.4.9 const Ogre::LightList& gazebo::rendering::MovableText::getLights (void) const [protected]

10.87.4.10 const Ogre::MaterialPtr& gazebo::rendering::MovableText::getMaterial (void) const [protected]

10.87.4.11 void gazebo::rendering::MovableText::getRenderOperation (Ogre::RenderOperation & op) [protected]

10.87.4.12 bool gazebo::rendering::MovableText::GetShowOnTop () const

True = text is displayed on top.

Returns

True if MovableText::SetShownOnTop(true) was called.

10.87.4.13 float gazebo::rendering::MovableText::GetSpaceWidth () const

Get the width of a space.

Returns

Space width

10.87.4.14 float gazebo::rendering::MovableText::getSquaredViewDepth (const Ogre::Camera * cam) const [protected]

10.87.4.15 const std::string& gazebo::rendering::MovableText::GetText () const

Get the displayed text.

Returns

The displayed text.

10.87.4.16 `void gazebo::rendering::MovableText::getWorldTransforms (Ogre::Matrix4 * xform) const` [protected]

10.87.4.17 `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

10.87.4.18 `void gazebo::rendering::MovableText::SetBaseline (float _height)`

Set the baseline height of the text.

Parameters

in	<i>_height</i>	Baseline height
----	----------------	-----------------

10.87.4.19 `void gazebo::rendering::MovableText::SetCharHeight (float _height)`

Set the height of a character.

Parameters

in	<i>_height</i>	Height of the characters.
----	----------------	---------------------------

10.87.4.20 `void gazebo::rendering::MovableText::SetColor (const common::Color & _color)`

Set the text color.

Parameters

in	<i>_color</i>	Text color.
----	---------------	-------------

10.87.4.21 `void gazebo::rendering::MovableText::SetFontName (const std::string & _font)`

Set the font.

Parameters

in	<i>_font</i>	Name of the font
----	--------------	------------------

10.87.4.22 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.
----	--------------	---

10.87.4.23 void gazebo::rendering::MovableText::SetSpaceWidth (float *_width*)

Set the width of a space.

Parameters

in	<i>_width</i>	space width
----	---------------	-------------

10.87.4.24 void gazebo::rendering::MovableText::SetText (const std::string & *_text*)

Set the text to display.

Parameters

in	<i>_text</i>	The text to display.
----	--------------	----------------------

10.87.4.25 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

10.87.4.26 void gazebo::rendering::MovableText::Update ()

Update the text.

10.87.4.27 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:

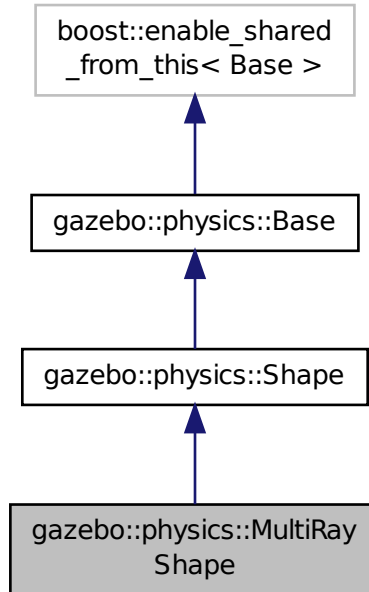
- **MovableText.hh**

10.88 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

10.88.1 Detailed Description

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

10.88.2 Constructor & Destructor Documentation

10.88.2.1 `gazebo::physics::MultiRayShape::MultiRayShape (CollisionPtr _parent) [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Parent collision shape.
----	----------------------	-------------------------

10.88.2.2 `virtual gazebo::physics::MultiRayShape::~~MultiRayShape () [virtual]`

Destructor.

10.88.3 Member Function Documentation

10.88.3.1 `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.

10.88.3.2 `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.

10.88.3.3 void gazebo::physics::MultiRayShape::DisconnectNewLaserScans (event::ConnectionPtr &_conn) [inline]

Disconnect from the new laser scans signal.

Parameters

in	<code>_conn</code>	Connection to remove.
----	--------------------	-----------------------

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

10.88.3.4 void gazebo::physics::MultiRayShape::FillMsg (msgs::Geometry &_msg) [virtual]

This function is not implemented.

Fill a message with this shape's values.

Parameters

out	<code>_msg</code>	Message that contains the shape's values.
-----	-------------------	---

Implements gazebo::physics::Shape (p. 669).

10.88.3.5 int gazebo::physics::MultiRayShape::GetFiducial (int _index)

Get detected fiducial value for a ray.

Parameters

in	<code>_index</code>	Index of the ray.
----	---------------------	-------------------

Returns

Fiducial value for the ray.

10.88.3.6 math::Angle gazebo::physics::MultiRayShape::GetMaxAngle () const

Get the maximum angle.

Returns

Maximum angle of ray scan.

10.88.3.7 `double gazebo::physics::MultiRayShape::GetMaxRange () const`

Get the maximum range.

Returns

Maximum range of all the rays.

10.88.3.8 `math::Angle gazebo::physics::MultiRayShape::GetMinAngle () const`

Get the minimum angle.

Returns

Minimum angle of ray scan.

10.88.3.9 `double gazebo::physics::MultiRayShape::GetMinRange () const`

Get the minimum range.

Returns

Minimum range of all the rays.

10.88.3.10 `double gazebo::physics::MultiRayShape::GetRange (int _index)`

Get detected range for a ray.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the ray.
-----------------	----------------------------	-------------------

Returns

Returns DBL_MAX for no detection.

10.88.3.11 `double gazebo::physics::MultiRayShape::GetResRange () const`

Get the range resolution.

Returns

Range resolution of all the rays.

10.88.3.12 `double gazebo::physics::MultiRayShape::GetRetro (int _index)`

Get detected retro (intensity) value for a ray.

Parameters

<code>in</code>	<code>_index</code>	Index of the ray.
-----------------	---------------------	-------------------

Returns

Retro value for the ray.

10.88.3.13 `int gazebo::physics::MultiRayShape::GetSampleCount () const`

Get the horizontal sample count.

Returns

Horizontal sample count.

10.88.3.14 `double gazebo::physics::MultiRayShape::GetScanResolution () const`

Get the horizontal resolution.

Returns

Horizontal resolution.

10.88.3.15 `math::Angle gazebo::physics::MultiRayShape::GetVerticalMaxAngle () const`

Get the vertical max angle.

Returns

Vertical max angle.

10.88.3.16 `math::Angle gazebo::physics::MultiRayShape::GetVerticalMinAngle () const`

Get the vertical min angle.

Returns

Vertical min angle.

10.88.3.17 `int gazebo::physics::MultiRayShape::GetVerticalSampleCount () const`

Get the vertical sample count.

Returns

Vertical sample count.

10.88.3.18 `double gazebo::physics::MultiRayShape::GetVerticalScanResolution () const`

Get the vertical range resolution.

Returns

Vertical range resolution.

10.88.3.19 `virtual void gazebo::physics::MultiRayShape::Init () [virtual]`

Init the shape.

Implements **`gazebo::physics::Shape`** (p. 670).

10.88.3.20 `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. 670).

10.88.3.21 `void gazebo::physics::MultiRayShape::Update () [virtual]`

Update the ray collisions.

Reimplemented from **`gazebo::physics::Base`** (p. 135).

10.88.3.22 `virtual void gazebo::physics::MultiRayShape::UpdateRays () [protected],[pure virtual]`

Physics engine specific method for updating the rays.

10.88.4 Member Data Documentation

10.88.4.1 `sdf::ElementPtr gazebo::physics::MultiRayShape::horzElem [protected]`

Horizontal SDF element pointer.

10.88.4.2 `event::EventT<void()> gazebo::physics::MultiRayShape::newLaserScans [protected]`

New laser scans event.

Referenced by `ConnectNewLaserScans()`, and `DisconnectNewLaserScans()`.

10.88.4.3 `math::Pose gazebo::physics::MultiRayShape::offset [protected]`

Pose offset of all the rays.

10.88.4.4 `sdf::ElementPtr gazebo::physics::MultiRayShape::rangeElem` [protected]

Range SDF element pointer.

10.88.4.5 `sdf::ElementPtr gazebo::physics::MultiRayShape::rayElem` [protected]

Ray SDF element pointer.

10.88.4.6 `std::vector<RayShapePtr> gazebo::physics::MultiRayShape::rays` [protected]

Ray data.

10.88.4.7 `sdf::ElementPtr gazebo::physics::MultiRayShape::scanElem` [protected]

Scan SDF element pointer.

10.88.4.8 `sdf::ElementPtr gazebo::physics::MultiRayShape::vertElem` [protected]

Vertical SDF element pointer.

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

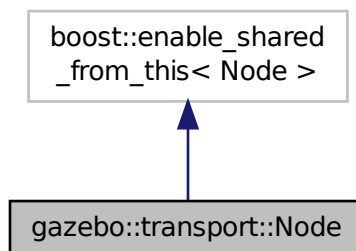
- **MultiRayShape.hh**

10.89 gazebo::transport::Node Class Reference

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

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

Inheritance diagram for gazebo::transport::Node:



Public Member Functions

- **Node** ()
Constructor.
- virtual **~Node** ()
Destructor.
- template<typename M >
transport::PublisherPtr Advertise (const std::string &_topic, unsigned int _queueLimit=1000, bool _latch=false)
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.
- 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.
- 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.

10.89.1 Detailed Description

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

10.89.2 Constructor & Destructor Documentation

10.89.2.1 gazebo::transport::Node::Node ()

Constructor.

10.89.2.2 virtual gazebo::transport::Node::~~Node () [virtual]

Destructor.

10.89.3 Member Function Documentation

10.89.3.1 template<typename M > transport::PublisherPtr gazebo::transport::Node::Advertise (const std::string & *_topic*, unsigned int *_queueLimit* = 1000, bool *_latch* = false) [inline]

Advertise a topic.

Parameters

in	<i>_topic</i>	The topic to advertise
in	<i>_queueLimit</i>	The maximum number of outgoing messages to queue for delivery
in	<i>_latch</i>	If true, latch the last message; otherwise, don't latch

Returns

Pointer to new publisher object

References DecodeTopicName(), and SingletonT< T >::Instance().

10.89.3.2 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().

10.89.3.3 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

10.89.3.4 void gazebo::transport::Node::Fini ()

Finalize the node.

10.89.3.5 unsigned int gazebo::transport::Node::GetId () const

Get the unique ID of the node.

Returns

The unique ID of the node

10.89.3.6 std::string gazebo::transport::Node::GetMsgType (const std::string & *_topic*) const

Get the message type for a topic.

Parameters

<i>in</i>	<i>_topic</i>	The topic
-----------	---------------	-----------

Returns

The message type

10.89.3.7 std::string gazebo::transport::Node::GetTopicNamespace () const

Get the topic namespace for this node.

Returns

The namespace

10.89.3.8 bool gazebo::transport::Node::HandleData (const std::string & *_topic*, const std::string & *_msg*)

Handle incoming data.

Parameters

<i>in</i>	<i>_topic</i>	Topic for which the data was received
<i>in</i>	<i>_msg</i>	The message that was received

Returns

true if the message was handled successfully, false otherwise

10.89.3.9 void gazebo::transport::Node::Init (const std::string & *_space* = " ")

Init the node.

Parameters

in	<code>_space</code>	Set the global namespace of all topics. If left blank, the topic will initialize to the first namespace on the Master (p. 426)
----	---------------------	---

10.89.3.10 `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

in	<code>_topic</code>	Name of the topic to publish data on.
in	<code>_msg</code>	The message to publish.

10.89.3.11 `void gazebo::transport::Node::ProcessIncoming ()`

Process incoming messages.

10.89.3.12 `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

10.89.3.13 `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. 718) object

References `DecodeTopicName()`, and `SingletonT< T >::Instance()`.

10.89.3.14 `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. 718) object

References `DecodeTopicName()`, and `SingletonT< T >::Instance()`.

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

- **Node.hh**

10.90 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

10.90.1 Detailed Description

Node animation.

10.90.2 Constructor & Destructor Documentation

10.90.2.1 gazebo::common::NodeAnimation::NodeAnimation (const std::string & *_name*)

constructor

Parameters

<code>in</code>	<code><i>_name</i></code>	the name of the node
-----------------	---------------------------	----------------------

10.90.2.2 gazebo::common::NodeAnimation::~~NodeAnimation ()

Destructor. It empties the key frames list.

10.90.3 Member Function Documentation

10.90.3.1 void gazebo::common::NodeAnimation::AddKeyFrame (const double *_time*, const **math::Matrix4** *_trans*)

Adds a key frame at a specific time.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time of the key frame
<code>in</code>	<code><i>_trans</i></code>	the transformation

10.90.3.2 void gazebo::common::NodeAnimation::AddKeyFrame (const double *_time*, const **math::Pose** *_pose*)

Adds a key fram at a specific time.

Parameters

<code>in</code>	<code><i>_time</i></code>	the tiem of the key frame
<code>in</code>	<code><i>_pose</i></code>	the pose

10.90.3.3 `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	<code><i>_time</i></code>	the time
in	<code><i>_loop</i></code>	when true, the time is divided by the duration (see <code>GetLength</code>)

10.90.3.4 `unsigned int gazebo::common::NodeAnimation::GetFrameCount () const`

Returns the number of key frames.

Returns

the count

10.90.3.5 `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	<code><i>_i</i></code>	the index
out	<code><i>_time</i></code>	the time of the frame, or -1 if the index id is out of bounds
out	<code><i>_trans</i></code>	the transformation for this key frame

10.90.3.6 `std::pair<double, math::Matrix4> gazebo::common::NodeAnimation::GetKeyFrame (const unsigned int _i) const`

Returns a key frame using the index.

Note the index of a key frame can change as frames are added.

Parameters

in	<code><i>_i</i></code>	the index
----	------------------------	-----------

Returns

a pair that contains the time and transformation. **Time** (p. 732) is -1 if the index is out of bounds

10.90.3.7 `double gazebo::common::NodeAnimation::GetLength () const`

Returns the duration of the animations.

Returns

the time of the last animation

10.90.3.8 `std::string gazebo::common::NodeAnimation::GetName () const`

Returns the name.

Returns

the name

10.90.3.9 `double gazebo::common::NodeAnimation::GetTimeAtX (const double _x) const`

Returns the time where a transformation's translational value along the X axis is equal to `_x`.

When no transformation is found (within a tolerance of 1e-6), the time is interpolated.

Parameters

<code>in</code>	<code>_x</code>	the value along x. You must ensure that <code>_x</code> is within a valid range.
-----------------	-----------------	--

10.90.3.10 `void gazebo::common::NodeAnimation::Scale (const double _scale)`

Scales each transformation in the key frames.

This only affects the translational values.

Parameters

<code>in</code>	<code>_scale</code>	the scaling factor
-----------------	---------------------	--------------------

10.90.3.11 `void gazebo::common::NodeAnimation::SetName (const std::string & _name)`

Changes the name of the animation.

Parameters

<code>in</code>	<code>the</code>	new name
-----------------	------------------	----------

10.90.4 Member Data Documentation**10.90.4.1** `std::map<double, math::Matrix4> gazebo::common::NodeAnimation::keyFrames` `[protected]`

the dictionary of key frames, indexed by time

10.90.4.2 `double gazebo::common::NodeAnimation::length` `[protected]`

the duration of the animations (time of last key frame)

10.90.4.3 `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**

10.91 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)

10.91.1 Detailed Description

Vertex to node weighted assignment for skeleton animation visualization.

10.91.2 Member Data Documentation

10.91.2.1 unsigned int `gazebo::common::NodeAssignment::nodeIndex`

node (or bone) index

10.91.2.2 unsigned int `gazebo::common::NodeAssignment::vertexIndex`

index of the vertex

10.91.2.3 float `gazebo::common::NodeAssignment::weight`

the weight (between 0 and 1)

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

- **Mesh.hh**

10.92 gazebo::common::NodeTransform Class Reference

NodeTransform (p. 509) **Skeleton.hh** (p. 1029) 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

10.92.1 Detailed Description

NodeTransform (p. 509) **Skeleton.hh** (p. 1029) `common/common.hh`

A transformation node

10.92.2 Member Enumeration Documentation

10.92.2.1 `enum gazebo::common::NodeTransform::TransformType`

Enumeration of the transform types.

Enumerator:

TRANSLATE
ROTATE
SCALE
MATRIX

10.92.3 Constructor & Destructor Documentation

10.92.3.1 `gazebo::common::NodeTransform::NodeTransform (TransformType _type = MATRIX)`

Constructor.

Parameters

<code>in</code>	<code>_type</code>	the type of transform
-----------------	--------------------	-----------------------

10.92.3.2 `gazebo::common::NodeTransform::NodeTransform (math::Matrix4 _mat, std::string _sid = "_default_", TransformType _type = MATRIX)`

Constructor.

Parameters

<code>in</code>	<code>_mat</code>	the matrix
<code>in</code>	<code>_sid</code>	identifier
<code>in</code>	<code>_type</code>	the type of transform

10.92.3.3 gazebo::common::NodeTransform::~~NodeTransform ()

Destructor. It does nothing.

10.92.4 Member Function Documentation

10.92.4.1 math::Matrix4 gazebo::common::NodeTransform::Get ()

Returns the transformation matrix.

Returns

the matrix

10.92.4.2 std::string gazebo::common::NodeTransform::GetSID ()

Returns the SID.

Returns

the SID

10.92.4.3 TransformType gazebo::common::NodeTransform::GetType ()

Returns the transformation type.

Returns

the type

10.92.4.4 math::Matrix4 gazebo::common::NodeTransform::operator() ()

Matrix cast operator.

Returns

the transform

10.92.4.5 math::Matrix4 gazebo::common::NodeTransform::operator* (NodeTransform _t)

Node transform multiplication operator.

Parameters

in	<code>_t</code>	a transform
----	-----------------	-------------

Returns

transform matrix multiplied by `_t`'s transform

10.92.4.6 `math::Matrix4 gazebo::common::NodeTransform::operator*(math::Matrix4 _m)`

Matrix multiplication operator.

Parameters

<code>in</code>	<code>_m</code>	a matrix
-----------------	-----------------	----------

Returns

transform matrix multiplied by `_m`

10.92.4.7 `void gazebo::common::NodeTransform::PrintSource ()`

Prints the transform matrix to `std::err` stream.

10.92.4.8 `void gazebo::common::NodeTransform::RecalculateMatrix ()`

Sets the transform matrix from the source according to the type.

10.92.4.9 `void gazebo::common::NodeTransform::Set (math::Matrix4 _mat)`

Assign a transformation.

Parameters

<code>in</code>	<code>_mat</code>	the transform
-----------------	-------------------	---------------

10.92.4.10 `void gazebo::common::NodeTransform::SetComponent (unsigned int _idx, double _value)`

Set a transformation matrix component value.

Parameters

<code>in</code>	<code>_idx</code>	the component index
<code>in</code>	<code>_value</code>	the value

10.92.4.11 `void gazebo::common::NodeTransform::SetSID (std::string _sid)`

Set the SID.

Parameters

<code>in</code>	<code>_sid</code>	the sid
-----------------	-------------------	---------

10.92.4.12 void gazebo::common::NodeTransform::SetSourceValues (math::Matrix4 *_mat*)

Set source data values *_param[in]* *_mat* the values.

10.92.4.13 void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 *_vec*)

Set source data values.

10.92.4.14 void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 *_axis*, double *_angle*)

Sets source matrix values from rotation.

Parameters

<i>in</i>	<i>_axis</i>	of rotation
<i>in</i>	<i>_angle</i>	of rotation

10.92.4.15 void gazebo::common::NodeTransform::SetType (TransformType *_type*)

Set transform type.

Parameters

<i>in</i>	<i>_type</i>	the type
-----------	--------------	----------

10.92.5 Member Data Documentation

10.92.5.1 std::string gazebo::common::NodeTransform::sid [protected]

the sid

10.92.5.2 std::vector<double> gazebo::common::NodeTransform::source [protected]

source data values (can be a matrix, a position or rotation)

10.92.5.3 math::Matrix4 gazebo::common::NodeTransform::transform [protected]

transform

10.92.5.4 TransformType gazebo::common::NodeTransform::type [protected]

transform type

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

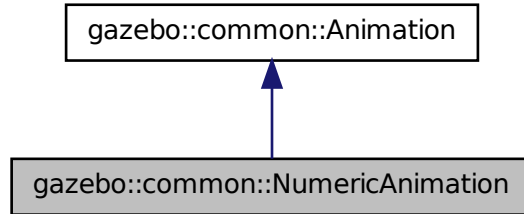
- **Skeleton.hh**

10.93 gazebo::common::NumericAnimation Class Reference

A numeric animation.

```
#include <Animation.hh>
```

Inheritance diagram for gazebo::common::NumericAnimation:



Public Member Functions

- **NumericAnimation** (const std::string &_name, double _length, bool _loop)
Constructor.
- virtual ~**NumericAnimation** ()
Destructor.
- **NumericKeyFrame * CreateKeyFrame** (double _time)
Create a numeric keyframe at the given time.
- void **GetInterpolatedKeyFrame** (**NumericKeyFrame** &_kf) const
Get a keyframe using the animation's current time.

Additional Inherited Members

10.93.1 Detailed Description

A numeric animation.

10.93.2 Constructor & Destructor Documentation

10.93.2.1 gazebo::common::NumericAnimation::NumericAnimation (const std::string & _name, double _length, bool _loop)

Constructor.

Parameters

in	<code>_name</code>	String name of the animation. This should be unique.
in	<code>_length</code>	Length of the animation in seconds
in	<code>_loop</code>	True == loop the animation

10.93.2.2 virtual gazebo::common::NumericAnimation::~~NumericAnimation () [virtual]

Destructor.

10.93.3 Member Function Documentation

10.93.3.1 NumericKeyFrame* gazebo::common::NumericAnimation::CreateKeyFrame (double *_time*)

Create a numeric keyframe at the given time.

Parameters

in	<i>_time</i>	Time (p. 732) at which to create the keyframe
----	--------------	--

Returns

Pointer to the new keyframe

10.93.3.2 void gazebo::common::NumericAnimation::GetInterpolatedKeyFrame (NumericKeyFrame & *_kf*) const

Get a keyframe using the animation's current time.

Parameters

out	<i>_kf</i>	NumericKeyFrame (p. 515) reference to hold the interpolated result
-----	------------	---

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

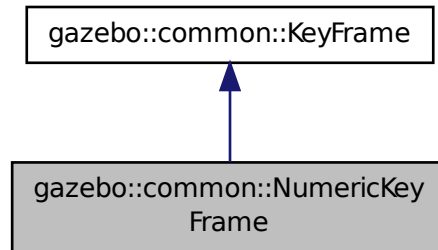
- **Animation.hh**

10.94 gazebo::common::NumericKeyFrame Class Reference

A keyframe for a **NumericAnimation** (p. 514).

```
#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

10.94.1 Detailed Description

A keyframe for a **NumericAnimation** (p. 514).

10.94.2 Constructor & Destructor Documentation

10.94.2.1 gazebo::common::NumericKeyFrame::NumericKeyFrame (double *_time*)

Constructor.

Parameters

<i>in</i>	<i>_time</i>	Time (p. 732) of the keyframe
-----------	--------------	--------------------------------------

10.94.2.2 `virtual gazebo::common::NumericKeyFrame::~~NumericKeyFrame () [virtual]`

Destructor.

10.94.3 Member Function Documentation

10.94.3.1 `const double& gazebo::common::NumericKeyFrame::GetValue () const`

Get the value of the keyframe.

Returns

the value of the keyframe

10.94.3.2 `void gazebo::common::NumericKeyFrame::SetValue (const double & _value)`

Set the value of the keyframe.

Parameters

<code>in</code>	<code>_value</code>	The new value
-----------------	---------------------	---------------

10.94.4 Member Data Documentation

10.94.4.1 `double gazebo::common::NumericKeyFrame::value [protected]`

numeric value

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

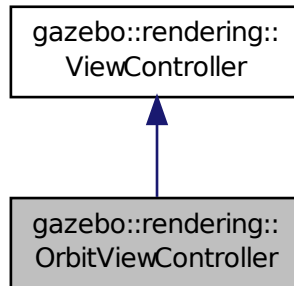
- [KeyFrame.hh](#)

10.95 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 **SetDistanceRange** (double _minDist, double _maxDist)
Set the min and max distance from the focal point.
- void **SetFocalPoint** (const **math::Vector3** &_fp)
Set the focal point.
- void **SetPitch** (double _pitch)
Set the pitch angle of the camera.
- void **SetYaw** (double _yaw)
Set the yaw angle of the camera.
- virtual void **Update** ()
Update.

Static Public Member Functions

- static std::string **GetTypeString** ()
Get the type name of this view controller.

Additional Inherited Members

10.95.1 Detailed Description

Orbit view controller.

10.95.2 Constructor & Destructor Documentation

10.95.2.1 gazebo::rendering::OrbitViewController::OrbitViewController (UserCameraPtr _camera)

Constructor.

Parameters

in	_camera	Pointer to the camera to control.
----	---------	-----------------------------------

10.95.2.2 virtual gazebo::rendering::OrbitViewController::~~OrbitViewController () [virtual]

Destructor.

10.95.3 Member Function Documentation

10.95.3.1 math::Vector3 gazebo::rendering::OrbitViewController::GetFocalPoint () const

Get the focal point.

Returns

The focal point

10.95.3.2 static std::string gazebo::rendering::OrbitViewController::GetTypeString () [static]

Get the type name of this view controller.

Returns

The view controller name: "orbit".

10.95.3.3 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. 826).

10.95.3.4 `void gazebo::rendering::OrbitViewController::HandleKeyReleaseEvent (const std::string & _key) [virtual]`

Handle a key release event.

Parameters

in	<code>_key</code>	The key that was released.
----	-------------------	----------------------------

Implements **gazebo::rendering::ViewController** (p. 827).

10.95.3.5 `virtual void gazebo::rendering::OrbitViewController::HandleMouseEvent (const common::MouseEvent & _event) [virtual]`

Handle a mouse event.

Parameters

in	<code>_event</code>	The mouse event.
----	---------------------	------------------

Implements **gazebo::rendering::ViewController** (p. 827).

10.95.3.6 `virtual void gazebo::rendering::OrbitViewController::Init () [virtual]`

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 827).

10.95.3.7 `virtual void gazebo::rendering::OrbitViewController::Init (const math::Vector3 & _focalPoint) [virtual]`

Initialize the controller with a focal point.

Parameters

in	<code>_focalPoint</code>	Point to look at.
----	--------------------------	-------------------

Reimplemented from **gazebo::rendering::ViewController** (p. 827).

10.95.3.8 `void gazebo::rendering::OrbitViewController::SetDistance (float _d)`

Set the distance to the focal point.

Parameters

in	<code>_d</code>	The distance from the focal point.
----	-----------------	------------------------------------

10.95.3.9 void gazebo::rendering::OrbitViewController::SetDistanceRange (double *_minDist*, double *_maxDist*)

Set the min and max distance from the focal point.

Parameters

in	<i>_minDist</i>	Min distance to the focal point.
in	<i>_maxDist</i>	Max distance from the focal point.

10.95.3.10 void gazebo::rendering::OrbitViewController::SetFocalPoint (const math::Vector3 & *_fp*)

Set the focal point.

Parameters

in	<i>_fp</i>	The focal point
----	------------	-----------------

10.95.3.11 void gazebo::rendering::OrbitViewController::SetPitch (double *_pitch*)

Set the pitch angle of the camera.

Parameters

in	<i>_pitch</i>	Pitch angle in radians.
----	---------------	-------------------------

10.95.3.12 void gazebo::rendering::OrbitViewController::SetYaw (double *_yaw*)

Set the yaw angle of the camera.

Parameters

in	<i>_yaw</i>	angle in radians
----	-------------	------------------

10.95.3.13 virtual void gazebo::rendering::OrbitViewController::Update () [virtual]

Update.

Implements **gazebo::rendering::ViewController** (p. 828).

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

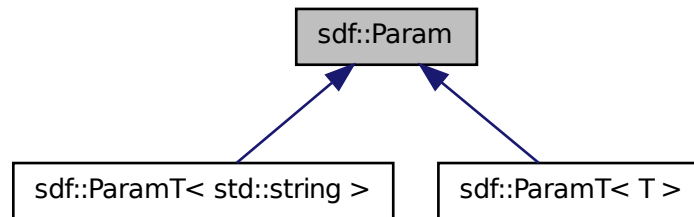
- **OrbitViewController.hh**

10.96 sdf::Param Class Reference

A 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**

10.96.1 Detailed Description

A parameter class.

10.96.2 Constructor & Destructor Documentation

10.96.2.1 sdf::Param::Param (Param * *_newParam*)

Constructor.

10.96.2.2 virtual sdf::Param::~~Param () [virtual]

Destructor.

10.96.3 Member Function Documentation

10.96.3.1 virtual Param* sdf::Param::Clone () const [pure virtual]

Implemented in **sdf::ParamT< T >** (p. 529), and **sdf::ParamT< std::string >** (p. 529).

10.96.3.2 bool sdf::Param::Get (bool & *_value*)

10.96.3.3 bool sdf::Param::Get (int & *_value*)

10.96.3.4 bool sdf::Param::Get (unsigned int & *_value*)

10.96.3.5 bool sdf::Param::Get (float & *_value*)

10.96.3.6 bool sdf::Param::Get (double & *_value*)

10.96.3.7 bool sdf::Param::Get (char & *_value*)

10.96.3.8 bool sdf::Param::Get (std::string & *_value*)

10.96.3.9 bool sdf::Param::Get (gazebo::math::Vector3 & *_value*)

10.96.3.10 bool sdf::Param::Get (gazebo::math::Vector2i & *_value*)

10.96.3.11 bool sdf::Param::Get (gazebo::math::Vector2d & *_value*)

10.96.3.12 bool sdf::Param::Get (gazebo::math::Quaternion & *_value*)

10.96.3.13 bool sdf::Param::Get (gazebo::math::Pose & *_value*)

10.96.3.14 bool sdf::Param::Get (gazebo::common::Color & *_value*)

10.96.3.15 bool sdf::Param::Get (gazebo::common::Time & *_value*)

10.96.3.16 virtual std::string sdf::Param::GetAsString () const [inline],[virtual]

Get the type.

Reimplemented in **sdf::ParamT< T >** (p. 529), and **sdf::ParamT< std::string >** (p. 529).

10.96.3.17 virtual std::string sdf::Param::GetDefaultAsString () const [inline],[virtual]

Reimplemented in **sdf::ParamT< T >** (p. 529), and **sdf::ParamT< std::string >** (p. 529).

10.96.3.18 std::string sdf::Param::GetDescription () const

Get the description of the parameter.

10.96.3.19 const std::string& sdf::Param::GetKey () const [inline]

References key.

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

10.96.3.20 bool sdf::Param::GetRequired () const [inline]

References required.

10.96.3.21 bool sdf::Param::GetSet () const [inline]

Return true if the parameter has been set.

10.96.3.22 std::string sdf::Param::GetTypeNames () const

10.96.3.23 bool sdf::Param::IsBool () const

10.96.3.24 bool sdf::Param::IsChar () const

10.96.3.25 bool sdf::Param::IsColor () const

10.96.3.26 bool sdf::Param::IsDouble () const

10.96.3.27 bool sdf::Param::IsFloat () const

10.96.3.28 bool sdf::Param::IsInt () const

10.96.3.29 bool sdf::Param::IsPose () const

10.96.3.30 bool sdf::Param::IsQuaternion () const

10.96.3.31 bool sdf::Param::IsStr () const

10.96.3.32 bool sdf::Param::IsTime () const

10.96.3.33 bool sdf::Param::IsUInt () const

10.96.3.34 bool sdf::Param::IsVector2d () const

10.96.3.35 bool sdf::Param::IsVector2i () const

10.96.3.36 `bool sdf::Param::IsVector3 () const`

10.96.3.37 `virtual void sdf::Param::Reset () [pure virtual]`

Reset the parameter.

Implemented in `sdf::ParamT< T >` (p. 530), and `sdf::ParamT< std::string >` (p. 530).

10.96.3.38 `bool sdf::Param::Set (const bool & _value)`

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

10.96.3.39 `bool sdf::Param::Set (const int & _value)`

10.96.3.40 `bool sdf::Param::Set (const unsigned int & _value)`

10.96.3.41 `bool sdf::Param::Set (const float & _value)`

10.96.3.42 `bool sdf::Param::Set (const double & _value)`

10.96.3.43 `bool sdf::Param::Set (const char & _value)`

10.96.3.44 `bool sdf::Param::Set (const std::string & _value)`

10.96.3.45 `bool sdf::Param::Set (const char * _value)`

10.96.3.46 `bool sdf::Param::Set (const gazebo::math::Vector3 & _value)`

10.96.3.47 `bool sdf::Param::Set (const gazebo::math::Vector2i & _value)`

10.96.3.48 `bool sdf::Param::Set (const gazebo::math::Vector2d & _value)`

10.96.3.49 `bool sdf::Param::Set (const gazebo::math::Quaternion & _value)`

10.96.3.50 `bool sdf::Param::Set (const gazebo::math::Pose & _value)`

10.96.3.51 `bool sdf::Param::Set (const gazebo::common::Color & _value)`

10.96.3.52 `bool sdf::Param::Set (const gazebo::common::Time & _value)`

10.96.3.53 `void sdf::Param::SetDescription (const std::string & _desc)`

Set the description of the parameter.

10.96.3.54 `virtual bool sdf::Param::SetFromString (const std::string &) [inline], [virtual]`

Set the parameter value from a string.

Reimplemented in `sdf::ParamT< T >` (p. 530), and `sdf::ParamT< std::string >` (p. 530).

10.96.3.55 `template<typename T> void sdf::Param::SetUpdateFunc (T updateFunc) [inline]`

Update function.

References `updateFunc`.

10.96.3.56 `virtual void sdf::Param::Update () [pure virtual]`

Implemented in `sdf::ParamT< T >` (p. 530), and `sdf::ParamT< std::string >` (p. 530).

10.96.4 Member Data Documentation

10.96.4.1 `std::string sdf::Param::description [protected]`

Referenced by `sdf::ParamT< std::string >::Clone()`, and `sdf::ParamT< std::string >::ParamT()`.

10.96.4.2 `std::string sdf::Param::key [protected]`

Referenced by `GetKey()`, `sdf::ParamT< std::string >::ParamT()`, and `sdf::ParamT< std::string >::Set()`.

10.96.4.3 `bool sdf::Param::required [protected]`

Referenced by `sdf::ParamT< std::string >::Clone()`, `GetRequired()`, `sdf::ParamT< std::string >::ParamT()`, and `sdf::ParamT< std::string >::Set()`.

10.96.4.4 `bool sdf::Param::set [protected]`

10.96.4.5 `std::string sdf::Param::typeName [protected]`

Referenced by `sdf::ParamT< std::string >::Clone()`, and `sdf::ParamT< std::string >::ParamT()`.

10.96.4.6 `boost::function<boost::any ()> sdf::Param::updateFunc [protected]`

Referenced by `SetUpdateFunc()`, and `sdf::ParamT< std::string >::Update()`.

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

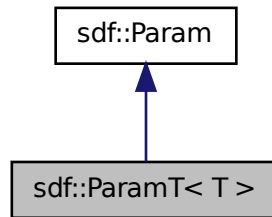
- **Param.hh**

10.97 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)

10.97.1 Detailed Description

template<typename T>class sdf::ParamT< T >

Templatized parameter class.

10.97.2 Constructor & Destructor Documentation

10.97.2.1 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.

10.97.2.2 template<typename T> virtual sdf::ParamT< T >::~~ParamT () [inline],[virtual]

Destructor.

10.97.3 Member Function Documentation

10.97.3.1 template<typename T> virtual Param* sdf::ParamT< T >::Clone () const [inline],[virtual]

Implements **sdf::Param** (p. 524).

10.97.3.2 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. 524).

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

10.97.3.3 template<typename T> virtual std::string sdf::ParamT< T >::GetDefaultAsString () const [inline],[virtual]

Reimplemented from **sdf::Param** (p. 525).

10.97.3.4 template<typename T> T sdf::ParamT< T >::GetDefaultValue () const [inline]

Get the value.

10.97.3.5 template<typename T> T sdf::ParamT< T >::GetValue () const [inline]

Get the value.

10.97.3.6 `template<typename T> T sdf::ParamT< T >::operator*() const [inline]`

10.97.3.7 `template<typename T> virtual void sdf::ParamT< T >::Reset() [inline],[virtual]`

Reset to default value.

Implements **sdf::Param** (p. 526).

10.97.3.8 `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()`.

10.97.3.9 `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. 526).

10.97.3.10 `template<typename T> void sdf::ParamT< T >::SetValue(const T & _value) [inline]`

Set the value of the parameter.

10.97.3.11 `template<typename T> virtual void sdf::ParamT< T >::Update() [inline],[virtual]`

Update param value.

Implements **sdf::Param** (p. 527).

10.97.4 Friends And Related Function Documentation

10.97.4.1 `template<typename T> std::ostream& operator<<(std::ostream & _out, const ParamT< T > & _p) [friend]`

10.97.5 Member Data Documentation

10.97.5.1 `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()`.

10.97.5.2 `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**

10.98 ParamT< T > Class Template Reference

```
#include <CommonTypes.hh>
```

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

- **CommonTypes.hh**

10.99 gazebo::physics::PhysicsEngine Class Reference

Base (p. 125) 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. 668) 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. 541).*
- virtual int **GetSORPGSIters** ()
: Remove this function, and replace it with a more generic property map
 - virtual int **GetSORPGSPreconIters** ()
: 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 **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.

10.99.1 Detailed Description

Base (p. 125) class for a physics engine.

10.99.2 Constructor & Destructor Documentation

10.99.2.1 gazebo::physics::PhysicsEngine::PhysicsEngine (WorldPtr _world) [explicit]

Default constructor.

Parameters

in	<code>_world</code>	Pointer to the world.
----	---------------------	-----------------------

10.99.2.2 virtual gazebo::physics::PhysicsEngine::~~PhysicsEngine () [virtual]

Destructor.

10.99.3 Member Function Documentation

10.99.3.1 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.

10.99.3.2 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.

10.99.3.3 virtual JointPtr gazebo::physics::PhysicsEngine::CreateJoint (const std::string & *_type*, ModelPtr *_parent*) [pure virtual]

Create a new joint.

Parameters

in	<i>_type</i>	Type of joint to create.
in	<i>_parent</i>	Model (p. 460) parent.

10.99.3.4 virtual LinkPtr gazebo::physics::PhysicsEngine::CreateLink (ModelPtr *_parent*) [pure virtual]

Create a new body.

Parameters

in	<i>_parent</i>	Parent model for the link.
----	----------------	----------------------------

10.99.3.5 virtual ShapePtr gazebo::physics::PhysicsEngine::CreateShape (const std::string & *_shapeType*, CollisionPtr *_collision*) [pure virtual]

Create a **physics::Shape** (p. 668) object.

Parameters

in	<i>_shapeType</i>	Type of shape to create.
in	<i>_collision</i>	Collision (p. 180) parent.

10.99.3.6 `virtual void gazebo::physics::PhysicsEngine::DebugPrint () const [pure virtual]`

Debug print out of the physic engine state.

10.99.3.7 `virtual void gazebo::physics::PhysicsEngine::Fini () [virtual]`

Finilize the physics engine.

10.99.3.8 `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.

10.99.3.9 `ContactManager* gazebo::physics::PhysicsEngine::GetContactManager () const`

Get a pointer to the contact manger.

Returns

Pointer to the contact manager.

10.99.3.10 `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.

10.99.3.11 `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. 218) suerface layer depth.

10.99.3.12 `virtual math::Vector3 gazebo::physics::PhysicsEngine::GetGravity () const [virtual]`

Return the gavity vector.

Returns

The gavity vector.

10.99.3.13 `virtual int gazebo::physics::PhysicsEngine::GetMaxContacts () [inline],[virtual]`

: Remove this function, and replace it with a more generic property map access functions to set ODE parameters.

Returns

Maximum number of allows contacts.

10.99.3.14 `boost::recursive_mutex* gazebo::physics::PhysicsEngine::GetPhysicsUpdateMutex () const [inline]`

returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 541).

Returns

Pointer to the physics mutex.

References physicsUpdateMutex.

10.99.3.15 `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.

10.99.3.16 `virtual int gazebo::physics::PhysicsEngine::GetSORPGSPreconlters () [inline],[virtual]`

: Remove this function, and replace it with a more generic property map access functions to set ODE parameters.

Returns

SORPGS precondition iterations.

10.99.3.17 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.

10.99.3.18 virtual double gazebo::physics::PhysicsEngine::GetStepTime () [pure virtual]

Get the simulation step time.

Returns

Simulation step time.

10.99.3.19 double gazebo::physics::PhysicsEngine::GetUpdatePeriod ()

Get the simulation update period.

Returns

Simulation update period.

10.99.3.20 double gazebo::physics::PhysicsEngine::GetUpdateRate ()

Get the simulation update rate.

Returns

Update rate.

10.99.3.21 virtual double gazebo::physics::PhysicsEngine::GetWorldCFM () [inline],[virtual]

: Remove this function, and replace it with a more generic property map

Get **World** (p. 853) CFM.

Returns

World (p. 853) CFM.

10.99.3.22 virtual double gazebo::physics::PhysicsEngine::GetWorldERP () [inline],[virtual]

: Remove this function, and replace it with a more generic property map

Get **World** (p. 853) ERP.

Returns

World (p. 853) ERP.

10.99.3.23 virtual void gazebo::physics::PhysicsEngine::Init () [pure virtual]

Initialize the physics engine.

10.99.3.24 virtual void gazebo::physics::PhysicsEngine::InitForThread () [pure virtual]

Init the engine for threads.

10.99.3.25 virtual void gazebo::physics::PhysicsEngine::Load (sdf::ElementPtr _sdf) [virtual]

Load the physics engine.

Parameters

in	<code>_sdf</code>	Pointer to the SDF parameters.
----	-------------------	--------------------------------

10.99.3.26 virtual void gazebo::physics::PhysicsEngine::OnPhysicsMsg (ConstPhysicsPtr & _msg) [protected],
[virtual]

virtual callback for gztopic "~/physics".

Parameters

in	<code>_msg</code>	Physics message.
----	-------------------	------------------

10.99.3.27 virtual void gazebo::physics::PhysicsEngine::OnRequest (ConstRequestPtr & _msg) [protected],
[virtual]

virtual callback for gztopic "~/request".

Parameters

in	<code>_msg</code>	Request message.
----	-------------------	------------------

10.99.3.28 virtual void gazebo::physics::PhysicsEngine::Reset () [inline],[virtual]

Rest the physics engine.

10.99.3.29 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.
----	---------------------------	--

10.99.3.30 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	<i>_vel</i>	Max correcting velocity.
----	-------------	--------------------------

10.99.3.31 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	<i>_layerDepth</i>	Surface layer depth
----	--------------------	---------------------

10.99.3.32 virtual void gazebo::physics::PhysicsEngine::SetGravity (const gazebo::math::Vector3 & *_gravity*) [pure virtual]

Set the gravity vector.

Parameters

in	<i>_gravity</i>	New gravity vector.
----	-----------------	---------------------

10.99.3.33 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	<i>_maxContacts</i>	Maximum number of contacts.
----	---------------------	-----------------------------

10.99.3.34 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	<i>_iter</i>	Number of iterations.
----	--------------	-----------------------

10.99.3.35 `virtual void gazebo::physics::PhysicsEngine::SetSORPGSPreconlters (unsigned int _iters) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

<code>in</code>	<code>_iter</code>	Number of iterations.
-----------------	--------------------	-----------------------

10.99.3.36 `virtual void gazebo::physics::PhysicsEngine::SetSORPGSW (double _w) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

<code>in</code>	<code>_w</code>	SORPGSW value.
-----------------	-----------------	----------------

10.99.3.37 `virtual void gazebo::physics::PhysicsEngine::SetStepTime (double _value) [pure virtual]`

Set the simulation step time.

Parameters

<code>in</code>	<code>_value</code>	Value of the step time.
-----------------	---------------------	-------------------------

10.99.3.38 `void gazebo::physics::PhysicsEngine::SetUpdateRate (double _value)`

Set the simulation update rate.

Parameters

<code>in</code>	<code>_value</code>	Value of the update rate.
-----------------	---------------------	---------------------------

10.99.3.39 `virtual void gazebo::physics::PhysicsEngine::SetWorldCFM (double _cfm) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

<code>in</code>	<code>_cfm</code>	Constraint force mixing.
-----------------	-------------------	--------------------------

10.99.3.40 `virtual void gazebo::physics::PhysicsEngine::SetWorldERP (double _erp) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

<code>in</code>	<code>_erp</code>	Error reduction parameter.
-----------------	-------------------	----------------------------

10.99.3.41 `virtual void gazebo::physics::PhysicsEngine::UpdateCollision () [pure virtual]`

Update the physics engine collision.

10.99.3.42 `virtual void gazebo::physics::PhysicsEngine::UpdatePhysics () [inline],[virtual]`

Update the physics engine.

10.99.4 Member Data Documentation

10.99.4.1 `ContactManager* gazebo::physics::PhysicsEngine::contactManager [protected]`

Class that handles all contacts generated by the physics engine.

10.99.4.2 `transport::NodePtr gazebo::physics::PhysicsEngine::node [protected]`

Node for communication.

10.99.4.3 `transport::SubscriberPtr gazebo::physics::PhysicsEngine::physicsSub [protected]`

Subscribe to the physics topic.

10.99.4.4 `boost::recursive_mutex* gazebo::physics::PhysicsEngine::physicsUpdateMutex [protected]`

Mutex to protect the update cycle.

Referenced by `GetPhysicsUpdateMutex()`.

10.99.4.5 `transport::SubscriberPtr gazebo::physics::PhysicsEngine::requestSub [protected]`

Subscribe to the request topic.

10.99.4.6 `transport::PublisherPtr gazebo::physics::PhysicsEngine::responsePub [protected]`

Response publisher.

10.99.4.7 `sdf::ElementPtr gazebo::physics::PhysicsEngine::sdf [protected]`

Our SDF values.

10.99.4.8 **WorldPtr** gazebo::physics::PhysicsEngine::world [protected]

Pointer to the world.

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

- **PhysicsEngine.hh**

10.100 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.

10.100.1 Detailed Description

The physics factory instantiates different physics engines.

10.100.2 Member Function Documentation

10.100.2.1 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>	World (p. 853) to pass to the created physics engine.

10.100.2.2 static void gazebo::physics::PhysicsFactory::RegisterAll () [static]

Register everything.

10.100.2.3 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**

10.101 gazebo::common::PID Class Reference

Generic **PID** (p. 543) 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. 543) controller.*
- void **GetErrors** (double &`_pe`, double &`_ie`, double &`_de`)
*Return **PID** (p. 543) 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. 543) controller.*
- void **SetCmdMax** (double `_c`)
Set the maximum value for the command.
- void **SetCmdMin** (double `_c`)
Set the maximum value for the command.
- void **SetDGain** (double `_d`)
Set the derivtive Gain.
- void **SetIGain** (double `_i`)
Set the integral Gain.
- void **SetIMax** (double `_i`)
Set the integral upper limit.
- void **SetIMin** (double `_i`)
Set the integral lower limit.
- void **SetPGain** (double `_p`)
Set the proportional Gain.
- double **Update** (double `_error`, **common::Time** `_dt`)
Update the Pid loop with nonuniform time step size.

10.101.1 Detailed Description

Generic **PID** (p. 543) 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.

10.101.2 Constructor & Destructor Documentation

10.101.2.1 `gazebo::common::PID::PID (double _p = 0.0, double _i = 0.0, double _d = 0.0, double _imax = 0.0, double _imin = 0.0, double _cmdMax = 0.0, double _cmdMin = 0.0)`

Constructor, zeros out Pid values when created and initialize Pid-gains and integral term limits:[iMax:iMin]-[I1:I2].

Parameters

in	<code>_p</code>	The proportional gain.
in	<code>_i</code>	The integral gain.
in	<code>_d</code>	The derivative gain.
in	<code>_imax</code>	The integral upper limit.
in	<code>_imin</code>	The integral lower limit.

10.101.2.2 `virtual gazebo::common::PID::~PID () [virtual]`

Destructor.

10.101.3 Member Function Documentation

10.101.3.1 `double gazebo::common::PID::GetCmd ()`

Return current command for this **PID** (p. 543) controller.

Returns

the command value

10.101.3.2 `void gazebo::common::PID::GetErrors (double & _pe, double & _ie, double & _de)`

Return **PID** (p. 543) error terms for the controller.

Parameters

in	<code>_pe</code>	The proportional error.
in	<code>_ie</code>	The integral error.
in	<code>_de</code>	The derivative error.

10.101.3.3 `void gazebo::common::PID::Init (double _p = 0.0, double _i = 0.0, double _d = 0.0, double _imax = 0.0, double _imin = 0.0, double _cmdMax = 0.0, double _cmdMin = 0.0)`

Initialize PID-gains and integral term limits:[iMax:iMin]-[I1:I2].

Parameters

in	<code>_p</code>	The proportional gain.
in	<code>_i</code>	The integral gain.
in	<code>_d</code>	The derivative gain.
in	<code>_imax</code>	The integral upper limit.
in	<code>_imin</code>	The integral lower limit.

10.101.3.4 `PID& gazebo::common::PID::operator=(const PID & _p) [inline]`

Assignment operator.

Parameters

in	<code>_p</code>	a reference to a PID (p. 543) to assign values from
----	-----------------	--

Returns

reference to this instance

References Reset().

10.101.3.5 `void gazebo::common::PID::Reset ()`

Reset the errors and command.

Referenced by operator=().

10.101.3.6 `void gazebo::common::PID::SetCmd (double _cmd)`

Set current target command for this **PID** (p. 543) controller.

Parameters

in	<code>_cmd</code>	New command
----	-------------------	-------------

10.101.3.7 `void gazebo::common::PID::SetCmdMax (double _c)`

Set the maximum value for the command.

Parameters

in	<code>_c</code>	The maximum value
----	-----------------	-------------------

10.101.3.8 void gazebo::common::PID::SetCmdMin (double *_c*)

Set the maximum value for the command.

Parameters

in	<i>_c</i>	The maximum value
----	-----------	-------------------

10.101.3.9 void gazebo::common::PID::SetDGain (double *_d*)

Set the derivative Gain.

Parameters

in	<i>_p</i>	derivative gain value
----	-----------	-----------------------

10.101.3.10 void gazebo::common::PID::SetIGain (double *_i*)

Set the integral Gain.

Parameters

in	<i>_p</i>	integral gain value
----	-----------	---------------------

10.101.3.11 void gazebo::common::PID::SetIMax (double *_i*)

Set the integral upper limit.

Parameters

in	<i>_p</i>	integral upper limit value
----	-----------	----------------------------

10.101.3.12 void gazebo::common::PID::SetIMin (double *_i*)

Set the integral lower limit.

Parameters

in	<i>_p</i>	integral lower limit value
----	-----------	----------------------------

10.101.3.13 void gazebo::common::PID::SetPGain (double *_p*)

Set the proportional Gain.

Parameters

in	<i>_p</i>	proportional gain value
----	-----------	-------------------------

10.101.3.14 double gazebo::common::PID::Update (double *_error*, common::Time *_dt*)

Update the Pid loop with nonuniform time step size.

Parameters

<i>_in]</i>	<i>_error</i> Error since last call (p_state - p_target).
<i>_in]</i>	<i>_dt</i> Change in time since last update call. Normally, this is called at every time step, The return value is an updated command to be passed to the object being controlled.

Returns

the command value

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

- **PID.hh**

10.102 gazebo::math::Plane Class Reference

A plane and related functions.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Plane** ()
Constructor.
- **Plane** (const **Vector3** &*_normal*, double *_offset*=0.0)
Constructor from a normal and a distanec.
- **Plane** (const **Vector3** &*_normal*, const **Vector2d** &*_size*, double *_offset*)
Constructor.
- virtual ~**Plane** ()
Destructor.
- double **Distance** (const **Vector3** &*_origin*, const **Vector3** &*_dir*) const
Get distance to the plane give an origin and direction.
- **Plane** & **operator=** (const **Plane** &*_p*)
Equal operator.
- void **Set** (const **Vector3** &*_normal*, const **Vector2d** &*_size*, double *offset*)
Set the plane.

Public Attributes

- double **d**
Plane (p. 547) *offset.*
- **Vector3** **normal**
Plane (p. 547) *normal.*
- **Vector2d** **size**
Plane (p. 547) *size.*

10.102.1 Detailed Description

A plane and related functions.

10.102.2 Constructor & Destructor Documentation

10.102.2.1 gazebo::math::Plane::Plane ()

Constructor.

10.102.2.2 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

10.102.2.3 gazebo::math::Plane::Plane (const Vector3 & *_normal*, const Vector2d & *_size*, double *_offset*)

Constructor.

Parameters

in	<i>_normal</i>	The plane normal
in	<i>_size</i>	Size of the plane
in	<i>_offset</i>	Offset along the normal

10.102.2.4 virtual gazebo::math::Plane::~~Plane () [virtual]

Destructor.

10.102.3 Member Function Documentation

10.102.3.1 double gazebo::math::Plane::Distance (const Vector3 & *_origin*, const Vector3 & *_dir*) const

Get distance to the plane given an origin and direction.

Parameters

in	<i>_origin</i>	the origin
in	<i>_dir</i>	a direction

Returns

the shortest distance

10.102.3.2 `Plane& gazebo::math::Plane::operator= (const Plane & _p)`

Equal operator.

Parameters

<code>_p</code>	another plane
-----------------	---------------

Returns

itself

10.102.3.3 `void gazebo::math::Plane::Set (const Vector3 & _normal, const Vector2d & _size, double offset)`

Set the plane.

Parameters

<code>in</code>	<code>_normal</code>	The plane normal
<code>in</code>	<code>_size</code>	Size of the plane
<code>in</code>	<code>_offset</code>	Offset along the normal

10.102.4 Member Data Documentation

10.102.4.1 `double gazebo::math::Plane::d`

Plane (p. 547) offset.

10.102.4.2 `Vector3 gazebo::math::Plane::normal`

Plane (p. 547) normal.

10.102.4.3 `Vector2d gazebo::math::Plane::size`

Plane (p. 547) size.

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

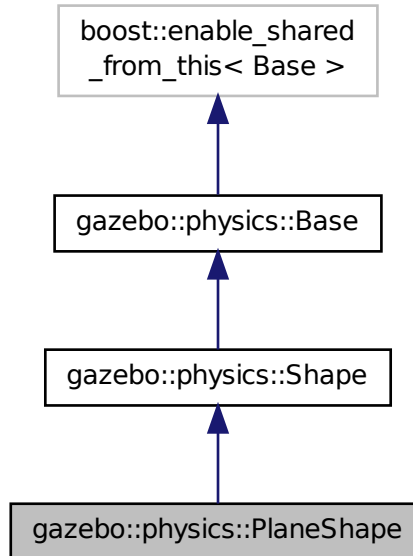
- **Plane.hh**

10.103 gazebo::physics::PlaneShape Class Reference

Collision (p. 180) for an infinite plane.

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

Inheritance diagram for gazebo::physics::PlaneShape:



Public Member Functions

- **PlaneShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~PlaneShape** ()
Destructor.
- virtual void **CreatePlane** ()
Create the plane.
- void **FillMsg** (msgs::Geometry &_msg)
Fill a geometry message with data from this object.
- **math::Vector3 GetNormal** () const
Get the plane normal.
- **math::Vector2d GetSize** () const
Get the size.
- virtual void **Init** ()
Initialize the plane.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
Process a geometry message and use the data to update this object.
- virtual void **SetAltitude** (const **math::Vector3** &_pos)
Set the altitude of the plane.
- void **SetNormal** (const **math::Vector3** &_norm)
Set the normal.

- void **SetSize** (const **math::Vector2d** &_size)

Set the size.

Additional Inherited Members

10.103.1 Detailed Description

Collision (p. 180) 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.

10.103.2 Constructor & Destructor Documentation

10.103.2.1 `gazebo::physics::PlaneShape::PlaneShape (CollisionPtr _parent)` [explicit]

Constructor.

Parameters

in	<code>_parent</code>	Link (p. 398) to which we are attached.
----	----------------------	--

10.103.2.2 `virtual gazebo::physics::PlaneShape::~~PlaneShape ()` [virtual]

Destructor.

10.103.3 Member Function Documentation

10.103.3.1 `virtual void gazebo::physics::PlaneShape::CreatePlane ()` [virtual]

Create the plane.

10.103.3.2 `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. 669).

10.103.3.3 `math::Vector3 gazebo::physics::PlaneShape::GetNormal ()` const

Get the plane normal.

Returns

The plane normal.

10.103.3.4 `math::Vector2d gazebo::physics::PlaneShape::GetSize () const`

Get the size.

Returns

Size of the plane.

10.103.3.5 `virtual void gazebo::physics::PlaneShape::Init () [virtual]`

Initialize the plane.

Implements `gazebo::physics::Shape` (p. 670).

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

Process a geometry message and use the data to update this object.

Parameters

in	_msg	Message to update from.
----	------	-------------------------

Implements `gazebo::physics::Shape` (p. 670).

10.103.3.7 `virtual void gazebo::physics::PlaneShape::SetAltitude (const math::Vector3 & _pos) [virtual]`

Set the altitude of the plane.

Parameters

in	_pos	Position of the plane.
----	------	------------------------

10.103.3.8 `void gazebo::physics::PlaneShape::SetNormal (const math::Vector3 & _norm)`

Set the normal.

Parameters

in	_norm	Plane normal.
----	-------	---------------

10.103.3.9 `void gazebo::physics::PlaneShape::SetSize (const math::Vector2d & _size)`

Set the size.

Parameters

in	_size	2D size of the plane.
----	-------	-----------------------

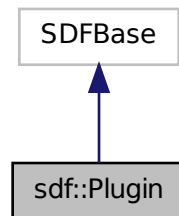
The documentation for this class was generated from the following file:

- [PlaneShape.hh](#)

10.104 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**

10.104.1 Constructor & Destructor Documentation

10.104.1.1 `sdf::Plugin::Plugin ()` [inline]

10.104.2 Member Function Documentation

10.104.2.1 `void sdf::Plugin::Clear ()` [inline]

References data.

10.104.2.2 `void sdf::Plugin::Print (const std::string & prefix)` [inline]

References filename, and name.

10.104.3 Member Data Documentation

10.104.3.1 `std::vector<ParamT<std::string>>` `sdf::Plugin::data`

Referenced by `Clear()`.

10.104.3.2 `ParamT<std::string>` `sdf::Plugin::filename`

Referenced by `Print()`.

10.104.3.3 `ParamT<std::string>` `sdf::Plugin::name`

Referenced by `Print()`.

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

- `sdf/interface/Plugin.hh`

10.105 `gazebo::PluginT< T >` Class Template Reference

A 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.

10.105.1 Detailed Description

```
template<class T>class gazebo::PluginT< T >
```

A class which all plugins must inherit from.

10.105.2 Member Typedef Documentation

10.105.2.1 `template<class T> typedef T* gazebo::PluginT< T >::TPtr`

plugin pointer type definition

10.105.3 Member Function Documentation

10.105.3.1 `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

<code>in</code>	<code>_filename</code>	the path to the shared library.
<code>in</code>	<code>_handle</code>	short name of the handler

Returns

Shared Pointer to this class type

10.105.3.2 `template<class T> std::string gazebo::PluginT< T >::GetFilename () const [inline]`

Get the name of the handler.

10.105.3.3 `template<class T> std::string gazebo::PluginT< T >::GetHandle () const [inline]`

Get the short name of the handler.

10.105.3.4 `template<class T> PluginType gazebo::PluginT< T >::GetType () const` `[inline]`

Returns the type of the plugin.

Returns

type of the plugin

10.105.4 Member Data Documentation

10.105.4.1 `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()`.

10.105.4.2 `template<class T> std::string gazebo::PluginT< T >::handle` `[protected]`

Short name.

Referenced by `gazebo::PluginT< ModelPlugin >::Create()`, and `gazebo::PluginT< ModelPlugin >::GetHandle()`.

10.105.4.3 `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`

10.106 gazebo::math::Pose Class Reference

Encapsulates a position and rotation in three space.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Pose** ()
Default constructors.
- **Pose** (const **Vector3** &_pos, const **Quaternion** &_rot)
Constructor.
- **Pose** (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)
Constructor.
- **Pose** (const **Pose** &_pose)
Copy constructor.
- virtual **~Pose** ()
Destructor.

- **Pose CoordPoseSolve** (const **Pose** &_b) const
Find the inverse of a pose; i.e., if $b = this + a$, given b and $this$, find a .
- **Vector3 CoordPositionAdd** (const **Vector3** &_pos) const
Add one point to a vector: $result = this + pos$.
- **Vector3 CoordPositionAdd** (const **Pose** &_pose) const
Add one point to another: $result = this + pose$.
- **Vector3 CoordPositionSub** (const **Pose** &_pose) const
Subtract one position from another: $result = this - pose$.
- **Quaternion CoordRotationAdd** (const **Quaternion** &_rot) const
Add one rotation to another: $result = this \rightarrow rot + rot$.
- **Quaternion CoordRotationSub** (const **Quaternion** &_rot) const
Subtract one rotation from another: $result = this \rightarrow rot - rot$.
- 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 **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. 799) and a **Quaternion** (p. 581).*
- 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.

10.106.1 Detailed Description

Encapsulates a position and rotation in three space.

10.106.2 Constructor & Destructor Documentation

10.106.2.1 gazebo::math::Pose::Pose ()

Default constructors.

Referenced by operator-().

10.106.2.2 gazebo::math::Pose::Pose (const Vector3 & _pos, const Quaternion & _rot)

Constructor.

Parameters

in	<i>_pos</i>	A position
in	<i>_rot</i>	A rotation

10.106.2.3 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.

10.106.2.4 gazebo::math::Pose::Pose (const Pose & *_pose*)

Copy constructor.

Parameters

in	<i>_pose</i>	The Pose (p. 556) to copy
----	--------------	----------------------------------

10.106.2.5 virtual gazebo::math::Pose::~~Pose () [virtual]

Destructor.

10.106.3 Member Function Documentation

10.106.3.1 Pose gazebo::math::Pose::CoordPoseSolve (const Pose & *_b*) const

Find the inverse of a pose; i.e., if $b = \text{this} + a$, given b and this , find a .

Parameters

in	<i>_b</i>	the other pose
----	-----------	----------------

10.106.3.2 Vector3 gazebo::math::Pose::CoordPositionAdd (const Vector3 & *_pos*) const

Add one point to a vector: $\text{result} = \text{this} + \text{pos}$.

Parameters

in	<i>_pos</i>	Position to add to this pose
----	-------------	------------------------------

Returns

the resulting position

10.106.3.3 Vector3 gazebo::math::Pose::CoordPositionAdd (const Pose & *_pose*) const

Add one point to another: $\text{result} = \text{this} + \text{pose}$.

Parameters

in	<i>_pose</i>	The Pose (p. 556) to add
----	--------------	---------------------------------

Returns

The resulting position

10.106.3.4 Vector3 gazebo::math::Pose::CoordPositionSub (const Pose & *_pose*) const [inline]

Subtract one position from another: $\text{result} = \text{this} - \text{pose}$.

Parameters

in	<code>_pose</code>	Pose (p. 556) 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-().

10.106.3.5 Quaternion gazebo::math::Pose::CoordRotationAdd (const Quaternion & _rot) const

Add one rotation to another: result = this->rot + rot.

Parameters

in	<code>_rot</code>	Rotation to add
----	-------------------	-----------------

Returns

The resulting rotation

10.106.3.6 Quaternion gazebo::math::Pose::CoordRotationSub (const Quaternion & _rot) const [inline]

Subtract one rotation from another: result = this->rot - rot.

Parameters

in	<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-().

10.106.3.7 void gazebo::math::Pose::Correct () [inline]

Fix any nan values.

References gazebo::math::Vector3::Correct(), gazebo::math::Quaternion::Correct(), pos, and rot.

10.106.3.8 Pose gazebo::math::Pose::GetInverse () const

Get the inverse of this pose.

Returns

the inverse pose

10.106.3.9 `bool gazebo::math::Pose::IsFinite () const`

See if a pose is finite (e.g., not nan)

10.106.3.10 `bool gazebo::math::Pose::operator!= (const Pose & _pose) const`

Inequality operator.

Parameters

<code>in</code>	<code><i>_pose</i></code>	Pose (p. 556) for comparison
-----------------	---------------------------	-------------------------------------

Returns

True if not equal

10.106.3.11 `Pose gazebo::math::Pose::operator* (const Pose & _pose)`

Multiplication operator.

Parameters

<code>in</code>	<code><i>_pose</i></code>	the other pose
-----------------	---------------------------	----------------

Returns

itself

10.106.3.12 `Pose gazebo::math::Pose::operator+ (const Pose & _pose) const`

Addition operator.

Parameters

<code>in</code>	<code><i>_pose</i></code>	Pose (p. 556) to add to this pose
-----------------	---------------------------	--

Returns

The resulting pose

10.106.3.13 `const Pose& gazebo::math::Pose::operator+=(const Pose & _pose)`

Add-Equals operator.

Parameters

in	<i>_pose</i>	Pose (p. 556) to add to this pose
----	--------------	--

Returns

The resulting pose

10.106.3.14 **Pose** gazebo::math::Pose::operator- (const **Pose** & *_pose*) const [inline]

Subtraction operator.

Parameters

in	<i>_pose</i>	Pose (p. 556) to subtract from this one
----	--------------	--

Returns

The resulting pose

References CoordPositionSub(), CoordRotationSub(), Pose(), and rot.

10.106.3.15 const **Pose**& gazebo::math::Pose::operator-= (const **Pose** & *_pose*)

Subtraction operator.

Parameters

in	<i>_pose</i>	Pose (p. 556) to subtract from this one
----	--------------	--

Returns

The resulting pose

10.106.3.16 bool gazebo::math::Pose::operator==(const **Pose** & *_pose*) const

Equality operator.

Parameters

in	<i>_pose</i>	Pose (p. 556) for comparison
----	--------------	-------------------------------------

Returns

True if equal

10.106.3.17 void gazebo::math::Pose::Reset ()

Reset the pose.

10.106.3.18 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

10.106.3.19 void gazebo::math::Pose::Round (int _precision)

Round all values to *_precision* decimal places.

Parameters

in	<i>_precision</i>	
----	-------------------	--

10.106.3.20 void gazebo::math::Pose::Set (const Vector3 & _pos, const Quaternion & _rot)

Set the pose from a **Vector3** (p. 799) and a **Quaternion** (p. 581).

Parameters

in	<i>_pos</i>	The position.
in	<i>_rot</i>	The rotation.

10.106.3.21 void gazebo::math::Pose::Set (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)

Set the pose from a six tuple.

Parameters

in	<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>	Pitch (rotation about z-axis) in radians.

10.106.4 Friends And Related Function Documentation

10.106.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Pose & _pose) [friend]

Stream insertion operator.

Parameters

<code>in</code>	<code>_out</code>	output stream
<code>in</code>	<code>_pose</code>	pose to output

Returns

the stream

10.106.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Pose & _pose)` [`friend`]

Stream extraction operator.

Parameters

<code>in</code>	<code>_in</code>	the input stream
<code>in</code>	<code>_pose</code>	the pose

Returns

the stream

10.106.5 Member Data Documentation

10.106.5.1 `Vector3 gazebo::math::Pose::pos`

The position.

Referenced by `CoordPositionSub()`, and `Correct()`.

10.106.5.2 `Quaternion gazebo::math::Pose::rot`

The rotation.

Referenced by `CoordPositionSub()`, `CoordRotationSub()`, `Correct()`, and `operator-`.

10.106.5.3 `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:

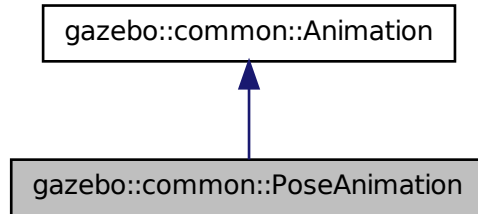
- **Pose.hh**

10.107 `gazebo::common::PoseAnimation` Class Reference

A pose animation.

```
#include <Animation.hh>
```


Inheritance diagram for gazebo::common::PoseAnimation:



Public Member Functions

- **PoseAnimation** (const std::string &_name, double _length, bool _loop)
Constructor.
- virtual **~PoseAnimation** ()
Destructor.
- **PoseKeyFrame * CreateKeyFrame** (double _time)
Create a pose keyframe at the given time.
- void **GetInterpolatedKeyFrame** (**PoseKeyFrame** &_kf) const
Get a keyframe using the animation's current time.

Protected Member Functions

- void **BuildInterpolationSplines** () const
Update the pose splines.
- void **GetInterpolatedKeyFrame** (double _time, **PoseKeyFrame** &_kf) const
Get a keyframe using a passed in time.

Additional Inherited Members

10.107.1 Detailed Description

A pose animation.

10.107.2 Constructor & Destructor Documentation

10.107.2.1 gazebo::common::PoseAnimation::PoseAnimation (const std::string & _name, double _length, bool _loop)

Constructor.

Parameters

in	<code>_name</code>	String name of the animation. This should be unique.
in	<code>_length</code>	Length of the animation in seconds
in	<code>_loop</code>	True == loop the animation

10.107.2.2 `virtual gazebo::common::PoseAnimation::~~PoseAnimation () [virtual]`

Destructor.

10.107.3 Member Function Documentation

10.107.3.1 `void gazebo::common::PoseAnimation::BuildInterpolationSplines () const [protected]`

Update the pose splines.

10.107.3.2 `PoseKeyFrame* gazebo::common::PoseAnimation::CreateKeyFrame (double _time)`

Create a pose keyframe at the given time.

Parameters

in	<code>_time</code>	Time (p. 732) at which to create the keyframe
----	--------------------	--

Returns

Pointer to the new keyframe

10.107.3.3 `void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (PoseKeyFrame & _kf) const`

Get a keyframe using the animation's current time.

Parameters

out	<code>_kf</code>	PoseKeyFrame (p. 567) reference to hold the interpolated result
-----	------------------	--

10.107.3.4 `void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (double _time, PoseKeyFrame & _kf) const [protected]`

Get a keyframe using a passed in time.

Parameters

in	<code>_time</code>	Time (p. 732) in seconds
out	<code>_kf</code>	PoseKeyFrame (p. 567) reference to hold the interpolated result

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

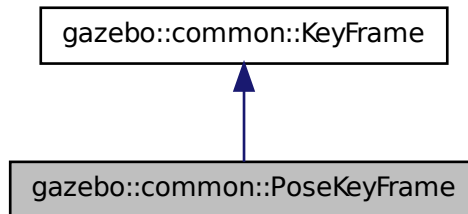
- **Animation.hh**

10.108 gazebo::common::PoseKeyFrame Class Reference

A keyframe for a **PoseAnimation** (p. 564).

```
#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

10.108.1 Detailed Description

A keyframe for a **PoseAnimation** (p. 564).

10.108.2 Constructor & Destructor Documentation

10.108.2.1 gazebo::common::PoseKeyFrame::PoseKeyFrame (double *_time*)

Constructor.

Parameters

in	<i>_time</i>	of the keyframe
----	--------------	-----------------

10.108.2.2 virtual gazebo::common::PoseKeyFrame::~~PoseKeyFrame () [virtual]

Destructor.

10.108.3 Member Function Documentation

10.108.3.1 const math::Quaternion& gazebo::common::PoseKeyFrame::GetRotation () const

Get the rotation of the keyframe.

Returns

The rotation amount

10.108.3.2 const math::Vector3& gazebo::common::PoseKeyFrame::GetTranslation () const

Get the translation of the keyframe.

Returns

The translation amount

10.108.3.3 void gazebo::common::PoseKeyFrame::SetRotation (const math::Quaternion & *_rot*)

Set the rotation for the keyframe.

Parameters

in	<i>_rot</i>	Rotation amount
----	-------------	-----------------

10.108.3.4 void gazebo::common::PoseKeyFrame::SetTranslation (const math::Vector3 & *_trans*)

Set the translation for the keyframe.

Parameters

in	<i>_trans</i>	Translation amount
----	---------------	--------------------

10.108.4 Member Data Documentation

10.108.4.1 `math::Quaternion` `gazebo::common::PoseKeyFrame::rotate` [protected]

the rotation quaternion

10.108.4.2 `math::Vector3` `gazebo::common::PoseKeyFrame::translate` [protected]

the translation vector

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

- **KeyFrame.hh**

10.109 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.

10.109.1 Detailed Description

Projects a material onto surface, light a light projector.

10.109.2 Constructor & Destructor Documentation

10.109.2.1 gazebo::rendering::Projector::Projector (VisualPtr *_parent*)

Constructor.

Parameters

in	<i>_parent</i>	Name of the parent visual.
----	----------------	----------------------------

10.109.2.2 virtual gazebo::rendering::Projector::~~Projector () [virtual]

Destructor.

10.109.3 Member Function Documentation

10.109.3.1 VisualPtr gazebo::rendering::Projector::GetParent ()

Get the parent visual.

Returns

Pointer of the parent visual.

10.109.3.2 void gazebo::rendering::Projector::Load (sdf::ElementPtr *_sdf*)

Load from an sdf pointer.

Parameters

in	<i>_sdf</i>	Pointer to the SDF element.
----	-------------	-----------------------------

10.109.3.3 void gazebo::rendering::Projector::Load (const msgs::Projector & *_msg*)

Load from a message.

Parameters

in	<i>_msg</i>	Load from a message.
----	-------------	----------------------

10.109.3.4 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.

10.109.3.5 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.
----	-----------------------	-------------------------------

10.109.3.6 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.
----	---------------------------	---------------------------------

10.109.3.7 void gazebo::rendering::Projector::Toggle ()

Toggle the activation of the projector.

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

- **Projector.hh**

10.110 gazebo::transport::Publication Class Reference

A publication for a topic.

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

Public Member Functions

- **Publication** (const std::string & *_topic*, const std::string & *_msgType*)
Constructor.
- virtual **~Publication** ()
Destructor.
- void **AddPublisher** (**PublisherPtr** *_pub*)
Add a publisher.
- void **AddSubscription** (const **CallbackHelperPtr** & *_callback*)

- Subscribe a callback to our topic.*

 - void **AddSubscription** (const **NodePtr** &_node)
- Subscribe a node to our topic.*

 - void **AddTransport** (const **PublicationTransportPtr** &_publink)
- Add a transport.*

 - unsigned int **GetCallbackCount** () const
- Get the number of callbacks.*

 - bool **GetLocallyAdvertised** () const
- Was the topic has been advertised from this process?*

 - std::string **GetMsgType** () const
- Get the type of message.*

 - unsigned int **GetNodeCount** () const
- Get the number of nodes.*

 - unsigned int **GetRemoteSubscriptionCount** ()
- Get the number of remote subscriptions.*

 - unsigned int **GetTransportCount** () const
- Get the number of transports.*

 - bool **HasTransport** (const std::string &_host, unsigned int _port)
- Does a given transport exist?*

 - void **LocalPublish** (const std::string &_data)
- Publish data to local subscribers (skip serialization)*

 - void **Publish** (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.*

10.110.1 Detailed Description

A publication for a topic.

This facilitates transport of messages

10.110.2 Constructor & Destructor Documentation

10.110.2.1 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

10.110.2.2 virtual gazebo::transport::Publication::~~Publication () [virtual]

Destructor.

10.110.3 Member Function Documentation

10.110.3.1 void gazebo::transport::Publication::AddPublisher (PublisherPtr *_pub*)

Add a publisher.

Parameters

in, out	<i>_pub</i>	Pointer to publisher object to be added
---------	-------------	---

Referenced by gazebo::transport::TopicManager::Advertise().

10.110.3.2 void gazebo::transport::Publication::AddSubscription (const CallbackHelperPtr & *_callback*)

Subscribe a callback to our topic.

Parameters

in	<i>_callback</i>	The callback
----	------------------	--------------

Referenced by gazebo::transport::TopicManager::Advertise().

10.110.3.3 void gazebo::transport::Publication::AddSubscription (const NodePtr & *_node*)

Subscribe a node to our topic.

Parameters

in	<i>_node</i>	The node
----	--------------	----------

10.110.3.4 void gazebo::transport::Publication::AddTransport (const PublicationTransportPtr & *_publink*)

Add a transport.

Parameters

in	<i>_publink</i>	Pointer to publication transport object to be added
----	-----------------	---

10.110.3.5 unsigned int gazebo::transport::Publication::GetCallbackCount () const

Get the number of callbacks.

Returns

The number of callbacks

10.110.3.6 `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()`.

10.110.3.7 `std::string gazebo::transport::Publication::GetMsgType () const`

Get the type of message.

Returns

The type of message

10.110.3.8 `unsigned int gazebo::transport::Publication::GetNodeCount () const`

Get the number of nodes.

Returns

The number of nodes

10.110.3.9 `unsigned int gazebo::transport::Publication::GetRemoteSubscriptionCount ()`

Get the number of remote subscriptions.

Returns

The number of remote subscriptions

10.110.3.10 `unsigned int gazebo::transport::Publication::GetTransportCount () const`

Get the number of transports.

Returns

The number of transports

10.110.3.11 `bool gazebo::transport::Publication::HasTransport (const std::string & _host, unsigned int _port)`

Does a given transport exist?

Parameters

<code>in</code>	<code>_host</code>	Hostname of the transport
<code>in</code>	<code>_port</code>	Port of the transport

Returns

true if the transport exists, false otherwise

10.110.3.12 `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
-----------	--------------	--------------------------

10.110.3.13 `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

10.110.3.14 `void gazebo::transport::Publication::RemoveSubscription (const NodePtr & _node)`

Unsubscribe a node from our topic.

Parameters

<i>in</i>	<i>_node</i>	The node
-----------	--------------	----------

10.110.3.15 `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

10.110.3.16 `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

10.110.3.17 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**

10.111 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*)
Initialize the transport.

10.111.1 Detailed Description

transport/transport.hh

Reads data from a remote advertiser, and passes the data along to local subscribers

10.111.2 Constructor & Destructor Documentation

10.111.2.1 `gazebo::transport::PublicationTransport::PublicationTransport (const std::string & _topic, const std::string & _msgType)`

Constructor.

Parameters

<code>in</code>	<code><i>_topic</i></code>	Topic that we're publishing
<code>in</code>	<code><i>_msgType</i></code>	Type of the topic that we're publishing

10.111.2.2 `virtual gazebo::transport::PublicationTransport::~~PublicationTransport () [virtual]`

Destructor.

10.111.3 Member Function Documentation

10.111.3.1 `void gazebo::transport::PublicationTransport::AddCallback (const boost::function< void(const std::string &)> & _cb)`

Add a callback to the transport.

Parameters

<code>in</code>	<code><i>_cb</i></code>	The callback to be added
-----------------	-------------------------	--------------------------

10.111.3.2 `void gazebo::transport::PublicationTransport::Fini ()`

Finalize the transport.

10.111.3.3 `const ConnectionPtr gazebo::transport::PublicationTransport::GetConnection () const`

Get the underlying connection.

Returns

Pointer to the underlying connection

10.111.3.4 `std::string gazebo::transport::PublicationTransport::GetMsgType () const`

Get the topic type.

Returns

The topic type

10.111.3.5 `std::string gazebo::transport::PublicationTransport::GetTopic () const`

Get the topic name.

Returns

The topic name

10.111.3.6 `void gazebo::transport::PublicationTransport::Init (const ConnectionPtr & _conn)`

Initialize the transport.

Parameters

in	_conn	The underlying connection
----	-------	---------------------------

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

- **PublicationTransport.hh**

10.112 gazebo::transport::Publisher Class Reference

A publisher of messages on a topic.

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

Public Member Functions

- **Publisher** (const std::string &_topic, const std::string &_msgType, unsigned int _limit, bool _latch)
Constructor.
- virtual **~Publisher** ()
Destructor.
- bool **GetLatching** () const
Are we latching the latest message?
- 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.

10.112.1 Detailed Description

A publisher of messages on a topic.

10.112.2 Constructor & Destructor Documentation

10.112.2.1 `gazebo::transport::Publisher::Publisher (const std::string & _topic, const std::string & _msgType, unsigned int _limit, bool _latch)`

Constructor.

Parameters

in	<code>_topic</code>	Name of topic to be published
in	<code>_msgType</code>	Type of the message to be published
in	<code>_limit</code>	Maximum number of outgoing messages to queue
in	<code>_latch</code>	If true, latch last message; if false, don't latch

10.112.2.2 `virtual gazebo::transport::Publisher::~~Publisher () [virtual]`

Destructor.

10.112.3 Member Function Documentation

10.112.3.1 `bool gazebo::transport::Publisher::GetLatching () const`

Are we latching the latest message?

Returns

true if we latching the latest message, false otherwise

10.112.3.2 `std::string gazebo::transport::Publisher::GetMsgType () const`

Get the message type.

Returns

The message type

10.112.3.3 `unsigned int gazebo::transport::Publisher::GetOutgoingCount () const`

Get the number of outgoing messages.

Returns

The number of outgoing messages

10.112.3.4 `std::string gazebo::transport::Publisher::GetPrevMsg () const`

Get the previously published message.

Returns

The previously published message, if any

10.112.3.5 `std::string gazebo::transport::Publisher::GetTopic () const`

Get the topic name.

Returns

The topic name

10.112.3.6 `bool gazebo::transport::Publisher::HasConnections () const`

Are there any connections?

Returns

true if there are any connections, false otherwise

10.112.3.7 `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>_message</code>	Message to be published
<code>in</code>	<code>_block</code>	Whether to block until the message is actually written out

10.112.3.8 `template<typename M > void gazebo::transport::Publisher::Publish (M _message, bool _block = false)`
`[inline]`

Publish an arbitrary message on the topic.

Parameters

in	<code>_message</code>	Message to be published
in	<code>_block</code>	Whether to block until the message is actually written out

10.112.3.9 void gazebo::transport::Publisher::SendMessage ()

Send latest message over the wire. For internal use only.

10.112.3.10 void gazebo::transport::Publisher::SetPublication (PublicationPtr & _publication, int _i)

Set the publication object for a particular publication.

Parameters

in	<code>_publication</code>	Pointer to the publication object to be set
in	<code>_i</code>	Index into publications vector that will be set

10.112.3.11 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**

10.113 gazebo::math::Quaternion Class Reference

A quaternion class.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Quaternion** ()
Default Constructor.
- **Quaternion** (const double &_w, const double &_x, const double &_y, const double &_z)
Constructor.
- **Quaternion** (const double &_roll, const double &_pitch, const double &_yaw)
Constructor from Euler angles in radians.
- **Quaternion** (const **Vector3** &_axis, const double &_angle)
Constructor from axis angle.
- **Quaternion** (const **Vector3** &_rpy)
Constructor.
- **Quaternion** (const **Quaternion** &_qt)
Copy constructor.
- **~Quaternion** ()
Destructor.

- void **Correct** ()
Correct any nan.
- double **Dot** (const **Quaternion** &_q) const
Dot product.
- void **GetAsAxis** (**Vector3** &_axis, double &_angle) const
Return rotation as axis and angle.
- **Vector3 GetAsEuler** () const
Return the rotation in Euler angles.
- **Matrix3 GetAsMatrix3** () const
Get the quaternion as a 3x3 matrix.
- **Matrix4 GetAsMatrix4** () const
Get the quaternion as a 4x4 matrix.
- **Quaternion GetExp** () const
Return the exponent.
- **Quaternion GetInverse** () const
Get the inverse of this quaternion.
- **Quaternion GetLog** () const
Return the logarithm.
- double **GetPitch** ()
Get the Euler pitch angle in radians.
- double **GetRoll** ()
Get the Euler roll angle in radians.
- **Vector3 GetXAxis** () const
Return the X axis.
- double **GetYaw** ()
Get the Euler yaw angle in radians.
- **Vector3 GetYAxis** () const
Return the Y axis.
- **Vector3 GetZAxis** () const
Return the Z axis.
- void **Invert** ()
Invert the quaternion.
- bool **IsFinite** () const
See if a quatern is finite (e.g., not nan)
- void **Normalize** ()
Normalize the quaternion.
- bool **operator!=** (const **Quaternion** &_qt) const
Not equal to operator.
- **Quaternion operator*** (const **Quaternion** &_q) const
Multiplication operator.
- **Quaternion operator*** (const double &_f) const
Multiplication operator.
- **Vector3 operator*** (const **Vector3** &_v) const
***Vector3** (p. 799) 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

Subtraction operator.

- **Quaternion operator-** () const

Unary minus operator.

- **Quaternion operator-=** (const **Quaternion** &_qt)

Subtraction 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 **SetToIdentity** ()

Set the quatern to the identity.

Static Public Member Functions

- static **Quaternion EulerToQuaternion** (const **Vector3** &_vec)

Convert euler angles to quatern.

- static **Quaternion EulerToQuaternion** (double _x, double _y, double _z)

Convert euler angles to quatern.

- static **Quaternion Slerp** (double _fT, const **Quaternion** &_rkP, const **Quaternion** &_rkQ, bool _shortestPath=false)

Spherical linear interpolation between 2 quaternions, given the ends and an interpolation parameter between 0 and 1.

- static **Quaternion Squad** (double _fT, const **Quaternion** &_rkP, const **Quaternion** &_rkA, const **Quaternion** &_rkB, const **Quaternion** &_rkQ, bool _shortestPath=false)

Spherical quadratic interpolation given the ends and an interpolation parameter between 0 and 1.

Public Attributes

- double **w**
Attributes of the quaternion.
- double **x**
Attributes of the quaternion.
- double **y**
Attributes of the quaternion.
- double **z**
Attributes of the quaternion.

Friends

- `std::ostream & operator<< (std::ostream &_out, const gazebo::math::Quaternion &_q)`
Stream insertion operator.
- `std::istream & operator>> (std::istream &_in, gazebo::math::Quaternion &_q)`
Stream extraction operator.

10.113.1 Detailed Description

A quaternion class.

10.113.2 Constructor & Destructor Documentation

10.113.2.1 gazebo::math::Quaternion::Quaternion ()

Default Constructor.

Referenced by operator*().

10.113.2.2 gazebo::math::Quaternion::Quaternion (const double &_w, const double &_x, const double &_y, const double &_z)

Constructor.

Parameters

<code>in</code>	<code>_w</code>	W param
<code>in</code>	<code>_x</code>	X param
<code>in</code>	<code>_y</code>	Y param
<code>in</code>	<code>_z</code>	Z param

10.113.2.3 gazebo::math::Quaternion::Quaternion (const double &_roll, const double &_pitch, const double &_yaw)

Constructor from Euler angles in radians.

Parameters

<code>in</code>	<code>_roll</code>	roll
<code>in</code>	<code>_pitch</code>	pitch
<code>in</code>	<code>_yaw</code>	yaw

10.113.2.4 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

10.113.2.5 gazebo::math::Quaternion::Quaternion (const Vector3 & *_rpy*)

Constructor.

Parameters

in	<i>_rpy</i>	euler angles
----	-------------	--------------

10.113.2.6 gazebo::math::Quaternion::Quaternion (const Quaternion & *_qt*)

Copy constructor.

Parameters

<i>qt</i>	Quaternion (p. 581) to copy
-----------	------------------------------------

10.113.2.7 gazebo::math::Quaternion::~~Quaternion ()

Destructor.

10.113.3 Member Function Documentation

10.113.3.1 void gazebo::math::Quaternion::Correct () [inline]

Correct any nan.

References gazebo::math::equal(), w, x, y, and z.

Referenced by gazebo::math::Pose::Correct().

10.113.3.2 double gazebo::math::Quaternion::Dot (const Quaternion & *_q*) const

Dot product.

Parameters

in	<i>_q</i>	the other quaternion
----	-----------	----------------------

Returns

the product

10.113.3.3 `static Quaternion gazebo::math::Quaternion::EulerToQuaternion (const Vector3 & _vec) [static]`

Convert euler angles to quaternion.

Parameters

<code>in</code>		
-----------------	--	--

10.113.3.4 `static Quaternion gazebo::math::Quaternion::EulerToQuaternion (double _x, double _y, double _z) [static]`

Convert euler angles to quaternion.

Parameters

<code>in</code>	<code>_x</code>	rotation along x
<code>in</code>	<code>_y</code>	rotation along y
<code>in</code>	<code>_z</code>	rotation along z

10.113.3.5 `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

10.113.3.6 `Vector3 gazebo::math::Quaternion::GetAsEuler () const`

Return the rotation in Euler angles.

Returns

This quaternion as an Euler vector

10.113.3.7 `Matrix3 gazebo::math::Quaternion::GetAsMatrix3 () const`

Get the quaternion as a 3x3 matrix.

10.113.3.8 `Matrix4 gazebo::math::Quaternion::GetAsMatrix4 () const`

Get the quaternion as a 4x4 matrix.

Returns

a 4x4 matrix

10.113.3.9 Quaternion gazebo::math::Quaternion::GetExp () const

Return the exponent.

Returns

the exp

10.113.3.10 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().

10.113.3.11 Quaternion gazebo::math::Quaternion::GetLog () const

Return the logarithm.

Returns

the log

10.113.3.12 double gazebo::math::Quaternion::GetPitch ()

Get the Euler pitch angle in radians.

Returns

the pitch

10.113.3.13 double gazebo::math::Quaternion::GetRoll ()

Get the Euler roll angle in radians.

Returns

the roll

10.113.3.14 Vector3 gazebo::math::Quaternion::GetXAxis () const

Return the X axis.

Returns

the vector

10.113.3.15 `double gazebo::math::Quaternion::GetYaw ()`

Get the Euler yaw angle in radians.

Returns

the yaw

10.113.3.16 `Vector3 gazebo::math::Quaternion::GetYAxis () const`

Return the Y axis.

Returns

the vector

10.113.3.17 `Vector3 gazebo::math::Quaternion::GetZAxis () const`

Return the Z axis.

Returns

the vector

10.113.3.18 `void gazebo::math::Quaternion::Invert ()`

Invert the quaternion.

10.113.3.19 `bool gazebo::math::Quaternion::IsFinite () const`

See if a quatern is finite (e.g., not nan)

Returns

True if quatern is finite

10.113.3.20 `void gazebo::math::Quaternion::Normalize ()`

Normalize the quaternion.

Referenced by `gazebo::math::Pose::CoordRotationSub()`.

10.113.3.21 `bool gazebo::math::Quaternion::operator!=(const Quaternion & _qt) const`

Not equal to operator.

Parameters

<code>in</code>	<code>_qt</code> Quaternion (p. 581) for comparison
-----------------	--

Returns

True if not equal

10.113.3.22 **Quaternion** gazebo::math::Quaternion::operator* (const Quaternion & _q) const [inline]

Multiplication operator.

Parameters

in	_qt	Quaternion (p. 581) for multiplication
----	-----	---

Returns

This quaternion multiplied by the parameter

References Quaternion(), w, x, y, and z.

10.113.3.23 **Quaternion** gazebo::math::Quaternion::operator* (const double & _f) const

Multiplication operator.

Parameters

in	_f	factor
----	----	--------

Returns

quaternion multiplied by _f

10.113.3.24 **Vector3** gazebo::math::Quaternion::operator* (const Vector3 & _v) const

Vector3 (p. 799) multiplication operator.

Parameters

in	_v	vector to multiply
----	----	--------------------

10.113.3.25 **Quaternion** gazebo::math::Quaternion::operator*= (const Quaternion & qt)

Multiplication operator.

Parameters

in	_qt	Quaternion (p. 581) for multiplication
----	-----	---

Returns

This quatern multiplied by the parameter

10.113.3.26 **Quaternion gazebo::math::Quaternion::operator+ (const Quaternion & _qt) const**

Addition operator.

Parameters

in	_qt	quaternion for addition
----	-----	-------------------------

Returns

this quaternion + _qt

10.113.3.27 **Quaternion gazebo::math::Quaternion::operator+= (const Quaternion & _qt)**

Addition operator.

Parameters

in	_qt	quaternion for addition
----	-----	-------------------------

Returns

this quaternion + qt

10.113.3.28 **Quaternion gazebo::math::Quaternion::operator- (const Quaternion & _qt) const**

Substraction operator.

Parameters

in	_qt	quaternion to subtract
----	-----	------------------------

Returns

this quaternion - _qt

10.113.3.29 **Quaternion gazebo::math::Quaternion::operator- () const**

Unary minus operator.

Returns

negates each component of the quaternion

10.113.3.30 **Quaternion gazebo::math::Quaternion::operator-= (const Quaternion & _qt)**

Substraction operator.

Parameters

in	_qt	Quaternion (p. 581) for subtraction
----	-----	-------------------------------------

Returns

This quatern - qt

10.113.3.31 Quaternion& gazebo::math::Quaternion::operator= (const Quaternion & _qt)

Equal operator.

Parameters

in	_qt	Quaternion (p. 581) to copy
----	-----	-----------------------------

10.113.3.32 bool gazebo::math::Quaternion::operator== (const Quaternion & _qt) const

Equal to operator.

Parameters

in	_qt	Quaternion (p. 581) for comparison
----	-----	------------------------------------

Returns

True if equal

10.113.3.33 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, x, gazebo::math::Vector3::y, y, gazebo::math::Vector3::z, and z.

10.113.3.34 Vector3 gazebo::math::Quaternion::RotateVectorReverse (Vector3 _vec) const

Do the reverse rotation of a vector by this quaternion.

Parameters

in	_vec	the vector
----	------	------------

Returns

the

10.113.3.35 `void gazebo::math::Quaternion::Round (int _precision)`

Round all values to `_precision` decimal places.

Parameters

<code>in</code>	<code><i>_precision</i></code>	the precision
-----------------	--------------------------------	---------------

10.113.3.36 `void gazebo::math::Quaternion::Scale (double _scale)`

Scale a Quaternionion.

Parameters

<code>in</code>	<code><i>_scale</i></code>	Amount to scale this rotation
-----------------	----------------------------	-------------------------------

10.113.3.37 `void gazebo::math::Quaternion::Set (double _u, double _x, double _y, double _z)`

Set this quaternion from 4 floating numbers.

Parameters

<code>in</code>	<code><i>_u</i></code>	<code>u</code>
<code>in</code>	<code><i>_x</i></code>	<code>x</code>
<code>in</code>	<code><i>_y</i></code>	<code>y</code>
<code>in</code>	<code><i>_z</i></code>	<code>z</code>

10.113.3.38 `void gazebo::math::Quaternion::SetFromAxis (double _x, double _y, double _z, double _a)`

Set the quaternion from an axis and angle.

Parameters

<code>in</code>	<code><i>_x</i></code>	X axis
<code>in</code>	<code><i>_y</i></code>	Y axis
<code>in</code>	<code><i>_z</i></code>	Z axis
<code>in</code>	<code><i>_a</i></code>	Angle (p. 107) in radians

10.113.3.39 `void gazebo::math::Quaternion::SetFromAxis (const Vector3 & _axis, double _a)`

Set the quaternion from an axis and angle.

Parameters

in	<code>_axis</code>	Axis
in	<code>_a</code>	Angle (p. 107) in radians

10.113.3.40 `void gazebo::math::Quaternion::SetFromEuler (const Vector3 & _vec)`

Set the quaternion from Euler angles.

Parameters

in	<code>vec</code>	Euler angle
----	------------------	-------------

10.113.3.41 `void gazebo::math::Quaternion::SetToIdentity ()`

Set the quatern to the identity.

10.113.3.42 `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

10.113.3.43 `static Quaternion gazebo::math::Quaternion::Squad (double _ft, const Quaternion & _rkP, const Quaternion & _rkA, const Quaternion & _rkB, const Quaternion & _rkQ, bool _shortestPath = false) [static]`

Spherical quadratic interpolation given the ends and an interpolation parameter between 0 and 1.

Parameters

in	<code>_ft</code>	the interpolation parameter
in	<code>_rkP</code>	the beginning quaternion
in	<code>_rkA</code>	first intermediate quaternion
in	<code>_rkB</code>	second intermediate quaternion
in	<code>_rkQ</code>	the end quaternion
in	<code>_shortestPath</code>	when true, the rotation may be inverted to get to minimize rotation

10.113.4 Friends And Related Function Documentation

10.113.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Quaternion & _q) [friend]`

Stream insertion operator.

Parameters

<code>in</code>	<code>_out</code>	output stream
<code>in</code>	<code>_q</code>	quaternion to output

Returns

the stream

10.113.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Quaternion & _q) [friend]`

Stream extraction operator.

Parameters

<code>in</code>	<code>_in</code>	input stream
<code>in</code>	<code>_q</code>	Quaternion (p. 581) to read values into

Returns

The istream

10.113.5 Member Data Documentation

10.113.5.1 `double gazebo::math::Quaternion::w`

Attributes of the quaternion.

Referenced by `Correct()`, `GetInverse()`, and `operator*()`.

10.113.5.2 `double gazebo::math::Quaternion::x`

Attributes of the quaternion.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `GetInverse()`, `operator*()`, and `RotateVector()`.

10.113.5.3 `double gazebo::math::Quaternion::y`

Attributes of the quaternion.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `GetInverse()`, `operator*()`, and `RotateVector()`.

10.113.5.4 `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**

10.114 gazebo::math::Rand Class Reference

Random number generator class.

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

Static Public Member Functions

- static double **GetDbNormal** (double *_mean*=0, double *_sigma*=1)
Get a double from a normal distribution.
- static double **GetDbUniform** (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.

10.114.1 Detailed Description

Random number generator class.

10.114.2 Member Function Documentation

10.114.2.1 static double gazebo::math::Rand::GetDbNormal (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

10.114.2.2 static double gazebo::math::Rand::GetDbUniform (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

10.114.2.3 static int gazebo::math::Rand::GetIntNormal (int *_mean*, int *_sigma*) [static]

Get a double from a normal distribution.

Parameters

in	<code>_mean</code>	Mean value for the distribution
in	<code>_sigma</code>	Sigma value for the distribution

10.114.2.4 `static int gazebo::math::Rand::GetIntUniform (int _min, int _max) [static]`

Get a integer from a uniform distribution.

Parameters

in	<code>_min</code>	Minimum bound for the random number
in	<code>_max</code>	Maximum bound for the random number

10.114.2.5 `static uint32_t gazebo::math::Rand::GetSeed () [static]`

Get the seed value.

Returns

The seed value used to initialize the random number generator.

10.114.2.6 `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:

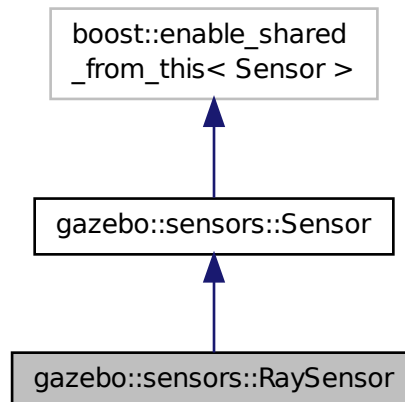
- **Rand.hh**

10.115 gazebo::sensors::RaySensor Class Reference

Sensor (p. 652) 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. 492).*
- 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 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

10.115.1 Detailed Description

Sensor (p. 652) 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).

10.115.2 Constructor & Destructor Documentation

10.115.2.1 gazebo::sensors::RaySensor::RaySensor ()

Constructor.

10.115.2.2 virtual gazebo::sensors::RaySensor::~RaySensor () [virtual]

Destructor.

10.115.3 Member Function Documentation

10.115.3.1 virtual void gazebo::sensors::RaySensor::Fini () [protected],[virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 655).

10.115.3.2 math::Angle gazebo::sensors::RaySensor::GetAngleMax () const

Get the maximum angle.

Returns

the maximum angle object

10.115.3.3 math::Angle gazebo::sensors::RaySensor::GetAngleMin () const

Get the minimum angle.

Returns

The minimum angle object

10.115.3.4 double gazebo::sensors::RaySensor::GetAngleResolution () const

Get the angle in radians between each range.

Returns

Resolution of the angle

10.115.3.5 int gazebo::sensors::RaySensor::GetFiducial (int *_index*)

Get detected fiducial value for a ray.

Warning: If you are accessing all the ray data in a loop it's possible that the Ray will update in the middle of your access loop. This means some data will come from one scan, and some from another scan. You can solve this problem by using `SetActive(false)` <your accessor loop> `SetActive(true)`.

Parameters

in	<i>_index</i>	Index value of specific ray
----	---------------	-----------------------------

Returns

Fiducial value

10.115.3.6 `physics::MultiRayShapePtr gazebo::sensors::RaySensor::GetLaserShape () const` `[inline]`

Returns a pointer to the internal `physics::MultiRayShape` (p. 492).

Returns

Pointer to ray shape

10.115.3.7 `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.

10.115.3.8 `int gazebo::sensors::RaySensor::GetRangeCount () const`

Get the range count.

Returns

The number of ranges

10.115.3.9 `double gazebo::sensors::RaySensor::GetRangeMax () const`

Get the maximum range.

Returns

The maximum range

10.115.3.10 `double gazebo::sensors::RaySensor::GetRangeMin () const`

Get the minimum range.

Returns

The minimum range

10.115.3.11 `double gazebo::sensors::RaySensor::GetRangeResolution () const`

Get the range resolution.

Returns

Resolution of the range

10.115.3.12 `void gazebo::sensors::RaySensor::GetRanges (std::vector< double > & _ranges)`

Get all the ranges.

Parameters

<code>_ranges</code>	A vector that will contain all the range data
----------------------	---

10.115.3.13 `int gazebo::sensors::RaySensor::GetRayCount () const`

Get the ray count.

Returns

The number of rays

10.115.3.14 `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

10.115.3.15 `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. 656).

10.115.3.16 **math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMax () const**

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.115.3.17 **math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMin () const**

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.115.3.18 **int gazebo::sensors::RaySensor::GetVerticalRangeCount () const**

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.115.3.19 **int gazebo::sensors::RaySensor::GetVerticalRayCount () const**

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.115.3.20 **virtual void gazebo::sensors::RaySensor::Init () [virtual]**

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 657).

10.115.3.21 **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. 658).

10.115.3.22 `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. 659).

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

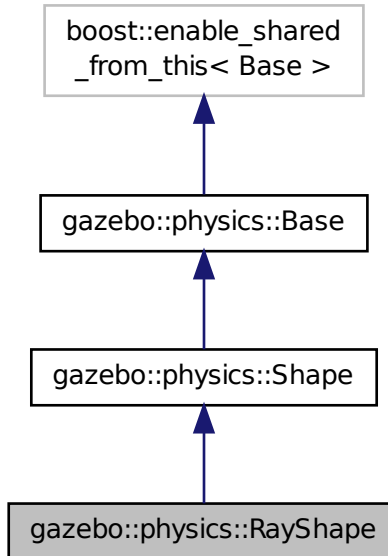
- **RaySensor.hh**

10.116 gazebo::physics::RayShape Class Reference

Base (p. 125) 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

10.116.1 Detailed Description

Base (p. 125) class for Ray collision geometry.

10.116.2 Constructor & Destructor Documentation

10.116.2.1 gazebo::physics::RayShape::RayShape (PhysicsEnginePtr _physicsEngine) [explicit]

Constructor for a global ray.

Parameters

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

10.116.2.2 `gazebo::physics::RayShape::RayShape (CollisionPtr _parent) [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Collision (p. 180) parent of the shape.
----	----------------------	--

10.116.2.3 `virtual gazebo::physics::RayShape::~~RayShape () [virtual]`

Destructor.

10.116.3 Member Function Documentation

10.116.3.1 `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. 669).

10.116.3.2 `int gazebo::physics::RayShape::GetFiducial () const`

Get the fiducial id detected by this ray.

Returns

Fiducial id detected.

10.116.3.3 `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.

10.116.3.4 `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.

10.116.3.5 `double gazebo::physics::RayShape::GetLength () const`

Get the length of the ray.

Returns

The ray length.

10.116.3.6 `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.

10.116.3.7 `float gazebo::physics::RayShape::GetRetro () const`

Get the retro-reflectivness detected by this ray.

Returns

Retro reflectance value.

10.116.3.8 `virtual void gazebo::physics::RayShape::Init () [virtual]`

In the ray.

Implements **`gazebo::physics::Shape`** (p. 670).

10.116.3.9 `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. 670).

10.116.3.10 `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.
----	-------------	-----------------------------------

10.116.3.11 `virtual void gazebo::physics::RayShape::SetLength (double _len) [virtual]`

Set the length of the ray.

Parameters

in	<i>_len</i>	Length of the array.
----	-------------	----------------------

10.116.3.12 `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.

10.116.3.13 `void gazebo::physics::RayShape::SetRetro (float _retro)`

Set the retro-reflectivness detected by this ray.

Parameters

in	<i>_retro</i>	Retro reflectance value.
----	---------------	--------------------------

10.116.3.14 `virtual void gazebo::physics::RayShape::Update () [pure virtual]`

Update the ray collision.

Reimplemented from `gazebo::physics::Base` (p. 135).

10.116.4 Member Data Documentation

10.116.4.1 `int gazebo::physics::RayShape::contactFiducial [protected]`

Fiducial ID value.

10.116.4.2 `double gazebo::physics::RayShape::contactLen` [protected]

Length of the ray.

10.116.4.3 `double gazebo::physics::RayShape::contactRetro` [protected]

Retro reflectance value.

10.116.4.4 `math::Vector3 gazebo::physics::RayShape::globalEndPos` [protected]

End position of the ray in global cs.

10.116.4.5 `math::Vector3 gazebo::physics::RayShape::globalStartPos` [protected]

Start position of the ray in global cs.

10.116.4.6 `math::Vector3 gazebo::physics::RayShape::relativeEndPos` [protected]

End position of the ray, relative to the body.

10.116.4.7 `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:

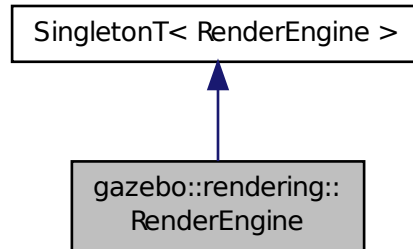
- **RayShape.hh**

10.117 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. 98) 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. 98). Load must happen before Init.*
- void **Load** ()
*Load the parameters for **Ogre** (p. 98). 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

10.117.1 Detailed Description

Adaptor to Ogre3d.

Provides the interface to load, initialize the rendering engine.

10.117.2 Member Enumeration Documentation

10.117.2.1 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.

10.117.3 Member Function Documentation

10.117.3.1 void gazebo::rendering::RenderEngine::AddResourcePath (const std::string & _uri)

Add a new path for **Ogre** (p. 98) to search for resources.

Parameters

in	_uri	URI of the path. The uri should be of the form <code>file://</code> or <code>model://</code>
----	------	--

10.117.3.2 ScenePtr gazebo::rendering::RenderEngine::CreateScene (const std::string & *_name*, bool *_enableVisualizations*)

Create a scene.

Parameters

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

10.117.3.3 void gazebo::rendering::RenderEngine::Fini ()

Tears down the rendering engine.

10.117.3.4 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

10.117.3.5 ScenePtr gazebo::rendering::RenderEngine::GetScene (const std::string & *_name*)

Get a scene by name.

Parameters

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

Returns

A pointer to the **Scene** (p. 632), or NULL if the scene doesn't exist.

10.117.3.6 ScenePtr gazebo::rendering::RenderEngine::GetScene (unsigned int *_index*)

Get a scene by index.

The index should be between 0 and **GetSceneCount()** (p. 613).

Parameters

in	<i>_index</i>	The index of the scene.
----	---------------	-------------------------

Returns

A pointer to a **Scene** (p. 632), or NULL if the index was invalid.

10.117.3.7 `unsigned int gazebo::rendering::RenderEngine::GetSceneCount () const`

Get the number of scenes.

Returns

The number of scenes created by the **RenderEngine** (p. 609).

10.117.3.8 `void gazebo::rendering::RenderEngine::Init ()`

Initialize **Ogre** (p. 98). Load must happen before Init.

10.117.3.9 `void gazebo::rendering::RenderEngine::Load ()`

Load the parameters for **Ogre** (p. 98). Load must happen before Init.

10.117.3.10 `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.
-----------------	--------------------	----------------------------------

10.117.4 Member Data Documentation

10.117.4.1 `void* gazebo::rendering::RenderEngine::dummyContext` [protected]

GLX context used to render the scenes.Used for gui-less operation.

10.117.4.2 `void* gazebo::rendering::RenderEngine::dummyDisplay` [protected]

Pointer to the dummy display.Used for gui-less operation.

10.117.4.3 `uint64_t gazebo::rendering::RenderEngine::dummyWindowId` [protected]

ID for a dummy window. Used for gui-less operation.

10.117.4.4 `Ogre::Root* gazebo::rendering::RenderEngine::root`

Pointer to the root scene node.

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

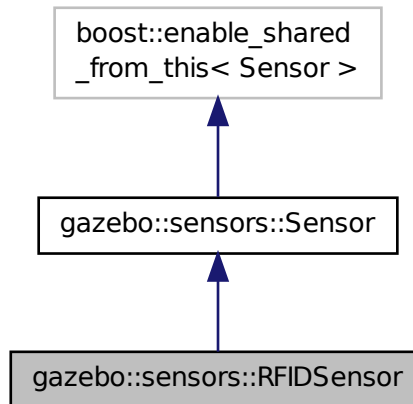
- **RenderEngine.hh**

10.118 gazebo::sensors::RFIDSensor Class Reference

Sensor (p. 652) 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

10.118.1 Detailed Description

Sensor (p. 652) class for RFID type of sensor.

10.118.2 Constructor & Destructor Documentation

10.118.2.1 gazebo::sensors::RFIDSensor::RFIDSensor ()

Constructor.

10.118.2.2 virtual gazebo::sensors::RFIDSensor::~~RFIDSensor () [virtual]

Destructor.

10.118.3 Member Function Documentation

10.118.3.1 void gazebo::sensors::RFIDSensor::AddTag (RFIDTag * *tag*)

10.118.3.2 virtual void gazebo::sensors::RFIDSensor::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 655).

10.118.3.3 virtual void gazebo::sensors::RFIDSensor::Init () [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 657).

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

Load the sensor with SDF parameters.

Parameters

in	<i>_sdf</i>	SDF Sensor (p. 652) parameters.
in	<i>_worldName</i>	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 657).

10.118.3.5 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. 658).

10.118.3.6 `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. 659).

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

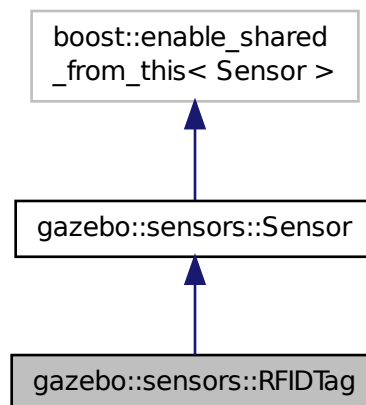
- **RFIDSensor.hh**

10.119 gazebo::sensors::RFIDTag Class Reference

RFIDTag (p. 616) 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

10.119.1 Detailed Description

RFIDTag (p. 616) to interact with RFIDTagSensors.

10.119.2 Constructor & Destructor Documentation

10.119.2.1 gazebo::sensors::RFIDTag::RFIDTag ()

Constructor.

10.119.2.2 virtual gazebo::sensors::RFIDTag::~~RFIDTag () [virtual]

Destructor.

10.119.3 Member Function Documentation

10.119.3.1 virtual void gazebo::sensors::RFIDTag::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 655).

10.119.3.2 `math::Pose gazebo::sensors::RFIDTag::GetTagPose () const` [inline]

Returns pose of tag in world coordinate.

Returns

Pose of object.

References `gazebo::physics::Entity::GetWorldPose()`.

10.119.3.3 `virtual void gazebo::sensors::RFIDTag::Init ()` [virtual]

Initialize the sensor.

Reimplemented from `gazebo::sensors::Sensor` (p. 657).

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

10.119.3.5 `virtual void gazebo::sensors::RFIDTag::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. 658).

10.119.3.6 `virtual void gazebo::sensors::RFIDTag::UpdateImpl (bool)` [protected],[virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

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

Reimplemented from `gazebo::sensors::Sensor` (p. 659).

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

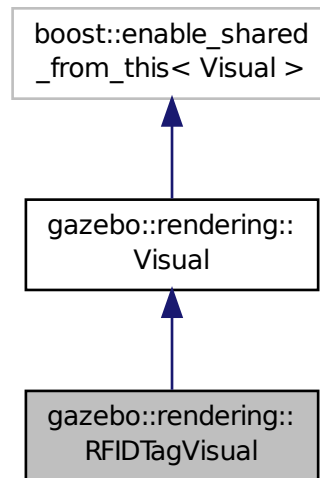
- `RFIDTag.hh`

10.120 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

10.120.1 Detailed Description

Visualization for RFID tags sensor.

10.120.2 Constructor & Destructor Documentation

10.120.2.1 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. 614)

10.120.2.2 virtual gazebo::rendering::RFIDTagVisual::~~RFIDTagVisual () [virtual]

Destructor.

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

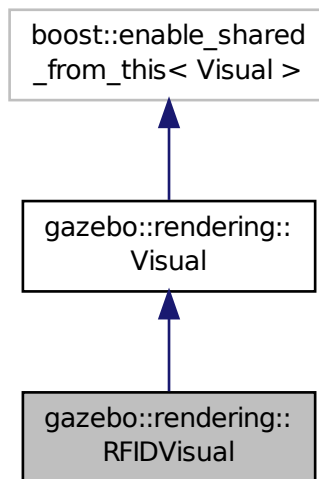
- **RFIDTagVisual.hh**

10.121 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

10.121.1 Detailed Description

Visualization for RFID sensor.

10.121.2 Constructor & Destructor Documentation

10.121.2.1 `gazebo::rendering::RFIDVisual::RFIDVisual (const std::string & _name, VisualPtr _vis, const std::string & _topicName)`

Constructor.

Parameters

<code>in</code>	<code><i>_name</i></code>	Name of the Visual (p. 828).
<code>in</code>	<code><i>_vis</i></code>	Parent Visual (p. 828).
<code>in</code>	<code><i>_topicName</i></code>	Name of the topic which publishes RFID data.

10.121.2.2 `virtual gazebo::rendering::RFIDVisual::~RFIDVisual () [virtual]`

Destructor.

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

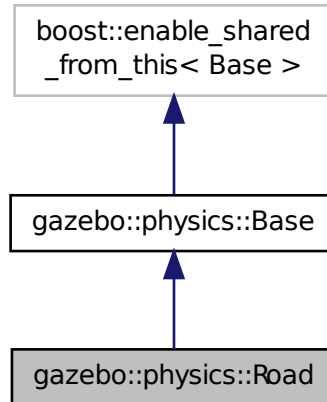
- **RFIDVisual.hh**

10.122 gazebo::physics::Road Class Reference

for building a **Road** (p. 621) 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

10.122.1 Detailed Description

for building a **Road** (p. 621) from SDF

10.122.2 Constructor & Destructor Documentation

10.122.2.1 gazebo::physics::Road::Road (**BasePtr** _parent) [explicit]

Constructor.

Parameters

in	_parent	Parent of this road object.
----	---------	-----------------------------

10.122.2.2 virtual gazebo::physics::Road::~~Road () [virtual]

Destructor.

10.122.3 Member Function Documentation

10.122.3.1 virtual void gazebo::physics::Road::Init () [virtual]

Initialize the road.

Reimplemented from **gazebo::physics::Base** (p. 132).

10.122.3.2 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. 132).

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

- **Road.hh**

10.123 Road Class Reference

Used to render a strip of road.

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

10.123.1 Detailed Description

Used to render a strip of road.

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

- **Road2d.hh**

10.124 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.

10.124.1 Constructor & Destructor Documentation

10.124.1.1 gazebo::rendering::Road2d::Road2d ()

Constructor.

10.124.1.2 virtual gazebo::rendering::Road2d::~~Road2d () [virtual]

Destructor.

10.124.2 Member Function Documentation

10.124.2.1 void gazebo::rendering::Road2d::Load (**VisualPtr** _parent)

Load the visual using a parent visual.

Parameters

in	_parent	Pointer to the parent visual.
----	----------------	-------------------------------

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

- **Road2d.hh**

10.125 gazebo::math::RotationSpline Class Reference

Spline (p. 697) 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

10.125.1 Detailed Description

Spline (p. 697) for rotations.

10.125.2 Constructor & Destructor Documentation

10.125.2.1 gazebo::math::RotationSpline::RotationSpline ()

Constructor. Sets the autoCalc to true.

10.125.2.2 gazebo::math::RotationSpline::~~RotationSpline ()

Destructor. Nothing is done.

10.125.3 Member Function Documentation

10.125.3.1 void gazebo::math::RotationSpline::AddPoint (const Quaternion & `_p`)

Adds a control point to the end of the spline.

Parameters

<code>in</code>	<code>_p</code>	control point
-----------------	-----------------	---------------

10.125.3.2 `void gazebo::math::RotationSpline::Clear ()`

Clears all the points in the spline.

10.125.3.3 `unsigned int gazebo::math::RotationSpline::GetNumPoints () const`

Gets the number of control points in the spline.

Returns

the count

10.125.3.4 `const Quaternion& gazebo::math::RotationSpline::GetPoint (unsigned int _index) const`

Gets the detail of one of the control points of the spline.

Parameters

<code>in</code>	<code><i>_index</i></code>	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)

10.125.3.5 `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

<code>in</code>	<code><i>_t</i></code>	Parametric value.
<code>in</code>	<code><i>_useShortestPath</i></code>	Defines if rotation should take the shortest possible path

Returns

the rotation

10.125.3.6 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>_useShortest-Path</i>	Defines if rotation should take the shortest possible path

Returns

the rotation

10.125.3.7 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.

10.125.3.8 void gazebo::math::RotationSpline::SetAutoCalculate (bool *_autoCalc*)

Tells the spline whether it should automatically calculate tangents on demand as points are added.

Remarks

The spline calculates tangents at each point automatically based on the input points. Normally it does this every time a point changes. However, if you have a lot of points to add in one go, you probably don't want to incur this overhead and would prefer to defer the calculation until you are finished setting all the points. You can do this by calling this method with a parameter of 'false'. Just remember to manually call the recalTangents method when you are done.

Parameters

in	<i>_autoCalc</i>	If true, tangents are calculated for you whenever a point changes. If false, you must call recalTangents to recalculate them when it best suits.
----	------------------	--

10.125.3.9 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

10.125.4 Member Data Documentation

10.125.4.1 `bool gazebo::math::RotationSpline::autoCalc` [protected]

Automatic recalculation of tangents when control points are updated.

10.125.4.2 `std::vector<Quaternion> gazebo::math::RotationSpline::points` [protected]

the control points

10.125.4.3 `std::vector<Quaternion> gazebo::math::RotationSpline::tangents` [protected]

the tangents

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

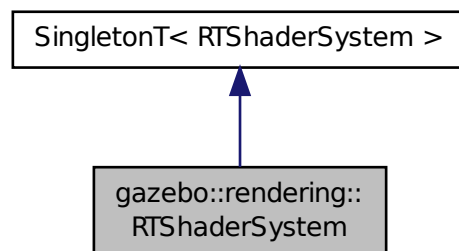
- **RotationSpline.hh**

10.126 gazebo::rendering::RTShaderSystem Class Reference

Implements **Ogre** (p. 98)'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.
- 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

10.126.1 Detailed Description

Implements **Ogre** (p. 98)'s Run-Time Shader system.

This class allows Gazebo to generate per-pixel shaders for every material at run-time.

10.126.2 Member Enumeration Documentation

10.126.2.1 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.

10.126.3 Member Function Documentation

10.126.3.1 void gazebo::rendering::RTShaderSystem::AddScene (ScenePtr _scene)

Add a scene manager.

Parameters

in	_scene	The scene to process
----	--------	----------------------

10.126.3.2 void gazebo::rendering::RTShaderSystem::ApplyShadows (ScenePtr _scene)

Apply shadows to a scene.

Parameters

in	_scene	The scene to receive shadows.
----	--------	-------------------------------

10.126.3.3 void gazebo::rendering::RTShaderSystem::AttachEntity (Visual * vis)

Set an Ogre::Entity to use RT shaders.

Parameters

in	_vis	Visual (p. 828) that will use the RTShaderSystem (p. 628).
----	------	--

10.126.3.4 static void gazebo::rendering::RTShaderSystem::AttachViewport (Ogre::Viewport * _viewport, ScenePtr _scene) [static]

Set a viewport to use shaders.

Parameters

in	_viewport	The viewport to add.
in	_scene	The scene that the viewport uses.

10.126.3.5 void gazebo::rendering::RTShaderSystem::Clear ()

Clear the shader system.

10.126.3.6 void gazebo::rendering::RTShaderSystem::DetachEntity (Visual * _vis)

Remove and entity.

Parameters

in	<i>_vis</i>	Remove this visual.
----	-------------	---------------------

10.126.3.7 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.

10.126.3.8 void gazebo::rendering::RTShaderSystem::Fini ()

Finalize the shader system.

10.126.3.9 void gazebo::rendering::RTShaderSystem::GenerateShaders (Visual * *_vis*)

Generate shaders for an entity.

Parameters

in	<i>_vis</i>	The visual to generate shaders for.
----	-------------	-------------------------------------

10.126.3.10 void gazebo::rendering::RTShaderSystem::Init ()

Init the run time shader system.

10.126.3.11 void gazebo::rendering::RTShaderSystem::RemoveScene (ScenePtr *_scene*)

Remove a scene.

Parameters

in	<i>The</i>	scene to remove
----	------------	-----------------

10.126.3.12 `void gazebo::rendering::RTShaderSystem::RemoveShadows (ScenePtr _scene)`

Remove shadows from a scene.

Parameters

in	<code>_scene</code>	The scene to remove shadows from.
----	---------------------	-----------------------------------

10.126.3.13 `void gazebo::rendering::RTShaderSystem::SetPerPixelLighting (bool _set)`

Set the lighting model to per pixel or per vertex.

Parameters

in	<code>_set</code>	True means to use per-pixel shaders.
----	-------------------	--------------------------------------

10.126.3.14 `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:

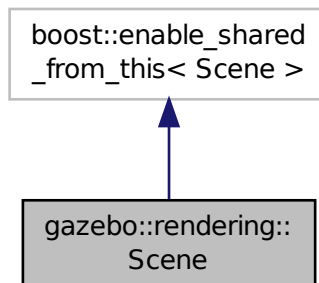
- **RTShaderSystem.hh**

10.127 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. 632).*
- **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.
- **math::Vector3 GetFirstContact** (**CameraPtr** _camera, const **math::Vector2i** &_mousePos)
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. 632).*

 - 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 **SetVisible** (const std::string &_name, bool _visible)
Hide or show a visual.
- 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.
- void **ViewContacts** (bool _view)
Enable or disable contact visualization.

Public Attributes

- SkyX::SkyX * **skyx**
Pointer to the sky.

10.127.1 Detailed Description

Representation of an entire scene graph.

Maintains all the Visuals, Lights, and Cameras for a World.

10.127.2 Constructor & Destructor Documentation

10.127.2.1 `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.

10.127.2.2 `virtual gazebo::rendering::Scene::~Scene () [virtual]`

Destructor.

10.127.3 Member Function Documentation

10.127.3.1 `void gazebo::rendering::Scene::AddVisual (VisualPtr _vis)`

Add a visual to the scene.

Parameters

<code>in</code>	<code>_vis</code>	Visual (p. 828) to add.
-----------------	-------------------	--------------------------------

10.127.3.2 `void gazebo::rendering::Scene::Clear ()`

Clear **rendering::Scene** (p. 632).

10.127.3.3 `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.

10.127.3.4 `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.

10.127.3.5 `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.

10.127.3.6 `void gazebo::rendering::Scene::CreateGrid (uint32_t _cellCount, float _cellLength, float _lineWidth, const common::Color & _color)`

Create a square grid of cells.

Parameters

<code>in</code>	<code>_cellCount</code>	Number of grid cells in one direction.
<code>in</code>	<code>_cellLength</code>	Length of one grid cell.
<code>in</code>	<code>_lineWidth</code>	Width of the grid lines.
<code>in</code>	<code>_color</code>	Color of the grid lines.

10.127.3.7 `UserCameraPtr gazebo::rendering::Scene::CreateUserCamera (const std::string & _name)`

Create a user camera.

A user camera is one design for use with a GUI.

Parameters

<code>in</code>	<code>_name</code>	Name of the UserCamera (p. 774).
-----------------	--------------------	---

Returns

A pointer to the new **UserCamera** (p. 774).

10.127.3.8 `void gazebo::rendering::Scene::DrawLine (const math::Vector3 & _start, const math::Vector3 & _end, const std::string & _name)`

Draw a named line.

Parameters

<code>in</code>	<code>_start</code>	Start position of the line.
<code>in</code>	<code>_end</code>	End position of the line.
<code>in</code>	<code>_name</code>	Name of the line.

10.127.3.9 `common::Color gazebo::rendering::Scene::GetAmbientColor () const`

Get the ambient color.

Returns

The scene's ambient color.

10.127.3.10 `common::Color gazebo::rendering::Scene::GetBackgroundColor () const`

Get the background color.

Returns

The background color.

10.127.3.11 `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. 638).

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.

10.127.3.12 `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.

10.127.3.13 `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.

10.127.3.14 `math::Vector3 gazebo::rendering::Scene::GetFirstContact (CameraPtr _camera, const math::Vector2i & _mousePos)`

Get the world pos of a the first contact at a pixel location.

Parameters

<code>in</code>	<code>_camera</code>	Pointer to the camera.
<code>in</code>	<code>_mousePos</code>	2D position of the mouse in pixels.

Returns

3D position of the first contact point.

10.127.3.15 `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. 639).

Parameters

<code>in</code>	<code>_index</code>	Index of the grid.
-----------------	---------------------	--------------------

10.127.3.16 `uint32_t gazebo::rendering::Scene::GetGridCount () const`

Get the number of grids.

Returns

The number of grids.

10.127.3.17 `double gazebo::rendering::Scene::GetHeightBelowPoint (const math::Vector3 & _pt)`

Get the Z-value of the first object below the given point.

Parameters

<code>in</code>	<code>_pt</code>	Position to search below for a visual.
-----------------	------------------	--

Returns

The Z-value of the nearest visual below the point. Zero is returned if no visual is found.

10.127.3.18 `Heightmap* gazebo::rendering::Scene::GetHeightmap () const`

Get a pointer to the heightmap.

Returns

Pointer to the heightmap, NULL if no heightmap.

10.127.3.19 `uint32_t gazebo::rendering::Scene::GetId () const`

Get the scene ID.

Returns

The ID of the scene.

10.127.3.20 `std::string gazebo::rendering::Scene::GetIdString () const`

Get the scene Id as a string.

Returns

The ID as a string.

10.127.3.21 `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.

10.127.3.22 `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. 640).

Parameters

<code>in</code>	<code>_index</code>	Index of the light.
-----------------	---------------------	---------------------

Returns

Pointer to the **Light** (p. 392) or NULL if index was invalid.

10.127.3.23 `uint32_t gazebo::rendering::Scene::GetLightCount () const`

Get the count of the lights.

Returns

The number of lights.

10.127.3.24 `Ogre::SceneManager* gazebo::rendering::Scene::GetManager () const`

Get the OGRE scene manager.

Returns

Pointer to the **Ogre** (p. 98) SceneManager.

10.127.3.25 `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.

10.127.3.26 `std::string gazebo::rendering::Scene::GetName () const`

Get the name of the scene.

Returns

Name of the scene.

10.127.3.27 `VisualPtr gazebo::rendering::Scene::GetSelectedVisual () const`

Get the currently selected visual.

Returns

Pointer to the currently selected visual, or NULL if nothing is selected.

10.127.3.28 `bool gazebo::rendering::Scene::GetShadowsEnabled () const`

Get whether shadows are on or off.

Returns

True if shadows are enabled.

10.127.3.29 `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. 642).

Parameters

in	_index	Index of the <code>UserCamera</code> (p. 774) to get.
----	--------	---

Returns

Pointer to the `UserCamera` (p. 774), or NULL if the index was invalid.

10.127.3.30 `uint32_t gazebo::rendering::Scene::GetUserCameraCount () const`

Get the number of user cameras in this scene.

Returns

The number of user cameras.

10.127.3.31 `VisualPtr gazebo::rendering::Scene::GetVisual (const std::string & _name) const`

Get a visual by name.

10.127.3.32 `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

10.127.3.33 `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.

10.127.3.34 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.

10.127.3.35 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.

10.127.3.36 VisualPtr gazebo::rendering::Scene::GetWorldVisual () const

Get the top level world visual.

Returns

Pointer to the world visual.

10.127.3.37 void gazebo::rendering::Scene::Init ()

Init **rendering::Scene** (p. 632).

10.127.3.38 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.
----	---------------	----------------------------

10.127.3.39 void gazebo::rendering::Scene::Load ()

Load the scene with default parameters.

10.127.3.40 void gazebo::rendering::Scene::PreRender ()

Process all received messages.

10.127.3.41 void gazebo::rendering::Scene::PrintSceneGraph ()

Print the scene graph to std_out.

10.127.3.42 void gazebo::rendering::Scene::RemoveVisual (VisualIPtr _vis)

Remove a visual from the scene.

Parameters

in	_vis	Visual (p. 828) to remove.
----	------	-----------------------------------

10.127.3.43 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).

10.127.3.44 void gazebo::rendering::Scene::SetAmbientColor (const common::Color & _color)

Set the ambient color.

Parameters

in	_color	The ambient color to use.
----	--------	---------------------------

10.127.3.45 void gazebo::rendering::Scene::SetBackgroundColor (const common::Color & _color)

Set the background color.

Parameters

in	_color	The background color.
----	--------	-----------------------

10.127.3.46 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.

10.127.3.47 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
----	-----------------	---------------------------------

10.127.3.48 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
----	---------------	--

10.127.3.49 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.

10.127.3.50 void gazebo::rendering::Scene::SnapVisualToNearestBelow (const std::string & *_visualName*)

Move the visual to be on top of the nearest visual below it.

Parameters

in	<i>_visualName</i>	Name of the visual to move.
----	--------------------	-----------------------------

10.127.3.51 `std::string gazebo::rendering::Scene::StripSceneName (const std::string & _name) const`

Remove the name of scene from a string.

Parameters

<code>in</code>	<code>_name</code>	Name to string the scene name from.
-----------------	--------------------	-------------------------------------

Returns

The stripped name.

10.127.3.52 `void gazebo::rendering::Scene::ViewContacts (bool _view)`

Enable or disable contact visualization.

Parameters

<code>in</code>	<code>_view</code>	True to enable contact visualization.
-----------------	--------------------	---------------------------------------

10.127.4 Member Data Documentation

10.127.4.1 `SkyX::SkyX* gazebo::rendering::Scene::skyx`

Pointer to the sky.

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

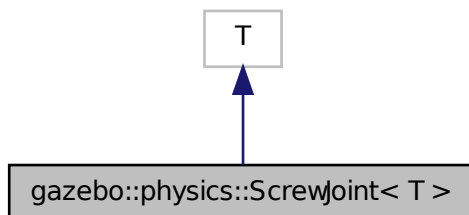
- **Scene.hh**

10.128 gazebo::physics::ScrewJoint< T > Class Template Reference

A screw joint, which has both prismatic and rotational DOFs.

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

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



Public Member Functions

- **ScrewJoint** (**BasePtr** _parent)
Constructor.
- virtual **~ScrewJoint** ()
Destructor.
- virtual **math::Vector3** **GetAnchor** (int _index) const
Get the anchor.
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (**sdf::ElementPtr** _sdf)
*Load a **ScrewJoint** (p. 646).*
- 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.

10.128.1 Detailed Description

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

A screw joint, which has both prismatic and rotational DOFs.

10.128.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	_parent	Parent of the joint.
----	---------	----------------------

References gazebo::physics::Base::SCREW_JOINT.

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

Destructor.

10.128.3 Member Function Documentation

10.128.3.1 `template<class T > math::Vector3 gazebo::physics::ScrewJoint< T >::GetAnchor (int _index) const`
`[virtual]`

Get the anchor.

Parameters

in	<i>_index</i>	Index of the axis. Not Used.
----	---------------	------------------------------

Returns

Anchor for the joint.

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

10.128.3.3 `template<class T > virtual void gazebo::physics::ScrewJoint< T >::Load (sdf::ElementPtr _sdf)`
`[inline],[virtual]`

Load a **ScrewJoint** (p. 646).

Parameters

in	<i>_sdf</i>	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`.

10.128.3.4 `template<class T > void gazebo::physics::ScrewJoint< T >::SetAnchor (int _index, const math::Vector3 & _anchor)`
`[virtual]`

Set the anchor.

Parameters

in	<i>_index</i>	Index of the axis. Not Used.
in	<i>_anchor</i>	Anchor value for the joint.

10.128.3.5 `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	<i>_index</i>	Index of the axis.
in	<i>_threadPitch</i>	Thread pitch value.

10.128.4 Member Data Documentation

10.128.4.1 `template<class T > math::Vector3 gazebo::physics::ScrewJoint< T >::fakeAnchor` [protected]

The anchor value is not used internally.

10.128.4.2 `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**

10.129 sdf::SDF Class Reference

Base **SDF** (p. 649) 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. 649) 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**

10.129.1 Detailed Description

Base **SDF** (p. 649) class.

10.129.2 Constructor & Destructor Documentation

10.129.2.1 `sdf::SDF::SDF ()`

10.129.3 Member Function Documentation

10.129.3.1 `void sdf::SDF::PrintDescription ()`

10.129.3.2 `void sdf::SDF::PrintDoc ()`

10.129.3.3 `void sdf::SDF::PrintValues ()`

10.129.3.4 `void sdf::SDF::PrintWiki ()`

10.129.3.5 `void sdf::SDF::SetFromString (const std::string & _sdfData)`

Set **SDF** (p. 649) values from a string.

10.129.3.6 `std::string sdf::SDF::ToString () const`

10.129.3.7 `void sdf::SDF::Write (const std::string & _filename)`

10.129.4 Member Data Documentation

10.129.4.1 `ElementPtr sdf::SDF::root`

10.129.4.2 `std::string sdf::SDF::version [static]`

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

- **SDF.hh**

10.130 gazebo::rendering::SelectionObj Class Reference

A 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. 650) object.
- `std::string` **GetVisualName** () const

Get the name of the visual the selection obj is attached to.

- void **Init** ()

Initialize the **rendering::SelectionObj** (p. 650) 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.

10.130.1 Detailed Description

A graphical selection object.

Used to draw a visual around a selected object.

10.130.2 Constructor & Destructor Documentation

10.130.2.1 gazebo::rendering::SelectionObj::SelectionObj (Scene * _scene)

Constructor.

Parameters

in	<i>_scene</i>	Scene (p. 632) to use.
----	---------------	-------------------------------

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

Destructor.

10.130.3 Member Function Documentation

10.130.3.1 void gazebo::rendering::SelectionObj::Attach (VisualPtr _visual)

Set the position of the node.

Parameters

in	<i>This</i>	draws the selection object around the passed in visual.
----	-------------	---

10.130.3.2 void gazebo::rendering::SelectionObj::Clear ()

Clear the **rendering::SelectionObj** (p. 650) object.

10.130.3.3 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.

10.130.3.4 `void gazebo::rendering::SelectionObj::Init ()`

Initialize the **rendering::SelectionObj** (p. 650) object.

10.130.3.5 `bool gazebo::rendering::SelectionObj::IsActive () const`

Return true if the user is move the selection obj.

Returns

True if something is selected.

10.130.3.6 `void gazebo::rendering::SelectionObj::SetActive (bool _active)`

Set true if the user is moving the selection obj.

Parameters

<code>in</code>	<code><i>_active</i></code>	True if the user is interacting with the selection object.
-----------------	-----------------------------	--

10.130.3.7 `void gazebo::rendering::SelectionObj::SetHighlight (const std::string & _mod)`

Highlight the selection object based on a modifier.

Parameters

<code>in</code>	<code><i>_mod</i></code>	Modifier used when highlighting the selection object.
-----------------	--------------------------	---

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

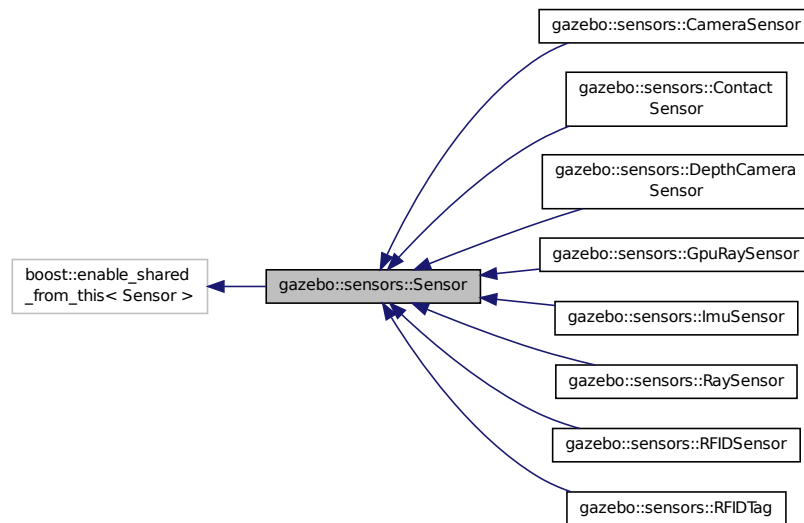
- **SelectionObj.hh**

10.131 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.
- 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.*

 - 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 to the SDF element for the sensor.

- **common::Time updatePeriod**

*Desired time between updates, set indirectly by **Sensor::SetUpdateRate** (p. 658).*

- **gazebo::physics::WorldPtr world**

Pointer to the world.

10.131.1 Detailed Description

Base class for sensors.

10.131.2 Constructor & Destructor Documentation

10.131.2.1 gazebo::sensors::Sensor::Sensor ()

Constructor.

10.131.2.2 virtual gazebo::sensors::Sensor::~Sensor () [virtual]

Destructor.

10.131.3 Member Function Documentation

10.131.3.1 void gazebo::sensors::Sensor::FillMsg (msgs::Sensor & _msg)

fills a msgs::Sensor message.

Parameters

out	<code>_msg</code>	Message to fill.
-----	-------------------	------------------

10.131.3.2 virtual void gazebo::sensors::Sensor::Fini () [virtual]

Finalize the sensor.

Reimplemented in **gazebo::sensors::CameraSensor** (p. 175), **gazebo::sensors::GpuRaySensor** (p. 321), **gazebo::sensors::ContactSensor** (p. 226), **gazebo::sensors::DepthCameraSensor** (p. 244), **gazebo::sensors::RFIDSensor** (p. 615), **gazebo::sensors::RaySensor** (p. 599), **gazebo::sensors::RFIDTag** (p. 617), and **gazebo::sensors::ImuSensor** (p. 355).

10.131.3.3 common::Time gazebo::sensors::Sensor::GetLastMeasurementTime ()

Return last measurement time.

Returns

Time of last measurement.

10.131.3.4 `common::Time gazebo::sensors::Sensor::GetLastUpdateTime ()`

Return last update time.

Returns

Time of last update.

10.131.3.5 `std::string gazebo::sensors::Sensor::GetName () const`

Get name.

Returns

Name of sensor.

10.131.3.6 `std::string gazebo::sensors::Sensor::GetParentName () const`

Returns the name of the sensor parent.

The parent name is set by **Sensor::SetParent** (p. 658).

Returns

Name of Parent.

10.131.3.7 `virtual math::Pose gazebo::sensors::Sensor::GetPose () const [virtual]`

Get the current pose.

Returns

Current pose of the sensor.

10.131.3.8 `std::string gazebo::sensors::Sensor::GetScopedName () const`

Get fully scoped name of the sensor.

Returns

world_name::parent_name::sensor_name.

10.131.3.9 `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. 601), and **gazebo::sensors::CameraSensor** (p. 176).

10.131.3.10 `std::string gazebo::sensors::Sensor::GetType () const`

Get sensor type.

Returns

Type of sensor.

10.131.3.11 `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.

10.131.3.12 `std::string gazebo::sensors::Sensor::GetWorldName () const`

Returns the name of the world the sensor is in.

Returns

Name of the world.

10.131.3.13 `virtual void gazebo::sensors::Sensor::Init () [virtual]`

Initialize the sensor.

Reimplemented in `gazebo::sensors::GpuRaySensor` (p. 327), `gazebo::sensors::ContactSensor` (p. 227), `gazebo::sensors::CameraSensor` (p. 176), `gazebo::sensors::DepthCameraSensor` (p. 244), `gazebo::sensors::RaySensor` (p. 602), `gazebo::sensors::RFIDSensor` (p. 615), `gazebo::sensors::RFIDTag` (p. 618), and `gazebo::sensors::ImuSensor` (p. 356).

10.131.3.14 `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::ContactSensor` (p. 227).

10.131.3.15 `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 Sensor (p. 652) parameters.
<code>in</code>	<code>_worldName</code>	Name of world to load from.

Reimplemented in **gazebo::sensors::ContactSensor** (p. 228), **gazebo::sensors::CameraSensor** (p. 176), **gazebo::sensors::RFIDSensor** (p. 615), and **gazebo::sensors::ImuSensor** (p. 356).

10.131.3.16 `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. 327), **gazebo::sensors::ContactSensor** (p. 228), **gazebo::sensors::CameraSensor** (p. 176), **gazebo::sensors::DepthCameraSensor** (p. 245), **gazebo::sensors::RaySensor** (p. 602), **gazebo::sensors::RFIDSensor** (p. 615), **gazebo::sensors::RFIDTag** (p. 618), and **gazebo::sensors::ImuSensor** (p. 356).

10.131.3.17 `virtual void gazebo::sensors::Sensor::SetActive (bool _value) [virtual]`

Set whether the sensor is active or not.

Parameters

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

Reimplemented in **gazebo::sensors::DepthCameraSensor** (p. 245).

10.131.3.18 `virtual void gazebo::sensors::Sensor::SetParent (const std::string & _name) [virtual]`

Set the parent of the sensor.

Parameters

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

Reimplemented in **gazebo::sensors::CameraSensor** (p. 177), and **gazebo::sensors::DepthCameraSensor** (p. 245).

10.131.3.19 `void gazebo::sensors::Sensor::SetUpdateRate (double _hz)`

Set the update rate of the sensor.

Parameters

<code>in</code>	<code>_hz</code>	update rate of sensor.
-----------------	------------------	------------------------

10.131.3.20 `void gazebo::sensors::Sensor::Update (bool _force)`

Update the sensor.

Parameters

in	<code>_force</code>	True to force update, false otherwise.
----	---------------------	--

10.131.3.21 `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	<code>_force</code>	True if update is forced, false if not
----	---------------------	--

Reimplemented in `gazebo::sensors::CameraSensor` (p. 177), `gazebo::sensors::GpuRaySensor` (p. 328), `gazebo::sensors::ContactSensor` (p. 228), `gazebo::sensors::DepthCameraSensor` (p. 245), `gazebo::sensors::RFIDSensor` (p. 616), `gazebo::sensors::RaySensor` (p. 603), `gazebo::sensors::RFIDTag` (p. 618), and `gazebo::sensors::ImuSensor` (p. 356).

10.131.4 Member Data Documentation

10.131.4.1 `bool gazebo::sensors::Sensor::active` `[protected]`

True if sensor generation is active.

10.131.4.2 `std::vector<event::ConnectionPtr> gazebo::sensors::Sensor::connections` `[protected]`

All event connections.

10.131.4.3 `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`.

10.131.4.4 `common::Time gazebo::sensors::Sensor::lastUpdateTime` `[protected]`

Time of the last update.

10.131.4.5 `transport::NodePtr gazebo::sensors::Sensor::node` `[protected]`

Node for communication.

10.131.4.6 `std::string gazebo::sensors::Sensor::parentName` `[protected]`

Name of the parent.

10.131.4.7 `std::vector<SensorPluginPtr> gazebo::sensors::Sensor::plugins` [protected]

All the plugins for the sensor.

10.131.4.8 `math::Pose gazebo::sensors::Sensor::pose` [protected]

Pose of the sensor.

10.131.4.9 `transport::SubscriberPtr gazebo::sensors::Sensor::poseSub` [protected]

Subscribe to pose updates.

10.131.4.10 `sdf::ElementPtr gazebo::sensors::Sensor::sdf` [protected]

Pointer the the SDF element for the sensor.

10.131.4.11 `common::Time gazebo::sensors::Sensor::updatePeriod` [protected]

Desired time between updates, set indirectly by `Sensor::SetUpdateRate` (p. 658).

10.131.4.12 `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`

10.132 SensorFactor Class Reference

The sensor factory; the class is just for namespacing purposes.

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

10.132.1 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`

10.133 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).

10.133.1 Member Function Documentation

10.133.1.1 static void gazebo::sensors::SensorFactory::GetSensorTypes (std::vector< std::string > & _types) [static]

Get all the sensor types.

Parameters

<code>_types</code>	Vector of strings of the sensor types, populated by function
---------------------	--

10.133.1.2 static **SensorPtr** gazebo::sensors::SensorFactory::NewSensor (const std::string & _className) [static]

Create a new instance of a sensor.

Used by the world when reading the world file.

Parameters

<code>in</code>	<code>_className</code>	Name of sensor class
-----------------	-------------------------	----------------------

Returns

Pointer to **Sensor** (p. 652)

10.133.1.3 static void gazebo::sensors::SensorFactory::RegisterAll () [static]

Register all known sensors.

- **sensors::CameraSensor** (p. 173)
- **sensors::DepthCameraSensor** (p. 242)
- **sensors::GpuRaySensor** (p. 316)
- **sensors::RaySensor** (p. 596)
- **sensors::ContactSensor** (p. 224)
- **sensors::RFIDSensor** (p. 614)
- **sensors::RFIDTag** (p. 616)

10.133.1.4 `static void gazebo::sensors::SensorFactory::RegisterSensor (const std::string & _className, SensorFactoryFn _factoryfn) [static]`

Register a sensor class (called by sensor registration function).

Parameters

in	<code>_className</code>	Name of class of sensor to register.
in	<code>_factoryfn</code>	Function handle for registration.

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

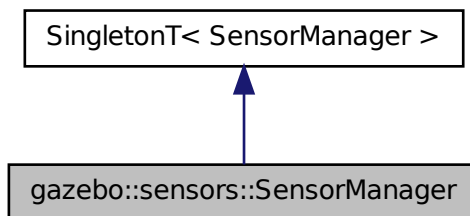
- **SensorFactory.hh**

10.134 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

10.134.1 Detailed Description

Class to manage and update all sensors.

10.134.2 Member Function Documentation

- 10.134.2.1 `std::string gazebo::sensors::SensorManager::CreateSensor (sdf::ElementPtr _elem, const std::string & _worldName, const std::string & _parentName)`

Add a sensor from an SDF element.

This function will also Load and Init the sensor.

Parameters

<code>in</code>	<code>_elem</code>	The SDF element that describes the sensor
<code>in</code>	<code>_worldName</code>	Name of the world in which to create the sensor
<code>in</code>	<code>_parentName</code>	The name of the parent link which the sensor is attached to.

Returns

The name of the sensor

- 10.134.2.2 `void gazebo::sensors::SensorManager::Fini ()`

Finalize all the sensors.

- 10.134.2.3 `SensorPtr gazebo::sensors::SensorManager::GetSensor (const std::string & _name)`

Get a sensor.

Parameters

in	<i>_name</i>	The name of a sensor to find.
----	--------------	-------------------------------

Returns

A pointer to the sensor. NULL if not found.

10.134.2.4 `Sensor_V gazebo::sensors::SensorManager::GetSensors () const`

Get all the sensors.

Returns

Vector of all the sensors.

10.134.2.5 `void gazebo::sensors::SensorManager::GetSensorTypes (std::vector< std::string > & _types) const`

Get all the sensor types.

Parameters

out	<i>All</i>	the sensor types.
-----	------------	-------------------

10.134.2.6 `void gazebo::sensors::SensorManager::Init ()`

Init all the sensors.

10.134.2.7 `void gazebo::sensors::SensorManager::RemoveSensor (const std::string & _name)`

Remove a sensor.

Parameters

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

10.134.2.8 `void gazebo::sensors::SensorManager::RemoveSensors ()`

Remove all sensors.

10.134.2.9 `void gazebo::sensors::SensorManager::Run ()`

Run the sensor manager update in a new thread.

10.134.2.10 `bool gazebo::sensors::SensorManager::SensorsInitialized ()`

True if `SensorManager::initSensors` queue is empty i.e.

all sensors managed by **SensorManager** (p. 662) have been initialized

10.134.2.11 void gazebo::sensors::SensorManager::Stop ()

Stop the run thread.

10.134.2.12 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:

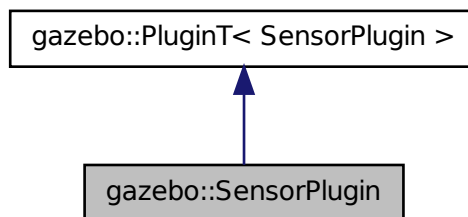
- **SensorManager.hh**

10.135 gazebo::SensorPlugin Class Reference

A plugin with access to physics::Sensor.

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

Inheritance diagram for gazebo::SensorPlugin:



Public Member Functions

- **SensorPlugin** ()
Constructor.
- virtual **~SensorPlugin** ()
Destructor.
- virtual void **Init** ()
Override this method for custom plugin initialization behavior.

- virtual void **Load** (**sensors::SensorPtr** _sensor, **sdf::ElementPtr** _sdf)=0
Load function.
- virtual void **Reset** ()
Override this method for custom plugin reset behavior.

Additional Inherited Members

10.135.1 Detailed Description

A plugin with access to physics::Sensor.

See reference.

10.135.2 Constructor & Destructor Documentation

10.135.2.1 gazebo::SensorPlugin::SensorPlugin () [inline]

Constructor.

References gazebo::SENSOR_PLUGIN, and gazebo::PluginT< SensorPlugin >::type.

10.135.2.2 virtual gazebo::SensorPlugin::~SensorPlugin () [inline],[virtual]

Destructor.

10.135.3 Member Function Documentation

10.135.3.1 virtual void gazebo::SensorPlugin::Init () [inline],[virtual]

Override this method for custom plugin initialization behavior.

10.135.3.2 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.

10.135.3.3 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**

10.136 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** ¶ms)
- void **Stop** ()

Public Attributes

- int **systemPluginsArgc**
- char ** **systemPluginsArgv**

10.136.1 Constructor & Destructor Documentation

10.136.1.1 gazebo::Server::Server ()

10.136.1.2 virtual gazebo::Server::~Server () [virtual]

10.136.2 Member Function Documentation

10.136.2.1 void gazebo::Server::Fini ()

10.136.2.2 bool gazebo::Server::GetInitialized () const

10.136.2.3 void gazebo::Server::Init ()

10.136.2.4 bool gazebo::Server::LoadFile (const std::string & *_filename* = "worlds/empty.world")

10.136.2.5 bool gazebo::Server::LoadString (const std::string & *_sdfString*)

10.136.2.6 bool gazebo::Server::ParseArgs (int *argc*, char ** *argv*)

10.136.2.7 void gazebo::Server::PrintUsage ()

10.136.2.8 void gazebo::Server::Run ()

10.136.2.9 void gazebo::Server::SetParams (const **common::StrStr_M** & *params*)

10.136.2.10 void gazebo::Server::Stop ()

10.136.3 Member Data Documentation

10.136.3.1 int gazebo::Server::systemPluginsArgc

10.136.3.2 char** gazebo::Server::systemPluginsArgv

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

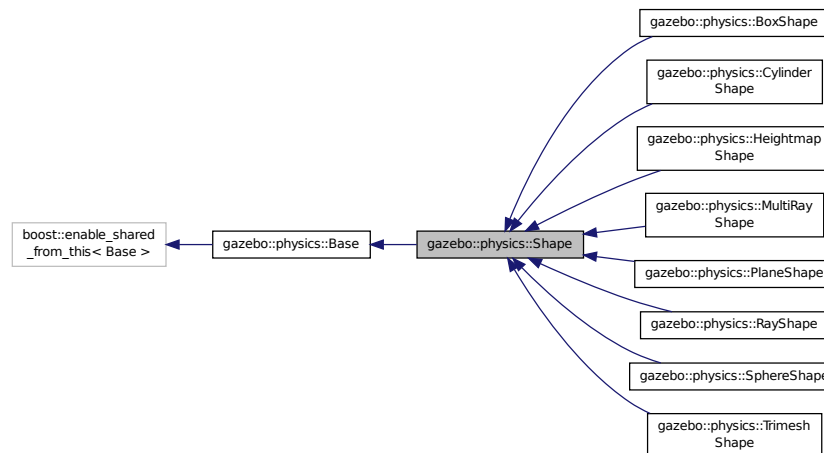
- **Server.hh**

10.137 gazebo::physics::Shape Class Reference

Base (p. 125) 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 **FillShapeMsg** (msgs::Geometry &_msg) **GAZEBO_DEPRECATED**
Deprecated.
- virtual void **GetInertial** (double _mass, **InertialPtr** _inertial) const **GAZEBO_DEPRECATED**

Deprecated.

- virtual double **GetMass** (double `_density`) const **GAZEBO_DEPRECATED**

Deprecated.

- 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

10.137.1 Detailed Description

Base (p. 125) class for all shapes.

10.137.2 Constructor & Destructor Documentation

10.137.2.1 `gazebo::physics::Shape::Shape (CollisionPtr _parent)` `[explicit]`

Constructor.

Parameters

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

10.137.2.2 `virtual gazebo::physics::Shape::~~Shape ()` `[virtual]`

Destructor.

10.137.3 Member Function Documentation

10.137.3.1 `virtual void gazebo::physics::Shape::FillMsg (msgs::Geometry & _msg)` `[pure virtual]`

Fill in the values for a geometry message.

Parameters

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

Implemented in `gazebo::physics::MultiRayShape` (p. 495), `gazebo::physics::RayShape` (p. 606), `gazebo::physics::HeightmapShape` (p. 343), `gazebo::physics::PlaneShape` (p. 551), `gazebo::physics::TrimeshShape` (p. 763), `gazebo::physics::CylinderShape` (p. 234), `gazebo::physics::BoxShape` (p. 142), and `gazebo::physics::SphereShape` (p. 696).

10.137.3.2 `virtual void gazebo::physics::Shape::FillShapeMsg (msgs::Geometry & _msg) [virtual]`

Deprecated.

Reimplemented in `gazebo::physics::BoxShape` (p. 142).

10.137.3.3 `virtual void gazebo::physics::Shape::GetInertial (double _mass, InertialPtr _inertial) const [virtual]`

Deprecated.

Reimplemented in `gazebo::physics::CylinderShape` (p. 234), `gazebo::physics::BoxShape` (p. 142), and `gazebo::physics::SphereShape` (p. 696).

10.137.3.4 `virtual double gazebo::physics::Shape::GetMass (double _density) const [inline],[virtual]`

Deprecated.

Reimplemented in `gazebo::physics::TrimeshShape` (p. 764), `gazebo::physics::CylinderShape` (p. 235), `gazebo::physics::BoxShape` (p. 142), and `gazebo::physics::SphereShape` (p. 696).

10.137.3.5 `virtual void gazebo::physics::Shape::Init () [pure virtual]`

Initialize the shape.

Reimplemented from `gazebo::physics::Base` (p. 132).

Implemented in `gazebo::physics::RayShape` (p. 607), `gazebo::physics::HeightmapShape` (p. 344), `gazebo::physics::TrimeshShape` (p. 764), `gazebo::physics::MultiRayShape` (p. 498), `gazebo::physics::PlaneShape` (p. 552), `gazebo::physics::SphereShape` (p. 697), `gazebo::physics::BoxShape` (p. 143), and `gazebo::physics::CylinderShape` (p. 235).

10.137.3.6 `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. 498), `gazebo::physics::RayShape` (p. 607), `gazebo::physics::HeightmapShape` (p. 345), `gazebo::physics::PlaneShape` (p. 552), `gazebo::physics::TrimeshShape` (p. 764), `gazebo::physics::CylinderShape` (p. 235), `gazebo::physics::BoxShape` (p. 143), and `gazebo::physics::SphereShape` (p. 697).

10.137.4 Member Data Documentation

10.137.4.1 `CollisionPtr gazebo::physics::Shape::collisionParent [protected]`

This shape's collision parent.

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

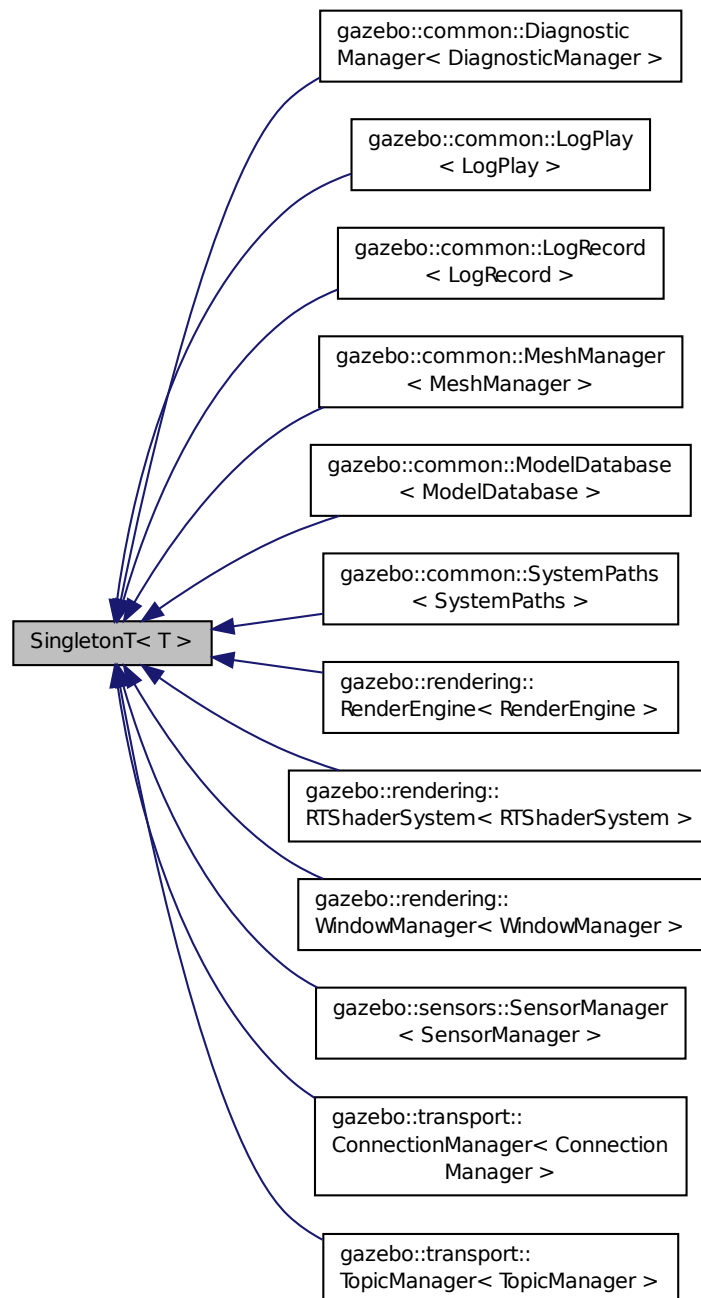
- **Shape.hh**

10.138 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.

10.138.1 Detailed Description

```
template<class T>class SingletonT< T >
```

Singleton template class.

10.138.2 Constructor & Destructor Documentation

10.138.2.1 `template<class T> SingletonT< T >::SingletonT ()` `[inline]`, `[protected]`

Constructor.

10.138.2.2 `template<class T> virtual SingletonT< T >::~~SingletonT ()` `[inline]`, `[protected]`, `[virtual]`

Destructor.

10.138.3 Member Function Documentation

10.138.3.1 `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**

10.139 gazebo::common::Skeleton Class Reference

A skeleton.

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

Public Member Functions

- **Skeleton** ()
Constructor.
- **Skeleton** (**SkeletonNode** * _root)
Constructor.
- virtual ~**Skeleton** ()
Destructor.
- void **AddAnimation** (**SkeletonAnimation** * _anim)
Add an animation.
- void **AddVertNodeWeight** (unsigned int _vertex, std::string _node, double _weight)
Add a new weight to a node (bone)
- **SkeletonAnimation** * **GetAnimation** (const unsigned int _i)
Find animation.
- **math::Matrix4** **GetBindShapeTransform** ()
Return bind pose skeletal transform.
- **SkeletonNode** * **GetNodeByHandle** (unsigned int _handle)
Find or create node with handle.
- **SkeletonNode** * **GetNodeById** (std::string _id)
Find node by index.
- **SkeletonNode** * **GetNodeByName** (std::string _name)
Find a node.
- **NodeMap** **GetNodes** ()
Get a copy of 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

10.139.1 Detailed Description

A skeleton.

10.139.2 Constructor & Destructor Documentation

10.139.2.1 gazebo::common::Skeleton::Skeleton ()

Constructor.

10.139.2.2 gazebo::common::Skeleton::Skeleton (**SkeletonNode** * *_root*)

Constructor.

Parameters

in	<i>_root</i>	node
----	--------------	------

10.139.2.3 virtual gazebo::common::Skeleton::~Skeleton () [virtual]

Destructor.

10.139.3 Member Function Documentation

10.139.3.1 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
----	--------------	----------------------

10.139.3.2 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)

10.139.3.3 void gazebo::common::Skeleton::BuildNodeMap () [protected]

Initializes the handle numbers for each node in the map using breadth first traversal.

10.139.3.4 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

10.139.3.5 math::Matrix4 gazebo::common::Skeleton::GetBindShapeTransform ()

Return bind pose skeletal transform.

Returns

a matrix

10.139.3.6 SkeletonNode* gazebo::common::Skeleton::GetNodeByHandle (unsigned int *_handle*)

Find or create node with handle.

Parameters

in	<i>_handle</i>	
----	----------------	--

Returns

the node. A new node is created if it didn't exist

10.139.3.7 SkeletonNode* gazebo::common::Skeleton::GetNodeById (std::string *_id*)

Find node by index.

Parameters

<i>in</i>	<i>_id</i>	the index
-----------	------------	-----------

Returns

the node, or NULL if not found

10.139.3.8 SkeletonNode* gazebo::common::Skeleton::GetNodeByName (std::string *_name*)

Find a node.

Parameters

<i>in</i>	<i>_name</i>	the name of the node to look for
-----------	--------------	----------------------------------

Returns

the node, or NULL if not found

10.139.3.9 NodeMap gazebo::common::Skeleton::GetNodes ()

Get a copy of the node dictionary.

10.139.3.10 unsigned int gazebo::common::Skeleton::GetNumAnimations ()

Returns the number of animations.

Returns

the count

10.139.3.11 unsigned int gazebo::common::Skeleton::GetNumJoints ()

Returns the number of joints.

Returns

the count

10.139.3.12 unsigned int gazebo::common::Skeleton::GetNumNodes ()

Returns the node count.

Returns

the count

10.139.3.13 `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

10.139.3.14 `SkeletonNode* gazebo::common::Skeleton::GetRootNode ()`

Return the root.

Returns

the root

10.139.3.15 `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

10.139.3.16 `void gazebo::common::Skeleton::PrintTransforms ()`

Outputs the transforms to `std::err` stream.

10.139.3.17 `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
-----------------	-------------------------	----------------

10.139.3.18 `void gazebo::common::Skeleton::SetBindShapeTransform (math::Matrix4 _trans)`

Set the bind pose skeletal transform.

Parameters

in	<i>_trans</i>	the transform
----	---------------	---------------

10.139.3.19 void gazebo::common::Skeleton::SetNumVertAttached (unsigned int *_vertices*)

Resizes the raw node weight array.

Parameters

in	<i>_vertices</i>	the new size
----	------------------	--------------

10.139.3.20 void gazebo::common::Skeleton::SetRootNode (**SkeletonNode** * *_node*)

Change the root node.

Parameters

in	<i>_node</i>	the new node
----	--------------	--------------

10.139.4 Member Data Documentation

10.139.4.1 **std::vector<SkeletonAnimation*>** gazebo::common::Skeleton::anim [protected]

the array of animations

10.139.4.2 **math::Matrix4** gazebo::common::Skeleton::bindShapeTransform [protected]

the bind pose skeletal transform

10.139.4.3 **NodeMap** gazebo::common::Skeleton::nodes [protected]

The dictionary of nodes, indexed by name.

10.139.4.4 **RawNodeWeights** gazebo::common::Skeleton::rawNW [protected]

the node weight table

10.139.4.5 **SkeletonNode*** gazebo::common::Skeleton::root [protected]

the root node

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

- **Skeleton.hh**

10.140 gazebo::common::SkeletonAnimation Class Reference

Skeleton (p. 672) 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

10.140.1 Detailed Description

Skeleton (p. 672) animation.

10.140.2 Constructor & Destructor Documentation

10.140.2.1 gazebo::common::SkeletonAnimation::SkeletonAnimation (const std::string & *_name*)

The Constructor.

Parameters

in	<i>_name</i>	the name of the animation
----	--------------	---------------------------

10.140.2.2 gazebo::common::SkeletonAnimation::~~SkeletonAnimation ()

The destructor.

Clears the list without destroying the animations

10.140.3 Member Function Documentation

10.140.3.1 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

10.140.3.2 void gazebo::common::SkeletonAnimation::AddKeyFrame (const std::string & *_node*, const double *_time*, const math::Pose *_pose*)

Adds or replaces a named key frame at a specific time.

Parameters

in	<i>_node</i>	the name of the new or existing node
in	<i>_time</i>	the time
in	<i>_pose</i>	the key frame transformation as a math::Pose (p. 556)

10.140.3.3 double gazebo::common::SkeletonAnimation::GetLength () const

Returns the duration of the animations.

Returns

the duration in seconds

10.140.3.4 `std::string gazebo::common::SkeletonAnimation::GetName () const`

Returns the name.

Returns

the name

10.140.3.5 `unsigned int gazebo::common::SkeletonAnimation::GetNodeCount () const`

Returns the number of animation nodes.

Returns

the count

10.140.3.6 `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

10.140.3.7 `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

10.140.3.8 `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)

10.140.3.9 `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

10.140.3.10 `void gazebo::common::SkeletonAnimation::Scale (const double _scale)`

Scales every animation in the animations list.

Parameters

in	<i>_scale</i>	the scaling factor
----	---------------	--------------------

10.140.3.11 `void gazebo::common::SkeletonAnimation::SetName (const std::string & _name)`

Changes the name.

Parameters

in	<i>_name</i>	the new name
----	--------------	--------------

10.140.4 Member Data Documentation

10.140.4.1 `std::map<std::string, NodeAnimation*> gazebo::common::SkeletonAnimation::animations` `[protected]`

a dictionary of node animations

10.140.4.2 double gazebo::common::SkeletonAnimation::length [protected]

the duration of the longest animation

10.140.4.3 std::string gazebo::common::SkeletonAnimation::name [protected]

the node name

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

- **SkeletonAnimation.hh**

10.141 gazebo::common::SkeletonNode Class Reference

A skeleton node.

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

Public Types

- enum **SkeletonNodeType** { **NODE**, **JOINT** }
enumeration of node types

Public Member Functions

- **SkeletonNode** (**SkeletonNode** *_parent)
Constructor.
- **SkeletonNode** (**SkeletonNode** *_parent, std::string _name, std::string _id, **SkeletonNodeType** _type=**JOINT**)
Constructor.
- virtual ~**SkeletonNode** ()
Destructor.
- void **AddChild** (**SkeletonNode** *_child)
Add a new child.
- void **AddRawTransform** (**NodeTransform** _t)
Add a raw transform.
- **SkeletonNode** * **GetChild** (unsigned int _index)
Find a child by index.
- **SkeletonNode** * **GetChildById** (std::string _id)
Get child by string id.
- **SkeletonNode** * **GetChildByName** (std::string _name)
Get child by name.
- unsigned int **GetChildCount** ()
Returns the children count.
- unsigned int **GetHandle** ()
Get the handle index.
- std::string **GetId** ()
Returns the index.

- **math::Matrix4 GetInverseBindTransform ()**
Retrieve the inverse of the bind pose skeletal transform.
- **math::Matrix4 GetModelTransform ()**
Retrieve the model transform.
- **std::string GetName ()**
Returns the name.
- **unsigned int GetNumRawTrans ()**
Return the raw transformations count.
- **SkeletonNode * GetParent ()**
Returns the parent node.
- **NodeTransform GetRawTransform (unsigned int _i)**
Find a raw transformation.
- **std::vector< NodeTransform > GetRawTransforms ()**
Retrieve the raw transformations.
- **math::Matrix4 GetTransform ()**
Get transform relative to parent.
- **std::vector< NodeTransform > GetTransforms ()**
Returns a copy of the array of transformations.
- **bool IsJoint ()**
Is a joint query.
- **bool IsRootNode ()**
Queries whether a node has no parent parent.
- **void Reset (bool _resetChildren)**
Reset the transformation to the initial transformation.
- **void SetHandle (unsigned int _h)**
Assign a handle number.
- **void SetId (std::string _id)**
Change the id string.
- **void SetInitialTransform (math::Matrix4 _tras)**
Sets the initial transformation.
- **void SetInverseBindTransform (math::Matrix4 _invBM)**
Assign the inverse of the bind pose skeletal transform.
- **void SetModelTransform (math::Matrix4 _trans, bool _updateChildren=true)**
Set the model transformation.
- **void SetName (std::string _name)**
Change the name.
- **void SetParent (SkeletonNode *_parent)**
Set the parent node.
- **void SetTransform (math::Matrix4 _trans, bool _updateChildren=true)**
Set a transformation.
- **void SetType (SkeletonNodeType _type)**
Change the skeleton node type.
- **void UpdateChildrenTransforms ()**
Apply model transformations in order for each node in the tree.

Protected Attributes

- `std::vector< SkeletonNode * >` **children**
the children nodes
- `unsigned int` **handle**
handle index number
- `std::string` **id**
a string identifier
- `math::Matrix4` **initialTransform**
the initial transformation
- `math::Matrix4` **invBindTransform**
the inverse of the bind pose skeletal transform
- `math::Matrix4` **modelTransform**
the model transformation
- `std::string` **name**
the name of the skeletal node
- `SkeletonNode *` **parent**
the parent node
- `std::vector< NodeTransform >` **rawTransforms**
the raw transformation
- `math::Matrix4` **transform**
the transform
- `SkeletonNodeType` **type**
the type fo node

10.141.1 Detailed Description

A skeleton node.

10.141.2 Member Enumeration Documentation

10.141.2.1 enum gazebo::common::SkeletonNode::SkeletonNodeType

enumeration of node types

Enumerator:

NODE

JOINT

10.141.3 Constructor & Destructor Documentation

10.141.3.1 gazebo::common::SkeletonNode::SkeletonNode (`SkeletonNode * _parent`)

Constructor.

Parameters

<code>in</code>	<code>_parent</code>	The parent node
-----------------	----------------------	-----------------

10.141.3.2 `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

10.141.3.3 `virtual gazebo::common::SkeletonNode::~~SkeletonNode () [virtual]`

Destructor.

10.141.4 Member Function Documentation

10.141.4.1 `void gazebo::common::SkeletonNode::AddChild (SkeletonNode * _child)`

Add a new child.

Parameters

in	<code>_child</code>	a child
----	---------------------	---------

10.141.4.2 `void gazebo::common::SkeletonNode::AddRawTransform (NodeTransform _t)`

Add a raw transform.

Parameters

in	<code>_t</code>	the transform
----	-----------------	---------------

10.141.4.3 `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

10.141.4.4 `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

10.141.4.5 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

10.141.4.6 unsigned int gazebo::common::SkeletonNode::GetChildCount ()

Returns the children count.

Returns

the count

10.141.4.7 unsigned int gazebo::common::SkeletonNode::GetHandle ()

Get the handle index.

Returns

the handle index

10.141.4.8 std::string gazebo::common::SkeletonNode::GetId ()

Returns the index.

Returns

the id string

10.141.4.9 math::Matrix4 gazebo::common::SkeletonNode::GetInverseBindTransform ()

Retrieve the inverse of the bind pose skeletal transform.

Returns

the transform

10.141.4.10 `math::Matrix4 gazebo::common::SkeletonNode::GetModelTransform ()`

Retrieve the model transform.

Returns

the transform

10.141.4.11 `std::string gazebo::common::SkeletonNode::GetName ()`

Returns the name.

Returns

the name

10.141.4.12 `unsigned int gazebo::common::SkeletonNode::GetNumRawTrans ()`

Return the raw transformations count.

Returns

the count

10.141.4.13 `SkeletonNode* gazebo::common::SkeletonNode::GetParent ()`

Returns the parent node.

Returns

the parent

10.141.4.14 `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

10.141.4.15 `std::vector<NodeTransform> gazebo::common::SkeletonNode::GetRawTransforms ()`

Retrieve the raw transformations.

Returns

an array of transformations

10.141.4.16 `math::Matrix4 gazebo::common::SkeletonNode::GetTransform ()`

Get transform relative to parent.

10.141.4.17 `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)

10.141.4.18 `bool gazebo::common::SkeletonNode::IsJoint ()`

Is a joint query.

Returns

true if the skeleton type is a joint, false otherwise

10.141.4.19 `bool gazebo::common::SkeletonNode::IsRootNode ()`

Queries whether a node has no parent parent.

Returns

true if the node has no parent, false otherwise

10.141.4.20 `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
-----------------	------------------------------------	--

10.141.4.21 `void gazebo::common::SkeletonNode::SetHandle (unsigned int _h)`

Assign a handle number.

Parameters

<code>in</code>	<code><i>_h</i></code>	the handle
-----------------	------------------------	------------

10.141.4.22 `void gazebo::common::SkeletonNode::SetId (std::string _id)`

Change the id string.

Parameters

<code>in</code>	<code><i>_id</i></code>	the new id string
-----------------	-------------------------	-------------------

10.141.4.23 `void gazebo::common::SkeletonNode::SetInitialTransform (math::Matrix4 _tras)`

Sets the initial transformation.

Parameters

<code>in</code>	<code><i>_tras</i></code>	the transformation matrix
-----------------	---------------------------	---------------------------

10.141.4.24 `void gazebo::common::SkeletonNode::SetInverseBindTransform (math::Matrix4 _invBM)`

Assign the inverse of the bind pose skeletal transform.

Parameters

<code>in</code>	<code><i>_invBM</i></code>	the transform
-----------------	----------------------------	---------------

10.141.4.25 `void gazebo::common::SkeletonNode::SetModelTransform (math::Matrix4 _trans, bool _updateChildren = true)`

Set the model transformation.

Parameters

<code>in</code>	<code><i>_trans</i></code>	the transformation
<code>in</code>	<code><i>_updateChildren</i></code>	when true the UpdateChildrenTransforms operation is performed

10.141.4.26 `void gazebo::common::SkeletonNode::SetName (std::string _name)`

Change the name.

Parameters

<code>in</code>	<code><i>_name</i></code>	the new name
-----------------	---------------------------	--------------

10.141.4.27 `void gazebo::common::SkeletonNode::SetParent (SkeletonNode * _parent)`

Set the parent node.

Parameters

<code>in</code>	<code><i>_parent</i></code>	the new parent
-----------------	-----------------------------	----------------

10.141.4.28 `void gazebo::common::SkeletonNode::SetTransform (math::Matrix4 _trans, bool _updateChildren = true)`

Set a transformation.

Parameters

<code>in</code>	<code>_trans</code>	the transformation
<code>in</code>	<code>_updateChildren</code>	when true the UpdateChildrenTransforms operation is performed

10.141.4.29 `void gazebo::common::SkeletonNode::SetType (SkeletonNodeType _type)`

Change the skeleton node type.

Parameters

<code>in</code>	<code>_type</code>	the new type
-----------------	--------------------	--------------

10.141.4.30 `void gazebo::common::SkeletonNode::UpdateChildrenTransforms ()`

Apply model transformations in order for each node in the tree.

10.141.5 Member Data Documentation

10.141.5.1 `std::vector<SkeletonNode*> gazebo::common::SkeletonNode::children` [protected]

the children nodes

10.141.5.2 `unsigned int gazebo::common::SkeletonNode::handle` [protected]

handle index number

10.141.5.3 `std::string gazebo::common::SkeletonNode::id` [protected]

a string identifier

10.141.5.4 `math::Matrix4 gazebo::common::SkeletonNode::initialTransform` [protected]

the initial transformation

10.141.5.5 `math::Matrix4 gazebo::common::SkeletonNode::invBindTransform` [protected]

the inverse of the bind pose skeletal transform

10.141.5.6 `math::Matrix4 gazebo::common::SkeletonNode::modelTransform` [protected]

the model transformation

10.141.5.7 `std::string gazebo::common::SkeletonNode::name` [protected]

the name of the skeletal node

10.141.5.8 `SkeletonNode* gazebo::common::SkeletonNode::parent` [protected]

the parent node

10.141.5.9 `std::vector<NodeTransform> gazebo::common::SkeletonNode::rawTransforms` [protected]

the raw transformation

10.141.5.10 `math::Matrix4 gazebo::common::SkeletonNode::transform` [protected]

the transform

10.141.5.11 `SkeletonNodeType gazebo::common::SkeletonNode::type` [protected]

the type fo node

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

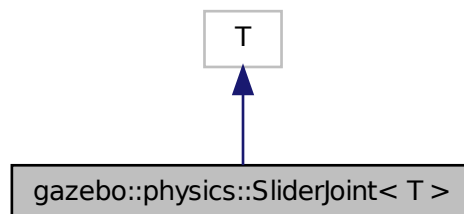
- **Skeleton.hh**

10.142 gazebo::physics::SliderJoint< T > Class Template Reference

A slider joint.

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

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



Public Member Functions

- **SliderJoint** (**BasePtr** _parent)

Constructor.

- virtual `~SliderJoint ()`

Destructor.

- virtual `math::Vector3 GetAnchor (int _index) const`

Get the anchor.

- virtual unsigned int `GetAngleCount () const`

- virtual void `Load (sdf::ElementPtr _sdf)`

Load a `SliderJoint` (p. 692).

- virtual void `SetAnchor (int _index, const math::Vector3 &_anchor)`

Set the anchor.

Protected Attributes

- `math::Vector3 fakeAnchor`

The anchor value is not used internally.

10.142.1 Detailed Description

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

A slider joint.

10.142.2 Constructor & Destructor Documentation

10.142.2.1 `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`.

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

Destructor.

10.142.3 Member Function Documentation

10.142.3.1 `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.

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

10.142.3.3 `template<class T > virtual void gazebo::physics::SliderJoint< T >::Load (sdf::ElementPtr _sdf)`
`[inline], [virtual]`

Load a **SliderJoint** (p. 692).

Parameters

in	<code>_sdf</code>	SDF values to load from
----	-------------------	-------------------------

10.142.3.4 `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.

10.142.4 Member Data Documentation

10.142.4.1 `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:

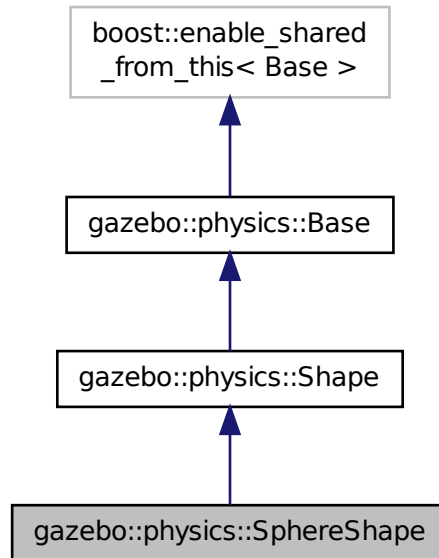
- **SliderJoint.hh**

10.143 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.
- virtual void **GetInertial** (double _mass, **InertialPtr** _inertial) const **GAZEBO_DEPRECATED**
Deprecated.
- virtual double **GetMass** (double _density) const **GAZEBO_DEPRECATED**
Deprecated.
- 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

10.143.1 Detailed Description

Sphere collision shape.

10.143.2 Constructor & Destructor Documentation

10.143.2.1 `gazebo::physics::SphereShape::SphereShape (CollisionPtr _parent)` [explicit]

Constructor.

Parameters

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

10.143.2.2 `virtual gazebo::physics::SphereShape::~~SphereShape ()` [virtual]

Destructor.

10.143.3 Member Function Documentation

10.143.3.1 `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. 669).

10.143.3.2 `virtual void gazebo::physics::SphereShape::GetInertial (double _mass, InertialPtr _inertial) const` [virtual]

Deprecated.

Reimplemented from `gazebo::physics::Shape` (p. 670).

10.143.3.3 `virtual double gazebo::physics::SphereShape::GetMass (double _density) const` [virtual]

Deprecated.

Reimplemented from `gazebo::physics::Shape` (p. 670).

10.143.3.4 `double gazebo::physics::SphereShape::GetRadius () const`

Get the sphere's radius.

Returns

Radius of the sphere.

10.143.3.5 virtual void gazebo::physics::SphereShape::Init () [virtual]

Initialize the sphere.

Implements **gazebo::physics::Shape** (p. 670).

10.143.3.6 virtual void gazebo::physics::SphereShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]

Process a geometry message.

Parameters

in	_msg	The message to set values from.
----	------	---------------------------------

Implements **gazebo::physics::Shape** (p. 670).

10.143.3.7 virtual void gazebo::physics::SphereShape::SetRadius (double _radius) [virtual]

Set the size.

Parameters

in	_radius	Radius of the sphere.
----	---------	-----------------------

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

- **SphereShape.hh**

10.144 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.

10.144.1 Detailed Description

Splines.

10.144.2 Constructor & Destructor Documentation

10.144.2.1 gazebo::math::Spline::Spline ()

constructor

10.144.2.2 gazebo::math::Spline::~~Spline ()

destructor

10.144.3 Member Function Documentation

10.144.3.1 void gazebo::math::Spline::AddPoint (const Vector3 & _pt)

Adds a control point to the end of the spline.

Parameters

in	_pt	point to add
----	-----	--------------

10.144.3.2 void gazebo::math::Spline::Clear ()

Clears all the points in the spline.

10.144.3.3 Vector3 gazebo::math::Spline::GetPoint (unsigned int _index) const

Gets the detail of one of the control points of the spline.

Parameters

in	_index	the control point index
----	--------	-------------------------

Returns

the control point, or [0,0,0] and a message on the error stream

10.144.3.4 unsigned int gazebo::math::Spline::GetPointCount () const

Gets the number of control points in the spline.

Returns

the count

10.144.3.5 Vector3 gazebo::math::Spline::GetTangent (unsigned int _index) const

Get the tangent value for a point.

Parameters

in	_index	the control point index
----	--------	-------------------------

10.144.3.6 double gazebo::math::Spline::GetTension () const

Get the tension value.

Returns

The value of the tension, which is between 0.0 and 1.0

10.144.3.7 Vector3 gazebo::math::Spline::Interpolate (double *_t*) const

Returns an interpolated point based on a parametric value over the whole series.

Parameters

<i>in</i>	<i>_t</i>	parameter (range 0 to 1)
-----------	-----------	--------------------------

10.144.3.8 Vector3 gazebo::math::Spline::Interpolate (unsigned int *_fromIndex*, double *_t*) const

Interpolates a single segment of the spline given a parametric value.

Parameters

<i>in</i>	<i>_fromIndex</i>	The point index to treat as t = 0. fromIndex + 1 is deemed to be t = 1
<i>in</i>	<i>_t</i>	Parametric value

10.144.3.9 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.

10.144.3.10 void gazebo::math::Spline::SetAutoCalculate (bool *_autoCalc*)

Tells the spline whether it should automatically calculate tangents on demand as points are added.

Remarks

The spline calculates tangents at each point automatically based on the input points. Normally it does this every time a point changes. However, if you have a lot of points to add in one go, you probably don't want to incur this overhead and would prefer to defer the calculation until you are finished setting all the points. You can do this by calling this method with a parameter of 'false'. Just remember to manually call the `recalcTangents` method when you are done.

Parameters

<i>in</i>	<i>_autoCalc</i>	If true, tangents are calculated for you whenever a point changes. If false, you must call <code>recalcTangents</code> to recalculate them when it best suits.
-----------	------------------	--

10.144.3.11 void gazebo::math::Spline::SetTension (double *_t*)

Set the tension parameter.

A value of 0 = Catmull-Rom spline.

Parameters

<i>in</i>	<i>_t</i>	Tension value between 0.0 and 1.0
-----------	-----------	-----------------------------------

10.144.3.12 void gazebo::math::Spline::UpdatePoint (unsigned int *_index*, const Vector3 & *_value*)

Updates a single point in the spline.

Remarks

an error to the error stream is printed when the index is out of bounds

Parameters

<i>in</i>	<i>_index</i>	the control point index
<i>in</i>	<i>_value</i>	the new position

10.144.4 Member Data Documentation

10.144.4.1 bool gazebo::math::Spline::autoCalc [protected]

when true, the tangents are recalculated when the control point change

10.144.4.2 Matrix4 gazebo::math::Spline::coeffs [protected]

Matrix of coefficients.

10.144.4.3 std::vector<Vector3> gazebo::math::Spline::points [protected]

control points

10.144.4.4 std::vector<Vector3> gazebo::math::Spline::tangents [protected]

tangents

10.144.4.5 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:

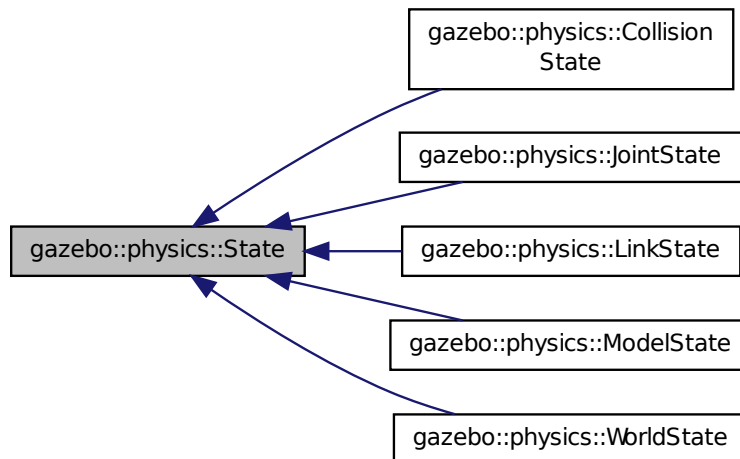
- **Spline.hh**

10.145 gazebo::physics::State Class Reference

State (p. 702) 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. 702).*
- **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. 702).*

Protected Attributes

- std::string **name**
*Name associated with this **State** (p. 702).*
- common::Time **realTime**
- common::Time **simTime**
- common::Time **wallTime**
Times for the state data.

10.145.1 Detailed Description

State (p. 702) of an entity.

This is the base class for all **State** (p. 702) information.

10.145.2 Constructor & Destructor Documentation

10.145.2.1 gazebo::physics::State::State ()

Default constructor.

10.145.2.2 gazebo::physics::State::State (const std::string & _name, const common::Time & _realTime, const common::Time & _simTime)

Constructor.

Construct a **State** (p. 702) object using some basic information.

Parameters

<code>_name</code>	Name associated with the State (p. 702) information. This is typically the name of an Entity (p. 265). <code>_realTime</code> Clock time since simulation started.
<code>_simTime</code>	Simulation time associated with this State (p. 702) info.

10.145.2.3 virtual gazebo::physics::State::~~State () [virtual]

Destructor.

10.145.3 Member Function Documentation

10.145.3.1 std::string gazebo::physics::State::GetName () const

Get the name associated with this **State** (p. 702).

Returns

Name associated with this state information. Typically a name of an **Entity** (p. 265).

10.145.3.2 `common::Time gazebo::physics::State::GetRealTime () const`

Get the real time when this state was generated.

Returns

Clock time since simulation was stated.

10.145.3.3 `common::Time gazebo::physics::State::GetSimTime () const`

Get the sim time when this state was generated.

Returns

Simulation time when the data was recorded.

10.145.3.4 `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. 702) data was recorded.

10.145.3.5 `virtual void gazebo::physics::State::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Populates the **State** (p. 702) information from data stored in an SDF::Element

Parameters

<code>_elem</code>	Pointer to the SDF::Element
--------------------	-----------------------------

Reimplemented in **gazebo::physics::ModelState** (p. 481), **gazebo::physics::LinkState** (p. 421), **gazebo::physics::WorldState** (p. 869), **gazebo::physics::CollisionState** (p. 192), and **gazebo::physics::JointState** (p. 384).

10.145.3.6 `State gazebo::physics::State::operator- (const State & _state) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_pt</code>	A state to subtract.
-----------------	------------------	----------------------

Returns

The resulting state.

10.145.3.7 State& gazebo::physics::State::operator= (const State & *_state*)

Assignment operator.

Parameters

<i>in</i>	<i>_state</i>	State (p. 702) value
-----------	---------------	-----------------------------

Returns

this

10.145.3.8 void gazebo::physics::State::SetName (const std::string & *_name*)

Set the name associated with this **State** (p. 702).

Parameters

<i>in</i>	<i>_name</i>	Name associated with this state information. Typically the name of an Entity (p. 265).
-----------	--------------	---

10.145.4 Member Data Documentation**10.145.4.1 std::string gazebo::physics::State::name** [protected]

Name associated with this **State** (p. 702).

10.145.4.2 common::Time gazebo::physics::State::realTime [protected]**10.145.4.3 common::Time gazebo::physics::State::simTime** [protected]**10.145.4.4 common::Time gazebo::physics::State::wallTime** [protected]

Times for the state data.

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

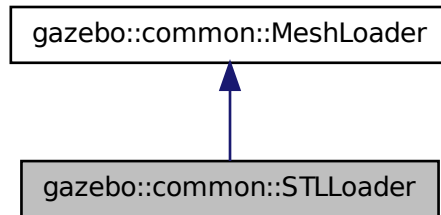
- **State.hh**

10.146 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.

10.146.1 Detailed Description

Class used to load STL mesh files.

10.146.2 Constructor & Destructor Documentation

10.146.2.1 gazebo::common::STLLoader::STLLoader ()

Constructor.

10.146.2.2 virtual gazebo::common::STLLoader::~~STLLoader () [virtual]

Destructor.

10.146.3 Member Function Documentation

10.146.3.1 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. 455).

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

- `STLloader.hh`

10.147 gazebo::common::SubMesh Class Reference

A child mesh.

```
#include <Mesh.hh>
```

Public Types

- enum `PrimitiveType` {
POINTS, **LINES**, **LINESTRIPS**, **TRIANGLES**,
TRIFANS, **TRISTRIPS** }
An enumeration of the geometric mesh primitives.

Public Member Functions

- `SubMesh ()`
Constructor.
- `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 **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.

10.147.1 Detailed Description

A child mesh.

10.147.2 Member Enumeration Documentation

10.147.2.1 enum gazebo::common::SubMesh::PrimitiveType

An enumeration of the geometric mesh primitives.

Enumerator:

POINTS

LINES

LINESTRIPS

TRIANGLES

TRIFANS

TRISTRIPS

10.147.3 Constructor & Destructor Documentation

10.147.3.1 gazebo::common::SubMesh::SubMesh ()

Constructor.

10.147.3.2 virtual gazebo::common::SubMesh::~~SubMesh () [virtual]

Destructor.

10.147.4 Member Function Documentation

10.147.4.1 void gazebo::common::SubMesh::AddIndex (unsigned int _i)

Add an index to the mesh.

Parameters

in	<code>_i</code>	the new vertex index
----	-----------------	----------------------

10.147.4.2 `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)

10.147.4.3 `void gazebo::common::SubMesh::AddNormal (const math::Vector3 & _n)`

Add a normal to the mesh.

Parameters

in	<code>_n</code>	the normal
----	-----------------	------------

10.147.4.4 `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

10.147.4.5 `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

10.147.4.6 `void gazebo::common::SubMesh::AddVertex (const math::Vector3 & _v)`

Add a vertex to the mesh.

Parameters

in	<code>_v</code>	the new position
----	-----------------	------------------

10.147.4.7 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

10.147.4.8 void gazebo::common::SubMesh::CopyNormals (const std::vector< math::Vector3 > & *_norms*)

Copy normals from a vector.

Parameters

in	<i>_norms</i>	to copy from
----	---------------	--------------

10.147.4.9 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
----	---------------	---------------------------

10.147.4.10 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>	

10.147.4.11 void gazebo::common::SubMesh::GenSphericalTexCoord (const math::Vector3 & *_center*)

Generate texture coordinates using spherical projection from center.

Parameters

in	<i>_center</i>	
----	----------------	--

10.147.4.12 unsigned int gazebo::common::SubMesh::GetIndex (unsigned int *_i*) const

Get an index.

Parameters

in	<i>_i</i>	
----	-----------	--

10.147.4.13 `unsigned int gazebo::common::SubMesh::GetIndexCount () const`

Return the number of indices.

10.147.4.14 `unsigned int gazebo::common::SubMesh::GetMaterialIndex () const`

Get the material index.

10.147.4.15 `math::Vector3 gazebo::common::SubMesh::GetMax () const`

Get the maximum X, Y, Z values.

Returns

10.147.4.16 `unsigned int gazebo::common::SubMesh::GetMaxIndex () const`

Get the highest index value.

10.147.4.17 `math::Vector3 gazebo::common::SubMesh::GetMin () const`

Get the minimum X, Y, Z values.

Returns

10.147.4.18 `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
----	-----------	-----------------------------

10.147.4.19 `unsigned int gazebo::common::SubMesh::GetNodeAssignmentsCount () const`

Return the number of vertex - skeleton node assignments.

10.147.4.20 `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

10.147.4.21 `unsigned int gazebo::common::SubMesh::GetNormalCount () const`

Return the number of normals.

10.147.4.22 `PrimitiveType gazebo::common::SubMesh::GetPrimitiveType () const`

Get the primitive type.

Returns

the primitive type

10.147.4.23 `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

10.147.4.24 `unsigned int gazebo::common::SubMesh::GetTexCoordCount () const`

Return the number of texture coordinates.

10.147.4.25 `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

10.147.4.26 `unsigned int gazebo::common::SubMesh::GetVertexCount () const`

Return the number of vertices.

10.147.4.27 `unsigned int gazebo::common::SubMesh::GetVertexIndex (const math::Vector3 & _v) const`

Get the index of the vertex.

Parameters

<code>in</code>	<code>_v</code>	
-----------------	-----------------	--

10.147.4.28 `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>	
-----------------	-----------------	--

10.147.4.29 `void gazebo::common::SubMesh::RecalculateNormals ()`

Recalculate all the normals.

10.147.4.30 `void gazebo::common::SubMesh::Scale (double _factor)`

Scale all vertices by `_factor`.

Parameters

<code>in</code>	<code>_factor</code>	Scaling factor
-----------------	----------------------	----------------

10.147.4.31 `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
-----------------	---------------------	---------------------------

10.147.4.32 `void gazebo::common::SubMesh::SetMaterialIndex (unsigned int _index)`

Set the material index.

Relates to the parent mesh material list

Parameters

in	<i>_index</i>	
----	---------------	--

10.147.4.33 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

10.147.4.34 void gazebo::common::SubMesh::SetNormalCount (unsigned int *_count*)

Resize the normal array.

Parameters

in	<i>_count</i>	the new size of the array
----	---------------	---------------------------

10.147.4.35 void gazebo::common::SubMesh::SetPrimitiveType (PrimitiveType *_type*)

Set the primitive type.

Parameters

in	<i>_type</i>	the type
----	--------------	----------

10.147.4.36 void gazebo::common::SubMesh::SetSubMeshCenter (math::Vector3 *_center*)

Reset mesh center to geometric center.

Parameters

in	<i>_center</i>	
----	----------------	--

10.147.4.37 void gazebo::common::SubMesh::SetTexCoord (unsigned int *_i*, const math::Vector2d & *_t*)

Set a tex coord.

Parameters

in	<i>_i</i>	
in	<i>_t</i>	

10.147.4.38 void gazebo::common::SubMesh::SetTexCoordCount (unsigned int *_count*)

Resize the texture coordinate array.

Parameters

in	<i>_count</i>	
----	---------------	--

10.147.4.39 void gazebo::common::SubMesh::SetVertex (unsigned int *_i*, const math::Vector3 & *_v*)

Set a vertex.

Parameters

in	<i>_i</i>	the index
in	<i>_v</i>	the position

10.147.4.40 void gazebo::common::SubMesh::SetVertexCount (unsigned int *_count*)

Resize the vertex array.

Parameters

in	<i>_count</i>	the new size of the array
----	---------------	---------------------------

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

- **Mesh.hh**

10.148 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.

10.148.1 Detailed Description

Options for a subscription.

10.148.2 Constructor & Destructor Documentation

10.148.2.1 gazebo::transport::SubscribeOptions::SubscribeOptions () [inline]

Constructor.

10.148.3 Member Function Documentation

10.148.3.1 bool gazebo::transport::SubscribeOptions::GetLatching () const [inline]

Are we latching?

Returns

true if we're latching the latest message, false otherwise

10.148.3.2 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

10.148.3.3 NodePtr gazebo::transport::SubscribeOptions::GetNode () const [inline]

Get the node we're subscribed to.

Returns

The associated node

10.148.3.4 std::string gazebo::transport::SubscribeOptions::GetTopic () const [inline]

Get the topic we're subscribed to.

Returns

The topic we're subscribed to

10.148.3.5 `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`.

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

- **SubscribeOptions.hh**

10.149 gazebo::transport::Subscriber Class Reference

A subscriber to a topic.

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

Public Member Functions

- **Subscriber** (const std::string &_topic, NodePtr _node)
Constructor.
- virtual ~**Subscriber** ()
Destructor.
- std::string **GetTopic** () const
Get the topic name.
- void **Unsubscribe** () const
Unsubscribe from the topic.

10.149.1 Detailed Description

A subscriber to a topic.

10.149.2 Constructor & Destructor Documentation

10.149.2.1 `gazebo::transport::Subscriber::Subscriber (const std::string & _topic, NodePtr _node)`

Constructor.

Parameters

<code>in</code>	<code>_topic</code>	The topic we're subscribing to
<code>in</code>	<code>_node</code>	The associated node

10.149.2.2 `virtual gazebo::transport::Subscriber::~~Subscriber () [virtual]`

Destructor.

10.149.3 Member Function Documentation

10.149.3.1 `std::string gazebo::transport::Subscriber::GetTopic () const`

Get the topic name.

Returns

The topic name

10.149.3.2 `void gazebo::transport::Subscriber::Unsubscribe () const`

Unsubscribe from the topic.

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

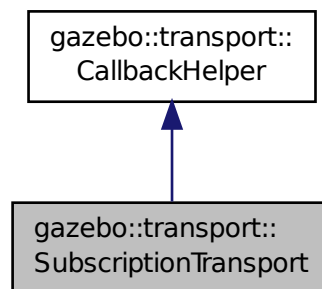
- **Subscriber.hh**

10.150 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

10.150.1 Detailed Description

transport/transport.hh

Handles sending data over the wire to remote subscribers

10.150.2 Constructor & Destructor Documentation

10.150.2.1 gazebo::transport::SubscriptionTransport::SubscriptionTransport ()

Constructor.

10.150.2.2 virtual gazebo::transport::SubscriptionTransport::~~SubscriptionTransport () [virtual]

Destructor.

10.150.3 Member Function Documentation

10.150.3.1 const ConnectionPtr& gazebo::transport::SubscriptionTransport::GetConnection () const

Get the connection we're using.

Returns

Pointer to the connection we're using

10.150.3.2 virtual bool gazebo::transport::SubscriptionTransport::HandleData (const std::string &_newdata) [virtual]

Output a message to a connection.

Parameters

<code>in</code>	<code>_newdata</code>	The message to be handled
-----------------	-----------------------	---------------------------

Returns

true if the message was handled successfully, false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 146).

10.150.3.3 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

10.150.3.4 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. 146).

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

- **SubscriptionTransport.hh**

10.151 gazebo::physics::SurfaceParams Class Reference

SurfaceParams (p. 721) 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.
- void **FillSurfaceMsg** (msgs::Surface &_msg) **GAZEBO_DEPRECATED**
Deprecated.
- 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. 725)) of the friction pyramid.*
- double **kd**
*spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 723) and **SurfaceParams::erp** (p. 724).*
- double **kp**
*spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 723) and **SurfaceParams::erp** (p. 724).*
- 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.

10.151.1 Detailed Description

SurfaceParams (p. 721) defines various Surface contact parameters.

These parameters defines the properties of a **physics::Contact** (p. 218) constraint.

10.151.2 Constructor & Destructor Documentation

10.151.2.1 gazebo::physics::SurfaceParams::SurfaceParams ()

Constructor.

10.151.2.2 virtual gazebo::physics::SurfaceParams::~~SurfaceParams () [virtual]

Destructor.

10.151.3 Member Function Documentation

10.151.3.1 void gazebo::physics::SurfaceParams::FillMsg (msgs::Surface & *_msg*)

Fill in a surface message.

Parameters

in	<i>_msg</i>	Message to fill with this object's values.
----	-------------	--

10.151.3.2 void gazebo::physics::SurfaceParams::FillSurfaceMsg (msgs::Surface & *_msg*)

Deprecated.

10.151.3.3 virtual void gazebo::physics::SurfaceParams::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the contact params.

Parameters

in	<i>_sdf</i>	SDF values to load from.
----	-------------	--------------------------

10.151.3.4 virtual void gazebo::physics::SurfaceParams::ProcessMsg (const msgs::Surface & *_msg*) [virtual]

10.151.4 Member Data Documentation

10.151.4.1 double gazebo::physics::SurfaceParams::bounce

bounce restitution coefficient [0,1], with 0 being inelastic, and 1 being perfectly elastic.

See Also

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.151.4.2 double gazebo::physics::SurfaceParams::bounceThreshold

minimum contact velocity for bounce to take effect, otherwise the collision is treated as an inelastic collision.

See Also

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.151.4.3 double gazebo::physics::SurfaceParams::cfm

Constraint Force Mixing parameter.

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_0 for more details.

10.151.4.4 `double gazebo::physics::SurfaceParams::erp`

Error Reduction Parameter.

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_0 for more details.

10.151.4.5 `math::Vector3 gazebo::physics::SurfaceParams::fdir1`

Primary friction direction for dry friction coefficient (**SurfaceParams::mu1** (p. 725)) of the friction pyramid.

If undefined, a vector constrained to be perpendicular to the contact normal in the global y-z plane is used.

See Also

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.151.4.6 `double gazebo::physics::SurfaceParams::kd`

spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 723) and **SurfaceParams::erp** (p. 724).

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_2 for more details.

10.151.4.7 `double gazebo::physics::SurfaceParams::kp`

spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 723) and **SurfaceParams::erp** (p. 724).

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_2 for more details.

10.151.4.8 `double gazebo::physics::SurfaceParams::maxVel`

Maximum interpenetration error correction velocity.

If set to 0, two objects interpenetrating each other will not be pushed apart.

See Also

See `dWroldSetContactMaxCorrectingVel` (http://www.ode.org/ode-latest-userguide.html#sec_5_2_0)

10.151.4.9 double gazebo::physics::SurfaceParams::minDepth

Minimum depth before ERP takes effect.

See Also

See `dWorldSetContactSurfaceLayer` (http://www.ode.org/ode-latest-userguide.html#sec_5-_2_0)

10.151.4.10 double gazebo::physics::SurfaceParams::mu1

Dry friction coefficient in the primary friction direction as defined by the friction pyramid.

This is `fdir1` if defined, otherwise, a vector constrained to be perpendicular to the contact normal in the global y-z plane is used.

See Also

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.151.4.11 double gazebo::physics::SurfaceParams::mu2

Dry friction coefficient in the second friction direction as defined by the friction pyramid.

This is `fdir1` if defined, otherwise, a vector constrained to be perpendicular to the contact normal in the global y-z plane is used.

See Also

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.151.4.12 double gazebo::physics::SurfaceParams::slip1

Artificial contact slip in the primary friction direction.

See Also

See `dContactSlip1` in http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.151.4.13 double gazebo::physics::SurfaceParams::slip2

Artificial contact slip in the secondary friction direction.

See Also

See `dContactSlip2` in http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

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

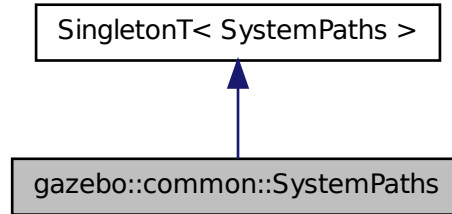
- **SurfaceParams.hh**

10.152 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. 729)*
- bool **modelPathsFromEnv**
*if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 729)*
- bool **ogrePathsFromEnv**
*if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 729)*
- bool **pluginPathsFromEnv**
*if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 729)*

Additional Inherited Members

10.152.1 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. 98) RenderSystem_GL.so et. al.
- SystemPaths::pluginPaths - plugin library paths for common::WorldPlugin

10.152.2 Member Function Documentation

10.152.2.1 void gazebo::common::SystemPaths::AddGazeboPaths (const std::string & *_path*)

Add colon delimited paths to Gazebo install.

Parameters

<i>in</i>	<i>_path</i>	the directory to add
-----------	--------------	----------------------

10.152.2.2 void gazebo::common::SystemPaths::AddOgrePaths (const std::string & *_path*)

Add colon delimited paths to ogre install.

Parameters

<i>in</i>	<i>_path</i>	the directory to add
-----------	--------------	----------------------

10.152.2.3 void gazebo::common::SystemPaths::AddPluginPaths (const std::string & *_path*)

Add colon delimited paths to plugins.

Parameters

in	<i>_path</i>	the directory to add
----	--------------	----------------------

10.152.2.4 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
----	----------------	-------------------

10.152.2.5 void gazebo::common::SystemPaths::ClearGazeboPaths ()

clear out SystemPaths::gazeboPaths

10.152.2.6 void gazebo::common::SystemPaths::ClearOgrePaths ()

clear out SystemPaths::ogrePaths

10.152.2.7 void gazebo::common::SystemPaths::ClearPluginPaths ()

clear out SystemPaths::pluginPaths

10.152.2.8 std::string gazebo::common::SystemPaths::FindFile (const std::string & *_filename*, bool *_searchLocalPath* = true)

Find a file in the gazebo paths.

Parameters

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

Returns

Returns full path name to file

10.152.2.9 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

10.152.2.10 `const std::list<std::string>& gazebo::common::SystemPaths::GetGazeboPaths ()`

Get the gazebo install paths.

Returns

a list of paths

10.152.2.11 `std::string gazebo::common::SystemPaths::GetLogPath () const`

Get the log path.

Returns

the path

10.152.2.12 `const std::list<std::string>& gazebo::common::SystemPaths::GetModelPaths ()`

Get the model paths.

Returns

a list of paths

10.152.2.13 `const std::list<std::string>& gazebo::common::SystemPaths::GetOgrePaths ()`

Get the ogre install paths.

Returns

a list of paths

10.152.2.14 `const std::list<std::string>& gazebo::common::SystemPaths::GetPluginPaths ()`

Get the plugin paths.

Returns

a list of paths

10.152.2.15 `std::string gazebo::common::SystemPaths::GetWorldPathExtension ()`

Returns the world path extension.

Returns

Right now, it just returns `"/worlds"`

10.152.3 Member Data Documentation

10.152.3.1 `bool gazebo::common::SystemPaths::gazeboPathsFromEnv`

if true, call `UpdateGazeboPaths()` within `GetGazeboPaths()` (p. 729)

10.152.3.2 `bool gazebo::common::SystemPaths::modelPathsFromEnv`

if true, call `UpdateGazeboPaths()` within `GetGazeboPaths()` (p. 729)

10.152.3.3 `bool gazebo::common::SystemPaths::ogrePathsFromEnv`

if true, call `UpdateOgrePaths()` within `GetOgrePaths()` (p. 729)

10.152.3.4 `bool gazebo::common::SystemPaths::pluginPathsFromEnv`

if true, call `UpdatePluginPaths()` within `GetPluginPaths()` (p. 729)

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

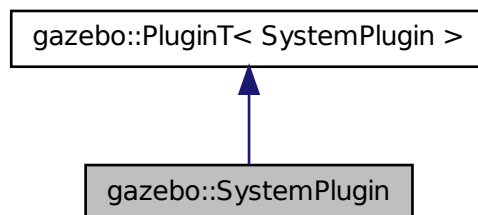
- `SystemPaths.hh`

10.153 gazebo::SystemPlugin Class Reference

A plugin loaded within the gzserver on startup.

```
#include <Plugin.hh>
```

Inheritance diagram for `gazebo::SystemPlugin`:



Public Member Functions

- **SystemPlugin** ()
Constructor.
- virtual **~SystemPlugin** ()
Destructor.
- virtual void **Init** ()
Initialize the plugin.
- virtual void **Load** (int _argc=0, char **_argv=NULL)=0
Load function.
- virtual void **Reset** ()
Override this method for custom plugin reset behavior.

Additional Inherited Members

10.153.1 Detailed Description

A plugin loaded within the gzserver on startup.

See [reference](#).

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

10.153.2 Constructor & Destructor Documentation

10.153.2.1 gazebo::SystemPlugin::SystemPlugin () [inline]

Constructor.

References `gazebo::SYSTEM_PLUGIN`, and `gazebo::PluginT< SystemPlugin >::type`.

10.153.2.2 virtual gazebo::SystemPlugin::~~SystemPlugin () [inline],[virtual]

Destructor.

10.153.3 Member Function Documentation

10.153.3.1 virtual void gazebo::SystemPlugin::Init () [inline],[virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.153.3.2 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.

10.153.3.3 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**

10.154 gazebo::common::Time Class Reference

A **Time** (p. 732) class, can be used to hold wall- or sim-time.

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

Public Member Functions

- **Time** ()
Constructors.
- **Time** (const **Time** &_time)
Copy constructor.
- **Time** (const struct timeval &_tv)
Constructor.
- **Time** (const struct timespec &_tv)
Constructor.
- **Time** (int32_t _sec, int32_t _nsec)
Constructor.
- **Time** (double _time)
Constructor.
- virtual ~**Time** ()
Destructor.
- double **Double** () const
Get the time as a double.
- float **Float** () const
Get the time as a float.
- bool **operator!=** (const struct timeval &_tv) const
Equal to operator.
- bool **operator!=** (const struct timespec &_tv) const
Equal to operator.
- bool **operator!=** (const **Time** &_time) const
Equal to operator.
- bool **operator!=** (double _time) const
Equal to operator.
- **Time operator*** (const struct timeval &_tv) const

- Multiplication operator.*

 - **Time operator*** (const struct timespec &_tv) const

Multiplication operator.
- **Time operator*** (const **Time** &_time) const

Multiplication operators.
- const **Time & operator*==** (const struct timeval &_tv)

Multiplication assignment operator.
- const **Time & operator*==** (const struct timespec &_tv)

Multiplication assignment operator.
- const **Time & operator*==** (const **Time** &_time)

Multiplication operators.
- **Time operator+** (const struct timeval &_tv) const

Addition operators.
- **Time operator+** (const struct timespec &_tv) const

Addition operators.
- **Time operator+** (const **Time** &_time) const

Addition operators.
- const **Time & operator+=** (const struct timeval &_tv)

Addition assignment operator.
- const **Time & operator+=** (const struct timespec &_tv)

Addition assignment operator.
- const **Time & operator+=** (const **Time** &_time)

Addition assignment operator.
- **Time operator-** (const struct timeval &_tv) const

Subtraction operator.
- **Time operator-** (const struct timespec &_tv) const

Subtraction operator.
- **Time operator-** (const **Time** &_time) const

Subtraction operator.
- const **Time & operator-=** (const struct timeval &_tv)

Subtraction assignment operator.
- const **Time & operator-=** (const struct timespec &_tv)

Subtraction assignment operator.
- const **Time & operator-=** (const **Time** &_time)

Subtraction assignment operator.
- **Time operator/** (const struct timeval &_tv) const

Division operator.
- **Time operator/** (const struct timespec &_tv) const

Division operator.
- **Time operator/** (const **Time** &_time) const

Division operator.
- const **Time & operator/=** (const struct timeval &_tv)

Division assignment operator.
- const **Time & operator/=** (const struct timespec &_tv)

Division assignment operator.
- const **Time & operator/=** (const **Time** &time)

Division assignment operator.

- bool **operator**< (const struct timeval &_tv) const
Less than operator.
- bool **operator**< (const struct timespec &_tv) const
Less than operator.
- bool **operator**< (const **Time** &_time) const
Less than operator.
- bool **operator**< (double _time) const
Less than operator.
- bool **operator**<= (const struct timeval &_tv) const
Less than or equal to operator.
- bool **operator**<= (const struct timespec &_tv) const
Less than or equal to operator.
- bool **operator**<= (const **Time** &_time) const
Less than or equal to operator.
- bool **operator**<= (double _time) const
Less than or equal to operator.
- **Time** & **operator**= (const struct timeval &_tv)
Assignment operator.
- **Time** & **operator**= (const struct timespec &_tv)
Assignment operator.
- **Time** & **operator**= (const **Time** &_time)
Assignment operator.
- bool **operator**== (const struct timeval &_tv) const
Equal to operator.
- bool **operator**== (const struct timespec &_tv) const
Equal to operator.
- bool **operator**== (const **Time** &_time) const
Equal to operator.
- bool **operator**== (double _time) const
Equal to operator.
- bool **operator**> (const struct timeval &_tv) const
Greater than operator.
- bool **operator**> (const struct timespec &_tv) const
Greater than operator.
- bool **operator**> (const **Time** &_time) const
Greater than operator.
- bool **operator**> (double _time) const
Greater than operator.
- bool **operator**>= (const struct timeval &_tv) const
Greater than or equal operator.
- bool **operator**>= (const struct timespec &_tv) const
Greater than or equal operator.
- bool **operator**>= (const **Time** &_time) const
Greater than or equal operator.
- bool **operator**>= (double _time) const
Greater than or equal operator.
- void **Set** (int32_t _sec, int32_t _nsec)

Set to sec and nsec.

- void **Set** (double _seconds)

Set to seconds.

- void **SetToWallTime** ()

Set the time to the wall time.

Static Public Member Functions

- static const **Time** & **GetWallTime** ()

Get the wall time.

- static 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.

10.154.1 Detailed Description

A **Time** (p. 732) class, can be used to hold wall- or sim-time.

stored as sec and nano-sec.

10.154.2 Constructor & Destructor Documentation

10.154.2.1 gazebo::common::Time::Time ()

Constructors.

10.154.2.2 gazebo::common::Time::Time (const Time & *time*)

Copy constructor.

Parameters

in	<i>time</i>	Time (p. 732) to copy
----	-------------	------------------------------

10.154.2.3 gazebo::common::Time::Time (const struct timeval & *_tv*)

Constructor.

Parameters

in	<i>_tv</i>	Time (p. 732) to initialize to
----	------------	---------------------------------------

10.154.2.4 gazebo::common::Time::Time (const struct timespec & *_tv*)

Constructor.

Parameters

in	<i>_tv</i>	Time (p. 732) to initialize to
----	------------	---------------------------------------

10.154.2.5 gazebo::common::Time::Time (int32_t *_sec*, int32_t *_nsec*)

Constructor.

Parameters

in	<i>_sec</i>	Seconds
in	<i>_nsec</i>	Nanoseconds

10.154.2.6 gazebo::common::Time::Time (double *time*)

Constructor.

Parameters

in	<i>time</i>	Time (p. 732) in double format sec.nsec
----	-------------	--

10.154.2.7 virtual gazebo::common::Time::~~Time () [virtual]

Destructor.

10.154.3 Member Function Documentation

10.154.3.1 `double gazebo::common::Time::Double () const`

Get the time as a double.

Returns

Time (p. 732) as a double in seconds

10.154.3.2 `float gazebo::common::Time::Float () const`

Get the time as a float.

Returns

Time (p. 732) as a float in seconds

10.154.3.3 `static const Time& gazebo::common::Time::GetWallTime () [static]`

Get the wall time.

Returns

the current time

10.154.3.4 `static double gazebo::common::Time::MicToNano (double _ms) [inline],[static]`

Convert microseconds to nanoseconds.

Parameters

<code><i>_ms</i></code>	microseconds
-------------------------	--------------

Returns

nanoseconds

10.154.3.5 `static double gazebo::common::Time::MilToNano (double _ms) [inline],[static]`

Convert milliseconds to nanoseconds.

Parameters

<code><i>in</i></code>	<code><i>_ms</i></code>	milliseconds
------------------------	-------------------------	--------------

Returns

nanoseconds

10.154.3.6 `static Time gazebo::common::Time::MSleep (unsigned int _ms) [static]`

Millisecond sleep.

Parameters

<code>in</code>	<code><i>_ms</i></code>	milliseconds
-----------------	-------------------------	--------------

10.154.3.7 `static Time gazebo::common::Time::NSleep (unsigned int _ns) [static]`

Nano sleep.

Parameters

<code>in</code>	<code><i>_ns</i></code>	nanoseconds
-----------------	-------------------------	-------------

10.154.3.8 `static Time gazebo::common::Time::NSleep (Time _time) [static]`

Nano sleep.

Parameters

<code>in</code>	<code><i>_time</i></code>	is a Time (p. 732)
-----------------	---------------------------	---------------------------

10.154.3.9 `bool gazebo::common::Time::operator!= (const struct timeval & _tv) const`

Equal to operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare to
-----------------	-------------------------	------------------------

Returns

true if values are the same, false otherwise

10.154.3.10 `bool gazebo::common::Time::operator!= (const struct timespec & _tv) const`

Equal to operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare to
-----------------	-------------------------	------------------------

Returns

true if values are the same, false otherwise

10.154.3.11 `bool gazebo::common::Time::operator!=(const Time & _time) const`

Equal to operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare to
-----------------	---------------------------	------------------------

Returns

true if values are the same, false otherwise

10.154.3.12 `bool gazebo::common::Time::operator!=(double _time) const`

Equal to operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare to
-----------------	---------------------------	------------------------

Returns

true if values are the same, false otherwise

10.154.3.13 `Time gazebo::common::Time::operator*(const struct timeval & _tv) const`

Multiplication operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	The scaling duration
-----------------	-------------------------	----------------------

Returns

Time (p. 732) instance

10.154.3.14 `Time gazebo::common::Time::operator*(const struct timespec & _tv) const`

Multiplication operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the scaling duration
-----------------	-------------------------	----------------------

Returns

Time (p. 732) instance

10.154.3.15 **Time** gazebo::common::Time::operator* (const Time & *_time*) const

Multiplication operators.

Parameters

<i>in</i>	<i>_time</i>	the scaling factor
-----------	--------------	--------------------

Returns

a scaled **Time** (p. 732) instance

10.154.3.16 **const Time&** gazebo::common::Time::operator*= (const struct timeval & *_tv*)

Multiplication assignment operator.

Parameters

<i>in</i>	<i>_tv</i>	the scaling duration
-----------	------------	----------------------

Returns

a reference to this instance

10.154.3.17 **const Time&** gazebo::common::Time::operator*= (const struct timespec & *_tv*)

Multiplication assignment operator.

Parameters

<i>in</i>	<i>_tv</i>	the scaling duration
-----------	------------	----------------------

Returns

a reference to this instance

10.154.3.18 **const Time&** gazebo::common::Time::operator*= (const Time & *_time*)

Multiplication operators.

Parameters

<i>in</i>	<i>_time</i>	scale factor
-----------	--------------	--------------

Returns

a scaled **Time** (p. 732) instance

10.154.3.19 Time gazebo::common::Time::operator+ (const struct timeval & *_tv*) const

Addition operators.

Parameters

<i>in</i>	<i>_tv</i>	the time to add
-----------	------------	-----------------

Returns

a **Time** (p. 732) instance

10.154.3.20 Time gazebo::common::Time::operator+ (const struct timespec & *_tv*) const

Addition operators.

Parameters

<i>in</i>	<i>_tv</i>	the time to add
-----------	------------	-----------------

Returns

a **Time** (p. 732) instance

10.154.3.21 Time gazebo::common::Time::operator+ (const Time & *_time*) const

Addition operators.

Parameters

<i>in</i>	<i>_time</i>	The time to add
-----------	--------------	-----------------

Returns

a **Time** (p. 732) instance

10.154.3.22 const Time& gazebo::common::Time::operator+= (const struct timeval & *_tv*)

Addition assignment operator.

Parameters

<i>in</i>	<i>_tv</i>	the time to add
-----------	------------	-----------------

Returns

a reference to this instance

10.154.3.23 `const Time& gazebo::common::Time::operator+=(const struct timespec & _tv)`

Addition assignment operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to add
-----------------	------------------	-----------------

Returns

a reference to this instance

10.154.3.24 `const Time& gazebo::common::Time::operator+=(const Time & _time)`

Addition assignemtn operator.

Parameters

<code>in</code>	<code>_time</code>	The time to add
-----------------	--------------------	-----------------

Returns

a **Time** (p. 732) instance

10.154.3.25 `Time gazebo::common::Time::operator- (const struct timeval & _tv) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_tv</code>	The time to subtract
-----------------	------------------	----------------------

Returns

a **Time** (p. 732) instance

10.154.3.26 `Time gazebo::common::Time::operator- (const struct timespec & _tv) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_tv</code>	The time to subtract
-----------------	------------------	----------------------

Returns

a **Time** (p. 732) instance

10.154.3.27 Time gazebo::common::Time::operator- (const Time & *_time*) const

Subtraction operator.

Parameters

<i>in</i>	<i>_time</i>	The time to subtract
-----------	--------------	----------------------

Returns

a **Time** (p. 732) instance

10.154.3.28 const Time& gazebo::common::Time::operator-= (const struct timeval & *_tv*)

Subtraction assignment operator.

Parameters

<i>in</i>	<i>_tv</i>	The time to subtract
-----------	------------	----------------------

Returns

a **Time** (p. 732) instance

10.154.3.29 const Time& gazebo::common::Time::operator-= (const struct timespec & *_tv*)

Subtraction assignment operator.

Parameters

<i>in</i>	<i>_tv</i>	The time to subtract
-----------	------------	----------------------

Returns

a **Time** (p. 732) instance

10.154.3.30 const Time& gazebo::common::Time::operator-= (const Time & *_time*)

Subtraction assignment operator.

Parameters

<i>in</i>	<i>_time</i>	The time to subtract
-----------	--------------	----------------------

Returns

a reference to this instance

10.154.3.31 **Time** gazebo::common::Time::operator/ (const struct timeval & *_tv*) const

Division operator.

Parameters

<i>in</i>	<i>_tv</i>	a timeval divisor
-----------	------------	-------------------

Returns

a **Time** (p. 732) instance

10.154.3.32 **Time** gazebo::common::Time::operator/ (const struct timespec & *_tv*) const

Division operator.

Parameters

<i>in</i>	<i>_tv</i>	a timespec divisor
-----------	------------	--------------------

Returns

a **Time** (p. 732) instance

10.154.3.33 **Time** gazebo::common::Time::operator/ (const Time & *_time*) const

Division operator.

Parameters

<i>in</i>	<i>_time</i>	the divisor
-----------	--------------	-------------

Returns

a **Time** (p. 732) instance

10.154.3.34 **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. 732) instance

10.154.3.35 `const Time& gazebo::common::Time::operator/= (const struct timespec & _tv)`

Division assignment operator.

Parameters

<code>in</code>	<code>_tv</code>	a divisor
-----------------	------------------	-----------

Returns

a **Time** (p. 732) instance

10.154.3.36 `const Time& gazebo::common::Time::operator/= (const Time & time)`

Division assignment operator.

Parameters

<code>in</code>	<code>time</code>	the divisor
-----------------	-------------------	-------------

Returns

a **Time** (p. 732) instance

10.154.3.37 `bool gazebo::common::Time::operator< (const struct timeval & _tv) const`

Less than operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare with
-----------------	------------------	--------------------------

Returns

true if tv is shorter than this, false otherwise

10.154.3.38 `bool gazebo::common::Time::operator< (const struct timespec & _tv) const`

Less than operator.

Parameters

<code>in</code>	<code>_tv</code>	the time to compare with
-----------------	------------------	--------------------------

Returns

true if tv is shorter than this, false otherwise

10.154.3.39 `bool gazebo::common::Time::operator< (const Time & _time) const`

Less than operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is shorter than this, false otherwise

10.154.3.40 `bool gazebo::common::Time::operator< (double _time) const`

Less than operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is shorter than this, false otherwise

10.154.3.41 `bool gazebo::common::Time::operator<= (const struct timeval & _tv) const`

Less than or equal to operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if tv is shorter than or equal to this, false otherwise

10.154.3.42 `bool gazebo::common::Time::operator<= (const struct timespec & _tv) const`

Less than or equal to operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if tv is shorter than or equal to this, false otherwise

10.154.3.43 `bool gazebo::common::Time::operator<= (const Time & _time) const`

Less than or equal to operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is shorter than or equal to this, false otherwise

10.154.3.44 `bool gazebo::common::Time::operator<= (double _time) const`

Less than or equal to operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is shorter than or equal to this, false otherwise

10.154.3.45 `Time& gazebo::common::Time::operator= (const struct timeval & _tv)`

Assignment operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the new time
-----------------	-------------------------	--------------

Returns

a reference to this instance

10.154.3.46 `Time& gazebo::common::Time::operator= (const struct timespec & _tv)`

Assignment operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the new time
-----------------	-------------------------	--------------

Returns

a reference to this instance

10.154.3.47 `Time& gazebo::common::Time::operator= (const Time & _time)`

Assignment operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the new time
-----------------	---------------------------	--------------

Returns

a reference to this instance

10.154.3.48 `bool gazebo::common::Time::operator== (const struct timeval & _tv) const`

Equal to operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare to
-----------------	-------------------------	------------------------

Returns

true if values are the same, false otherwise

10.154.3.49 `bool gazebo::common::Time::operator== (const struct timespec & _tv) const`

Equal to operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare to
-----------------	-------------------------	------------------------

Returns

true if values are the same, false otherwise

10.154.3.50 `bool gazebo::common::Time::operator== (const Time & _time) const`

Equal to operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare to
-----------------	---------------------------	------------------------

Returns

true if values are the same, false otherwise

10.154.3.51 `bool gazebo::common::Time::operator==(double _time) const`

Equal to operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare to
-----------------	---------------------------	------------------------

Returns

true if values are the same, false otherwise

10.154.3.52 `bool gazebo::common::Time::operator>(const struct timeval & _tv) const`

Greater than operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.154.3.53 `bool gazebo::common::Time::operator>(const struct timespec & _tv) const`

Greater than operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.154.3.54 `bool gazebo::common::Time::operator>(const Time & _time) const`

Greater than operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.154.3.55 `bool gazebo::common::Time::operator> (double _time) const`

Greater than operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.154.3.56 `bool gazebo::common::Time::operator>= (const struct timeval & _tv) const`

Greater than or equal operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if tv is greater than or equal to this, false otherwise

10.154.3.57 `bool gazebo::common::Time::operator>= (const struct timespec & _tv) const`

Greater than or equal operator.

Parameters

<code>in</code>	<code><i>_tv</i></code>	the time to compare with
-----------------	-------------------------	--------------------------

Returns

true if tv is greater than or equal to this, false otherwise

10.154.3.58 `bool gazebo::common::Time::operator>= (const Time & _time) const`

Greater than or equal operator.

Parameters

<code>in</code>	<code><i>_time</i></code>	the time to compare with
-----------------	---------------------------	--------------------------

Returns

true if time is greater than or equal to this, false otherwise

10.154.3.59 `bool gazebo::common::Time::operator>= (double _time) const`

Greater than or equal operator.

Parameters

<code>in</code>	<code>_time</code>	the time to compare with
-----------------	--------------------	--------------------------

Returns

true if time is greater than or equal to this, false otherwise

10.154.3.60 `static double gazebo::common::Time::SecToNano (double _sec) [inline],[static]`

Convert seconds to nanoseconds.

Parameters

<code>in</code>	<code>_sec</code>	duration in seconds
-----------------	-------------------	---------------------

Returns

nanoseconds

10.154.3.61 `void gazebo::common::Time::Set (int32_t _sec, int32_t _nsec)`

Set to sec and nsec.

Parameters

<code>in</code>	<code>_sec</code>	Seconds
<code>in</code>	<code>_nsec</code>	Nanoseconds

10.154.3.62 `void gazebo::common::Time::Set (double _seconds)`

Set to seconds.

Parameters

<code>in</code>	<code>_seconds</code>	Number of seconds
-----------------	-----------------------	-------------------

10.154.3.63 `void gazebo::common::Time::SetToWallTime ()`

Set the time to the wall time.

10.154.4 Friends And Related Function Documentation

10.154.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::common::Time & _time)` [*friend*]

Stream insertion operator.

Parameters

<code>in</code>	<code><i>_out</i></code>	the output stream
<code>in</code>	<code><i>_time</i></code>	time to write to the stream

Returns

the output stream

10.154.4.2 `std::istream& operator>> (std::istream & _in, gazebo::common::Time & _time)` [*friend*]

Stream extraction operator.

Parameters

<code>in</code>	<code><i>_in</i></code>	the input stream
<code>in</code>	<code><i>_time</i></code>	time to read from to the stream

Returns

the input stream

10.154.5 Member Data Documentation

10.154.5.1 `int32_t gazebo::common::Time::nsec`

Microseconds.

10.154.5.2 `int32_t gazebo::common::Time::sec`

Seconds.

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

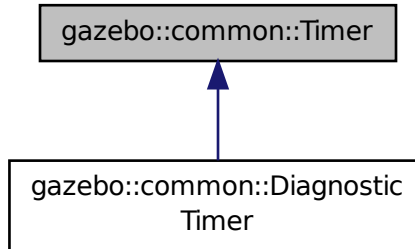
- **Time.hh**

10.155 gazebo::common::Timer Class Reference

A timer class, used to time things in real world walltime.

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

Inheritance diagram for gazebo::common::Timer:



Public Member Functions

- **Timer** ()
Constructor.
- virtual **~Timer** ()
Destructor.
- **Time GetElapsed** () const
Get the elapsed time.
- void **Start** ()
Start the timer.

Friends

- `std::ostream & operator<< (std::ostream &out, const gazebo::common::Timer &t)`
stream operator friendly

10.155.1 Detailed Description

A timer class, used to time things in real world walltime.

10.155.2 Constructor & Destructor Documentation

10.155.2.1 gazebo::common::Timer::Timer ()

Constructor.

10.155.2.2 virtual gazebo::common::Timer::~~Timer () [virtual]

Destructor.

10.155.3 Member Function Documentation

10.155.3.1 Time `gazebo::common::Timer::GetElapsed ()` const

Get the elapsed time.

Returns

The time

10.155.3.2 void `gazebo::common::Timer::Start ()`

Start the timer.

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

10.155.4 Friends And Related Function Documentation

10.155.4.1 `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:

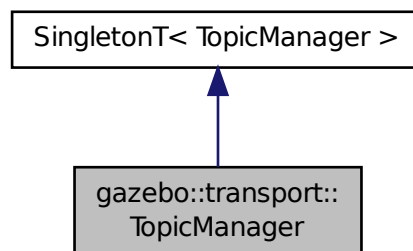
- **Timer.hh**

10.156 gazebo::transport::TopicManager Class Reference

Manages topics and their subscriptions.

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

Inheritance diagram for `gazebo::transport::TopicManager`:



Public Types

- typedef std::map< std::string, std::list< **NodePtr** > > **SubNodeMap**
A map of string->list of **Node** (p. 499) 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, bool _latch)
Advertise on a topic.
- void **ClearBuffers** ()
Clear all buffers.
- void **ConnectPubToSub** (const std::string &_topic, const **SubscriptionTransportPtr** &_sublink)
Connection (p. 207) a local **Publisher** (p. 578) to a remote **Subscriber** (p. 718).
- 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. 718) to a remote **Publisher** (p. 578).
- void **DisconnectPubFromSub** (const std::string &_topic, const std::string &_host, unsigned int _port)
Disconnect a local publisher from a remote subscriber.
- void **DisconnectSubFromPub** (const std::string &_topic, const std::string &_host, unsigned int _port)
Disconnect all local subscribers from a remote publisher.
- **PublicationPtr FindPublication** (const std::string &_topic)
Find a publication object by topic.
- void **Fini** ()
Finalize the manager.
- void **GetTopicNamespaces** (std::list< std::string > &_namespaces)
Get all the topic namespaces.
- void **Init** ()
Initialize the manager.
- bool **IsAdvertised** (const std::string &_topic)
Has the topic been advertised?
- void **PauseIncoming** (bool _pause)
Pause or unpaue processing of incoming messages.
- void **ProcessNodes** (bool _onlyOut=false)
Process all nodes under management.
- void **Publish** (const std::string &_topic, 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

10.156.1 Detailed Description

Manages topics and their subscriptions.

10.156.2 Member Typedef Documentation

10.156.2.1 `typedef std::map<std::string, std::list<NodePtr> > gazebo::transport::TopicManager::SubNodeMap`

A map of string->list of **Node** (p. 499) pointers.

10.156.3 Member Function Documentation

10.156.3.1 `void gazebo::transport::TopicManager::AddNode (NodePtr _node)`

Add a node to the manager.

Parameters

<code>in, out</code>	<code>_node</code>	The node to be added
----------------------	--------------------	----------------------

10.156.3.2 `template<typename M > PublisherPtr gazebo::transport::TopicManager::Advertise (const std::string & _topic, unsigned int _queueLimit, bool _latch) [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
<code>in</code>	<code>_latch</code>	If true, latch the latest message; if false, don't latch

Returns

Pointer to the newly created **Publisher** (p. 578)

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()`.

10.156.3.3 void gazebo::transport::TopicManager::ClearBuffers ()

Clear all buffers.

10.156.3.4 void gazebo::transport::TopicManager::ConnectPubToSub (const std::string & *_topic*, const SubscriptionTransportPtr & *_sublink*)

Connection (p. 207) a local **Publisher** (p. 578) to a remote **Subscriber** (p. 718).

Parameters

in	<i>_topic</i>	The topic to use
in	<i>_sublink</i>	The subscription transport object to use

10.156.3.5 void gazebo::transport::TopicManager::ConnectSubscribers (const std::string & *_topic*)

Connect all subscribers on a topic to known publishers.

Parameters

in	<i>_topic</i>	The topic to be connected
----	---------------	---------------------------

10.156.3.6 void gazebo::transport::TopicManager::ConnectSubToPub (const msgs::Publish & *_pub*)

Connect a local **Subscriber** (p. 718) to a remote **Publisher** (p. 578).

Parameters

in	<i>_pub</i>	The publish object to use
----	-------------	---------------------------

10.156.3.7 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

in	<i>_topic</i>	The topic to be disconnected
in	<i>_host</i>	The host to be disconnected
in	<i>_port</i>	The port to be disconnected

10.156.3.8 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

in	<code>_topic</code>	The topic to be disconnected
in	<code>_host</code>	The host to be disconnected
in	<code>_port</code>	The port to be disconnected

10.156.3.9 `PublicationPtr gazebo::transport::TopicManager::FindPublication (const std::string & _topic)`

Find a publication object by topic.

Parameters

in	<code>_topic</code>	The topic to search for
----	---------------------	-------------------------

Returns

Pointer to the publication object, if found (can be null)

Referenced by `Advertise()`.

10.156.3.10 `void gazebo::transport::TopicManager::Fini ()`

Finalize the manager.

10.156.3.11 `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
-----	--------------------------	---

10.156.3.12 `void gazebo::transport::TopicManager::Init ()`

Initialize the manager.

10.156.3.13 `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

10.156.3.14 `void gazebo::transport::TopicManager::PauseIncoming (bool _pause)`

Pause or unpause processing of incoming messages.

Parameters

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

10.156.3.15 `void gazebo::transport::TopicManager::ProcessNodes (bool _onlyOut = false)`

Process all nodes under management.

Parameters

<code>in</code>	<code><i>_onlyOut</i></code>	True means only outbound messages on nodes will be sent. False means nodes process both outbound and inbound messages
-----------------	------------------------------	---

10.156.3.16 `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. 578) instead of calling this function directly.

Parameters

<code><i>_topic</i></code>	Name of the topic
<code><i>_message</i></code>	The message to send.
<code><i>_cb</i></code>	Callback, used when the publish is completed.

10.156.3.17 `void gazebo::transport::TopicManager::RegisterTopicNamespace (const std::string & _name)`

Register a new topic namespace.

Parameters

<code>in</code>	<code><i>_name</i></code>	The name of the new namespace
-----------------	---------------------------	-------------------------------

10.156.3.18 `void gazebo::transport::TopicManager::RemoveNode (unsigned int _id)`

Remove a node by its id.

Parameters

<code>in</code>	<code><i>_id</i></code>	The ID of the node to be removed
-----------------	-------------------------	----------------------------------

10.156.3.19 `SubscriberPtr gazebo::transport::TopicManager::Subscribe (const SubscribeOptions & _options)`

Subscribe to a topic.

Parameters

in	<code>_options</code>	The options to use for the subscription
----	-----------------------	---

Returns

Pointer to the newly created subscriber

10.156.3.20 `void gazebo::transport::TopicManager::Unadvertise (const std::string & _topic)`

Unadvertise a topic.

Parameters

in	<code>_topic</code>	The topic to be unadvertised
----	---------------------	------------------------------

10.156.3.21 `void gazebo::transport::TopicManager::Unsubscribe (const std::string & _topic, const NodePtr & _sub)`

Unsubscribe from a topic.

Use a **Subscriber** (p. 718) rather than calling this function directly

Parameters

in	<code>_topic</code>	The topic to unsubscribe from
in	<code>_sub</code>	The node to unsubscribe

10.156.3.22 `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`

10.157 gazebo::physics::TrajectoryInfo Struct Reference

```
#include <Actor.hh>
```

Public Attributes

- double **duration**
- double **endTime**
- unsigned int **id**
- double **startTime**
- bool **translated**
- std::string **type**

10.157.1 Member Data Documentation

10.157.1.1 double gazebo::physics::TrajectoryInfo::duration

10.157.1.2 double gazebo::physics::TrajectoryInfo::endTime

10.157.1.3 unsigned int gazebo::physics::TrajectoryInfo::id

10.157.1.4 double gazebo::physics::TrajectoryInfo::startTime

10.157.1.5 bool gazebo::physics::TrajectoryInfo::translated

10.157.1.6 std::string gazebo::physics::TrajectoryInfo::type

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

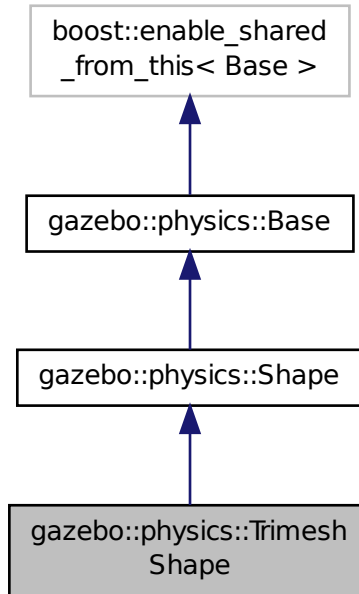
- **Actor.hh**

10.158 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 double **GetMass** (double _density) const **GAZEBO_DEPRECATED**
Deprecated.
- 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

10.158.1 Detailed Description

Triangle mesh collision shape.

10.158.2 Constructor & Destructor Documentation

10.158.2.1 gazebo::physics::TrimeshShape::TrimeshShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

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

10.158.2.2 virtual gazebo::physics::TrimeshShape::~~TrimeshShape () [virtual]

Destructor.

10.158.3 Member Function Documentation

10.158.3.1 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. 669).

10.158.3.2 std::string gazebo::physics::TrimeshShape::GetFilename () const

Get the filename of the mesh data.

Returns

The filename of the mesh data.

10.158.3.3 `virtual double gazebo::physics::TrimeshShape::GetMass (double _density) const` [virtual]

Deprecated.

Reimplemented from `gazebo::physics::Shape` (p. 670).

10.158.3.4 `virtual math::Vector3 gazebo::physics::TrimeshShape::GetSize () const` [virtual]

Get the size of the triangle mesh.

Returns

The size of the triangle mesh.

10.158.3.5 `virtual void gazebo::physics::TrimeshShape::Init ()` [virtual]

Initialize the shape.

Implements `gazebo::physics::Shape` (p. 670).

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

Update this shape from a message.

Parameters

<code>in</code>	<code><i>_msg</i></code>	Message that contains triangle mesh info.
-----------------	--------------------------	---

Implements `gazebo::physics::Shape` (p. 670).

10.158.3.7 `void gazebo::physics::TrimeshShape::SetFilename (const std::string & _filename)`

Set the filename of the triangle mesh.

Parameters

<code>in</code>	<code><i>_filename</i></code>	Filename of the mesh file to load from.
-----------------	-------------------------------	---

10.158.3.8 `void gazebo::physics::TrimeshShape::SetScale (const math::Vector3 & _scale)`

Set the scaling factor.

Parameters

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

10.158.3.9 `virtual void gazebo::physics::TrimeshShape::Update ()` [inline], [virtual]

Update the tri mesh.

Reimplemented from `gazebo::physics::Base` (p. 135).

10.158.4 Member Data Documentation

10.158.4.1 `const common::Mesh* gazebo::physics::TrimeshShape::mesh` [protected]

Pointer to the mesh data.

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

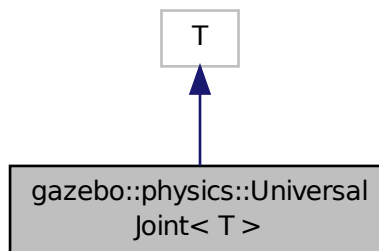
- `TrimeshShape.hh`

10.159 gazebo::physics::UniversalJoint< T > Class Template Reference

A universal joint.

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

Inheritance diagram for `gazebo::physics::UniversalJoint< T >`:



Public Member Functions

- **UniversalJoint** (`BasePtr` _parent)
Constructor.
- virtual `~UniversalJoint` ()
Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (`sdf::ElementPtr` _sdf)
*Load a **UniversalJoint** (p. 765).*

10.159.1 Detailed Description

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

A universal joint.

10.159.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	_parent	Parent link of the univernal joint.
----	---------	-------------------------------------

References gazebo::physics::Base::UNIVERSAL_JOINT.

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

Destuctor.

10.159.3 Member Function Documentation

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

10.159.3.2 `template<class T > virtual void gazebo::physics::UniversalJoint< T >::Load (sdf::ElementPtr _sdf)`
`[inline], [virtual]`

Load a **UniversalJoint** (p. 765).

Parameters

in	_sdf	SDF values to load from.
----	------	--------------------------

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

- **UniversalJoint.hh**

10.160 urdf2gazebo::URDF2Gazebo Class Reference

```
#include <parser_urdf.hh>
```

Public Member Functions

- **URDF2Gazebo ()**

- `~URDF2Gazebo ()`
- void **addKeyValue** (TiXmlElement *elem, const std::string &key, const std::string &value)
append key value pair to the end of the xml element
- void **addTransform** (TiXmlElement *elem, const gazebo::math::Pose &transform)
append transform (pose) to the end of the xml element
- gazebo::math::Pose **copyPose** (urdf::Pose pose)
reduced fixed joints: utility to copy between urdf::Pose and math::Pose
- urdf::Pose **copyPose** (gazebo::math::Pose pose)
reduced fixed joints: utility to copy between urdf::Pose and math::Pose
- void **createCollision** (TiXmlElement *elem, const urdf::Link *link, urdf::Collision *collision, std::string old_link_name=std::string(""))
create SDF Collision block based on URDF
- void **createCollisions** (TiXmlElement *elem, const urdf::Link *link)
create collision blocks from urdf collisions
- void **createGeometry** (TiXmlElement *elem, urdf::Geometry *geometry)
create SDF geometry block based on URDF
- void **createInertial** (TiXmlElement *elem, const urdf::Link *link)
create SDF Inertial block based on URDF
- void **createJoint** (TiXmlElement *root, const urdf::Link *link, gazebo::math::Pose ¤tTransform)
create SDF Joint block based on URDF
- void **createLink** (TiXmlElement *root, const urdf::Link *link, gazebo::math::Pose ¤tTransform)
create SDF Link block based on URDF
- void **createSDF** (TiXmlElement *root, const urdf::Link *link, const gazebo::math::Pose &transform)
create SDF from URDF link
- void **createVisual** (TiXmlElement *elem, const urdf::Link *link, urdf::Visual *visual, std::string old_link_name=std::string(""))
create SDF Visual block based on URDF
- void **createVisuals** (TiXmlElement *elem, const urdf::Link *link)
create visual blocks from urdf visuals
- std::string **getGeometryBoundingBox** (urdf::Geometry *geometry, double *sizeVals)
- std::string **getKeyValueAsString** (TiXmlElement *elem)
get value from <key value="..."> pair and return it as string
- TiXmlDocument **initModelDoc** (TiXmlDocument *_xmlDoc)
- TiXmlDocument **initModelFile** (std::string filename)
- TiXmlDocument **initModelString** (std::string urdf_str)
- TiXmlDocument **initModelString** (std::string urdf_str, bool _enforce_limits)
- void **insertGazeboExtensionCollision** (TiXmlElement *elem, std::string link_name)
insert extensions into collision geoms
- void **insertGazeboExtensionJoint** (TiXmlElement *elem, std::string joint_name)
insert extensions into joints
- void **insertGazeboExtensionLink** (TiXmlElement *elem, std::string link_name)
insert extensions into links
- void **insertGazeboExtensionRobot** (TiXmlElement *elem)
insert extensions into model
- void **insertGazeboExtensionVisual** (TiXmlElement *elem, std::string link_name)
insert extensions into visuals
- gazebo::math::Pose **inverseTransformToParentFrame** (gazebo::math::Pose transform_in_link_frame, urdf::Pose parent_to_link_transform)

- reduced fixed joints: transform to parent frame*
- void **listGazeboExtensions** ()
 - list extensions for debugging*
- void **listGazeboExtensions** (std::string reference)
 - list extensions for debugging*
- void **parseGazeboExtension** (TiXmlDocument &urdf_xml)
 - things that do not belong in urdf but should be mapped into sdf*
- urdf::Vector3 **parseVector3** (TiXmlNode *key, double scale=1.0)
 - parser xml for vector 3*
- void **printCollisionGroups** (urdf::Link *link)
 - print collision groups for debugging purposes*
- void **printMass** (urdf::Link *link)
 - print mass for link for debugging*
- void **printMass** (std::string link_name, dMass mass)
 - print mass for link for debugging*
- void **reduceCollisionsToParent** (urdf::Link *link)
 - reduce fixed joints: lump collisions to parent link*
- void **reduceCollisionToParent** (urdf::Link *link, std::string group_name, urdf::Collision *collision)
 - reduce fixed joints: lump collision when reducing fixed joints*
- void **reduceFixedJoints** (TiXmlElement *root, urdf::Link *link)
 - reduce fixed joints by lumping inertial, visual and*
- void **reduceGazeboExtensionContactSensorFrameReplace** (std::vector< TiXmlElement * >::iterator blob_it, urdf::Link *link)
 - reduced fixed joints: apply appropriate frame updates in urdf extensions when doing fixed joint reduction*
- void **reduceGazeboExtensionFrameReplace** (**GazeboExtension** *ge, urdf::Link *link)
 - reduced fixed joints: apply appropriate frame updates in urdf extensions when doing fixed joint reduction*
- void **reduceGazeboExtensionGripperFrameReplace** (std::vector< TiXmlElement * >::iterator blob_it, urdf::Link *link)
 - reduced fixed joints: apply appropriate frame updates in gripper inside urdf extensions when doing fixed joint reduction*
- void **reduceGazeboExtensionJointFrameReplace** (std::vector< TiXmlElement * >::iterator blob_it, urdf::Link *link)
 - reduced fixed joints: apply appropriate frame updates in joint inside urdf extensions when doing fixed joint reduction*
- void **reduceGazeboExtensionPluginFrameReplace** (std::vector< TiXmlElement * >::iterator blob_it, urdf::Link *link, std::string plugin_name, std::string element_name, **gazebo::math::Pose** reduction_transform)
 - reduced fixed joints: apply appropriate frame updates in plugins inside urdf extensions when doing fixed joint reduction*
- void **reduceGazeboExtensionProjectorFrameReplace** (std::vector< TiXmlElement * >::iterator blob_it, urdf::Link *link)
 - reduced fixed joints: apply appropriate frame updates in projector inside urdf extensions when doing fixed joint reduction*
- void **reduceGazeboExtensionProjectorTransformReduction** (std::vector< TiXmlElement * >::iterator blob_it, **gazebo::math::Pose** reduction_transform)
 - reduced fixed joints: apply transform reduction for projectors in extensions when doing fixed joint reduction*
- void **reduceGazeboExtensionSensorTransformReduction** (std::vector< TiXmlElement * >::iterator blob_it, **gazebo::math::Pose** reduction_transform)
 - reduced fixed joints: apply transform reduction for ray sensors in extensions when doing fixed joint reduction*
- void **reduceGazeboExtensionsTransformReduction** (**GazeboExtension** *ge)
 - reduced fixed joints: apply transform reduction to extensions when doing fixed joint reduction*
- void **reduceGazeboExtensionToParent** (urdf::Link *link)
 - reduced fixed joints: apply appropriate updates to urdf extensions when doing fixed joint reduction*

- void **reduceInertialToParent** (urdf::Link *link)
reduce fixed joints: lump inertial to parent link
- void **reduceJointsToParent** (urdf::Link *link)
reduce fixed joints: lump joints to parent link
- void **reduceVisualsToParent** (urdf::Link *link)
reduce fixed joints: lump visuals to parent link
- void **reduceVisualToParent** (urdf::Link *link, std::string group_name, urdf::Visual *visual)
reduce fixed joints: lump visuals when reducing fixed joints
- urdf::Pose **transformToParentFrame** (urdf::Pose transform_in_link_frame, urdf::Pose parent_to_link_transform)
reduced fixed joints: transform to parent frame
- gazebo::math::Pose **transformToParentFrame** (gazebo::math::Pose transform_in_link_frame, urdf::Pose parent_to_link_transform)
reduced fixed joints: transform to parent frame
- gazebo::math::Pose **transformToParentFrame** (gazebo::math::Pose transform_in_link_frame, gazebo::math::Pose parent_to_link_transform)
reduced fixed joints: transform to parent frame
- std::string **values2str** (unsigned int count, const double *values)
convert values to string
- std::string **vector32str** (const urdf::Vector3 vector)
convert Vector3 to string

Public Attributes

- std::map< std::string, std::vector< **GazeboExtension** * > > **gazebo_extensions_**

10.160.1 Constructor & Destructor Documentation

10.160.1.1 urdf2gazebo::URDF2Gazebo::URDF2Gazebo ()

10.160.1.2 urdf2gazebo::URDF2Gazebo::~~URDF2Gazebo ()

10.160.2 Member Function Documentation

10.160.2.1 void urdf2gazebo::URDF2Gazebo::addKeyValue (TiXmlElement * *elem*, const std::string & *key*, const std::string & *value*)

append key value pair to the end of the xml element

10.160.2.2 void urdf2gazebo::URDF2Gazebo::addTransform (TiXmlElement * *elem*, const gazebo::math::Pose & *transform*)

append transform (pose) to the end of the xml element

10.160.2.3 gazebo::math::Pose urdf2gazebo::URDF2Gazebo::copyPose (urdf::Pose *pose*)

reduced fixed joints: utility to copy between urdf::Pose and math::Pose

10.160.2.4 `urdf::Pose urdf2gazebo::URDF2Gazebo::copyPose (gazebo::math::Pose pose)`

reduced fixed joints: utility to copy between urdf::Pose and math::Pose

10.160.2.5 `void urdf2gazebo::URDF2Gazebo::createCollision (TiXmlElement * elem, const urdf::Link * link, urdf::Collision * collision, std::string old_link_name = std::string(""))`

create SDF Collision block based on URDF

10.160.2.6 `void urdf2gazebo::URDF2Gazebo::createCollisions (TiXmlElement * elem, const urdf::Link * link)`

create collision blocks from urdf collisions

10.160.2.7 `void urdf2gazebo::URDF2Gazebo::createGeometry (TiXmlElement * elem, urdf::Geometry * geometry)`

create SDF geometry block based on URDF

10.160.2.8 `void urdf2gazebo::URDF2Gazebo::createInertial (TiXmlElement * elem, const urdf::Link * link)`

create SDF Inertial block based on URDF

10.160.2.9 `void urdf2gazebo::URDF2Gazebo::createJoint (TiXmlElement * root, const urdf::Link * link, gazebo::math::Pose & currentTransform)`

create SDF Joint block based on URDF

10.160.2.10 `void urdf2gazebo::URDF2Gazebo::createLink (TiXmlElement * root, const urdf::Link * link, gazebo::math::Pose & currentTransform)`

create SDF Link block based on URDF

10.160.2.11 `void urdf2gazebo::URDF2Gazebo::createSDF (TiXmlElement * root, const urdf::Link * link, const gazebo::math::Pose & transform)`

create SDF from URDF link

10.160.2.12 `void urdf2gazebo::URDF2Gazebo::createVisual (TiXmlElement * elem, const urdf::Link * link, urdf::Visual * visual, std::string old_link_name = std::string(""))`

create SDF Visual block based on URDF

10.160.2.13 `void urdf2gazebo::URDF2Gazebo::createVisuals (TiXmlElement * elem, const urdf::Link * link)`

create visual blocks from urdf visuals

10.160.2.14 `std::string urdf2gazebo::URDF2Gazebo::getGeometryBoundingBox (urdf::Geometry * geometry, double * sizeVals)`

10.160.2.15 `std::string urdf2gazebo::URDF2Gazebo::getKeyValueAsString (TiXmlElement * elem)`

get value from <key value="..."> pair and return it as string

10.160.2.16 `TiXmlDocument urdf2gazebo::URDF2Gazebo::initModelDoc (TiXmlDocument * _xmlDoc)`

10.160.2.17 `TiXmlDocument urdf2gazebo::URDF2Gazebo::initModelFile (std::string filename)`

10.160.2.18 `TiXmlDocument urdf2gazebo::URDF2Gazebo::initModelString (std::string urdf_str)`

10.160.2.19 `TiXmlDocument urdf2gazebo::URDF2Gazebo::initModelString (std::string urdf_str, bool _enforce_limits)`

10.160.2.20 `void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionCollision (TiXmlElement * elem, std::string link_name)`

insert extensions into collision geoms

10.160.2.21 `void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionJoint (TiXmlElement * elem, std::string joint_name)`

insert extensions into joints

10.160.2.22 `void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionLink (TiXmlElement * elem, std::string link_name)`

insert extensions into links

10.160.2.23 `void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionRobot (TiXmlElement * elem)`

insert extensions into model

10.160.2.24 `void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionVisual (TiXmlElement * elem, std::string link_name)`

insert extensions into visuals

10.160.2.25 `gazebo::math::Pose urdf2gazebo::URDF2Gazebo::inverseTransformToParentFrame (gazebo::math::Pose transform_in_link_frame, urdf::Pose parent_to_link_transform)`

reduced fixed joints: transform to parent frame

10.160.2.26 `void urdf2gazebo::URDF2Gazebo::listGazeboExtensions ()`

list extensions for debugging

10.160.2.27 `void urdf2gazebo::URDF2Gazebo::listGazeboExtensions (std::string reference)`

list extensions for debugging

10.160.2.28 void urdf2gazebo::URDF2Gazebo::parseGazeboExtension (TiXmlDocument & *urdf_xml*)

things that do not belong in urdf but should be mapped into sdf

Todo : do this using sdf definitions, not hard coded stuff

10.160.2.29 urdf::Vector3 urdf2gazebo::URDF2Gazebo::parseVector3 (TiXmlNode * *key*, double *scale* = 1.0)

parser xml for vector 3

10.160.2.30 void urdf2gazebo::URDF2Gazebo::printCollisionGroups (urdf::Link * *link*)

print collision groups for debugging purposes

10.160.2.31 void urdf2gazebo::URDF2Gazebo::printMass (urdf::Link * *link*)

print mass for link for debugging

10.160.2.32 void urdf2gazebo::URDF2Gazebo::printMass (std::string *link_name*, dMass *mass*)

print mass for link for debugging

10.160.2.33 void urdf2gazebo::URDF2Gazebo::reduceCollisionsToParent (urdf::Link * *link*)

reduce fixed joints: lump collisions to parent link

10.160.2.34 void urdf2gazebo::URDF2Gazebo::reduceCollisionToParent (urdf::Link * *link*, std::string *group_name*, urdf::Collision * *collision*)

reduce fixed joints: lump collision when reducing fixed joints

10.160.2.35 void urdf2gazebo::URDF2Gazebo::reduceFixedJoints (TiXmlElement * *root*, urdf::Link * *link*)

reduce fixed joints by lumping inertial, visual and

10.160.2.36 void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionContactSensorFrameReplace (std::vector< TiXmlElement * >::iterator *blob_it*, urdf::Link * *link*)

reduced fixed joints: apply appropriate frame updates in urdf extensions when doing fixed joint reduction

10.160.2.37 void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionFrameReplace (GazeboExtension * *ge*, urdf::Link * *link*)

reduced fixed joints: apply appropriate frame updates in urdf extensions when doing fixed joint reduction

10.160.2.38 `void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionGripperFrameReplace (std::vector< TiXmlElement * >::iterator blob_it, urdf::Link * link)`

reduced fixed joints: apply appropriate frame updates in gripper inside urdf extensions when doing fixed joint reduction

10.160.2.39 `void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionJointFrameReplace (std::vector< TiXmlElement * >::iterator blob_it, urdf::Link * link)`

reduced fixed joints: apply appropriate frame updates in joint inside urdf extensions when doing fixed joint reduction

10.160.2.40 `void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionPluginFrameReplace (std::vector< TiXmlElement * >::iterator blob_it, urdf::Link * link, std::string plugin_name, std::string element_name, gazebo::math::Pose reduction_transform)`

reduced fixed joints: apply appropriate frame updates in plugins inside urdf extensions when doing fixed joint reduction

10.160.2.41 `void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionProjectorFrameReplace (std::vector< TiXmlElement * >::iterator blob_it, urdf::Link * link)`

reduced fixed joints: apply appropriate frame updates in projector inside urdf extensions when doing fixed joint reduction

10.160.2.42 `void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionProjectorTransformReduction (std::vector< TiXmlElement * >::iterator blob_it, gazebo::math::Pose reduction_transform)`

reduced fixed joints: apply transform reduction for projectors in extensions when doing fixed joint reduction

10.160.2.43 `void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionSensorTransformReduction (std::vector< TiXmlElement * >::iterator blob_it, gazebo::math::Pose reduction_transform)`

reduced fixed joints: apply transform reduction for ray sensors in extensions when doing fixed joint reduction

10.160.2.44 `void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionsTransformReduction (GazeboExtension * ge)`

reduced fixed joints: apply transform reduction to extensions when doing fixed joint reduction

10.160.2.45 `void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionToParent (urdf::Link * link)`

reduced fixed joints: apply appropriate updates to urdf extensions when doing fixed joint reduction

10.160.2.46 `void urdf2gazebo::URDF2Gazebo::reduceInertialToParent (urdf::Link * link)`

reduce fixed joints: lump inertial to parent link

10.160.2.47 `void urdf2gazebo::URDF2Gazebo::reduceJointsToParent (urdf::Link * link)`

reduce fixed joints: lump joints to parent link

10.160.2.48 `void urdf2gazebo::URDF2Gazebo::reduceVisualsToParent (urdf::Link * link)`

reduce fixed joints: lump visuals to parent link

10.160.2.49 `void urdf2gazebo::URDF2Gazebo::reduceVisualToParent (urdf::Link * link, std::string group_name, urdf::Visual * visual)`

reduce fixed joints: lump visuals when reducing fixed joints

10.160.2.50 `urdf::Pose urdf2gazebo::URDF2Gazebo::transformToParentFrame (urdf::Pose transform_in_link_frame, urdf::Pose parent_to_link_transform)`

reduced fixed joints: transform to parent frame

10.160.2.51 `gazebo::math::Pose urdf2gazebo::URDF2Gazebo::transformToParentFrame (gazebo::math::Pose transform_in_link_frame, urdf::Pose parent_to_link_transform)`

reduced fixed joints: transform to parent frame

10.160.2.52 `gazebo::math::Pose urdf2gazebo::URDF2Gazebo::transformToParentFrame (gazebo::math::Pose transform_in_link_frame, gazebo::math::Pose parent_to_link_transform)`

reduced fixed joints: transform to parent frame

10.160.2.53 `std::string urdf2gazebo::URDF2Gazebo::values2str (unsigned int count, const double * values)`

convert values to string

10.160.2.54 `std::string urdf2gazebo::URDF2Gazebo::vector32str (const urdf::Vector3 vector)`

convert Vector3 to string

10.160.3 Member Data Documentation

10.160.3.1 `std::map<std::string, std::vector<GazeboExtension*> > urdf2gazebo::URDF2Gazebo::gazebo_extensions_`

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

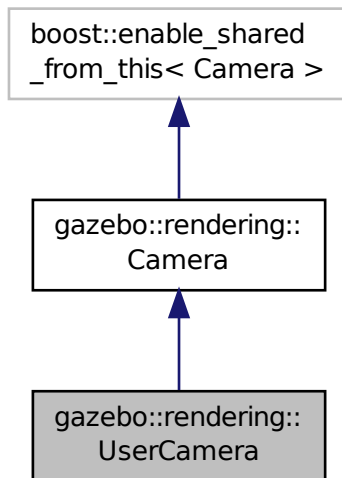
- `parser_urdf.hh`

10.161 gazebo::rendering::UserCamera Class Reference

A camera used for user visualization of a scene.

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

Inheritance diagram for gazebo::rendering::UserCamera:



Public Member Functions

- **UserCamera** (const std::string &_name, **ScenePtr** _scene)
Constructor.
- virtual **~UserCamera** ()
Destructor.
- void **EnableViewController** (bool _value) const
Set whether the view controller is enabled.
- void **Fini** ()
Finalize.
- float **GetAvgFPS** () const
Get the average frames per second.
- **GUIOverlay** * **GetGUIOverlay** ()
Get the GUI overlay.
- float **GetTriangleCount** () const
Get the triangle count.
- **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

10.161.1 Detailed Description

A camera used for user visualization of a scene.

10.161.2 Constructor & Destructor Documentation

10.161.2.1 gazebo::rendering::UserCamera::UserCamera (const std::string & *_name*, ScenePtr *_scene*)

Constructor.

Parameters

in	<i>_name</i>	Name of the camera.
in	<i>_scene</i>	Scene (p. 632) to put the camera in.

10.161.2.2 virtual gazebo::rendering::UserCamera::~~UserCamera () [virtual]

Destructor.

10.161.3 Member Function Documentation

10.161.3.1 virtual void gazebo::rendering::UserCamera::AnimationComplete () [protected], [virtual]

Internal function used to indicate that an animation has completed.

Reimplemented from **gazebo::rendering::Camera** (p. 155).

10.161.3.2 virtual bool gazebo::rendering::UserCamera::AttachToVisualImpl (VisualPtr *_visual*, bool *_inheritOrientation*, double *_minDist* = 0, double *_maxDist* = 0) [protected], [virtual]

Set the camera to be attached to a visual.

This causes the camera to move in relation to the specified visual.

Parameters

in	<i>_visual</i>	The visual to attach to.
in	<i>_inheritOrientation</i>	True if the camera should also rotate when the visual rotates.
in	<i>_minDist</i>	Minimum distance the camera can get to the visual.
in	<i>_maxDist</i>	Maximum distance the camera can get from the visual.

Returns

True if successfully attach to the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 156).

10.161.3.3 void gazebo::rendering::UserCamera::EnableViewController (bool *_value*) const

Set whether the view controller is enabled.

The view controller is used to handle user camera movements.

Parameters

in	<i>_value</i>	True to enable viewcontroller, False to disable.
----	---------------	--

10.161.3.4 `void gazebo::rendering::UserCamera::Fini () [virtual]`

Finalize.

Reimplemented from `gazebo::rendering::Camera` (p. 157).

10.161.3.5 `float gazebo::rendering::UserCamera::GetAvgFPS () const`

Get the average frames per second.

Returns

The average rendering frames per second

10.161.3.6 `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. 335).

10.161.3.7 `float gazebo::rendering::UserCamera::GetTriangleCount () const`

Get the triangle count.

Returns

The number of triangles currently being rendered.

10.161.3.8 `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

10.161.3.9 `VisualPtr gazebo::rendering::UserCamera::GetVisual (const math::Vector2i & _mousePos) const`

Get a visual at a mouse position.

Parameters

in	<code>_mousePos</code>	2D position of the mouse in pixels.
----	------------------------	-------------------------------------

10.161.3.10 `void gazebo::rendering::UserCamera::HandleKeyPressEvent (const std::string & _key)`

Handle a key press.

Parameters

in	<code>_key</code>	The key pressed.
----	-------------------	------------------

10.161.3.11 `void gazebo::rendering::UserCamera::HandleKeyReleaseEvent (const std::string & _key)`

Handle a key release.

Parameters

in	<code>_key</code>	The key released.
----	-------------------	-------------------

10.161.3.12 `void gazebo::rendering::UserCamera::HandleMouseEvent (const common::MouseEvent & _evt)`

Handle a mouse event.

Parameters

in	<code>_evt</code>	The mouse event.
----	-------------------	------------------

10.161.3.13 `void gazebo::rendering::UserCamera::Init () [virtual]`

Initialize.

Reimplemented from `gazebo::rendering::Camera` (p. 164).

10.161.3.14 `void gazebo::rendering::UserCamera::Load (sdf::ElementPtr _sdf) [virtual]`

Load the user camera.

Parameters

in	<code>_sdf</code>	Parameters for the camera.
----	-------------------	----------------------------

Reimplemented from `gazebo::rendering::Camera` (p. 165).

10.161.3.15 `void gazebo::rendering::UserCamera::Load () [virtual]`

Generic load function.

Reimplemented from `gazebo::rendering::Camera` (p. 165).

10.161.3.16 `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. 165)

Parameters

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

Reimplemented from **gazebo::rendering::Camera** (p. 165).

10.161.3.17 `void gazebo::rendering::UserCamera::MoveToVisual (VisualPtr _visual)`

Move the camera to focus on a visual.

Parameters

<code>in</code>	<code>_visual</code>	Visual (p. 828) to move the camera to.
-----------------	----------------------	---

10.161.3.18 `void gazebo::rendering::UserCamera::MoveToVisual (const std::string & _visualName)`

Move the camera to focus on a visual.

Parameters

<code>in</code>	<code>_visualName</code>	Name of the visual to move the camera to.
-----------------	--------------------------	---

10.161.3.19 `virtual void gazebo::rendering::UserCamera::PostRender ()` `[virtual]`

Post render.

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

10.161.3.20 `void gazebo::rendering::UserCamera::Resize (unsigned int _w, unsigned int _h)`

Resize the camera.

Parameters

<code>in</code>	<code>_w</code>	Width of the camera image.
<code>in</code>	<code>_h</code>	Height of the camera image.

10.161.3.21 `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
----	------------------	-----------------

10.161.3.22 `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. 168).

10.161.3.23 `void gazebo::rendering::UserCamera::SetViewController (const std::string & _type)`

Set view controller.

Parameters

in	<code>_type</code>	The type of view controller: "orbit", "fps"
----	--------------------	---

10.161.3.24 `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.

10.161.3.25 `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.

10.161.3.26 `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. 169).

10.161.3.27 `virtual bool gazebo::rendering::UserCamera::TrackVisualImpl (VisualPtr _visual)` [protected], [virtual]

Set the camera to track a scene node.

Tracking just causes the camera to rotate to follow the visual.

Parameters

in	<code>_visual</code>	Visual (p. 828) to track.
----	----------------------	----------------------------------

Returns

True if the camera is now tracking the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 170).

10.161.3.28 `virtual void gazebo::rendering::UserCamera::Update ()` [virtual]

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 171).

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

- **UserCamera.hh**

10.162 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.

10.162.1 Detailed Description

Generic double x, y vector.

10.162.2 Constructor & Destructor Documentation

10.162.2.1 `gazebo::math::Vector2d::Vector2d ()`

Constructor.

10.162.2.2 `gazebo::math::Vector2d::Vector2d (const double &_x, const double &_y)`

Constructor.

Parameters

<code>in</code>	<code>_x</code>	value along x
<code>in</code>	<code>_y</code>	value along y

10.162.2.3 `gazebo::math::Vector2d::Vector2d (const Vector2d &_v)`

Copy constructor.

Parameters

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

10.162.2.4 `virtual gazebo::math::Vector2d::~~Vector2d () [virtual]`

Destructor.

10.162.3 Member Function Documentation

10.162.3.1 `Vector2d gazebo::math::Vector2d::Cross (const Vector2d &_v) const`

Return the cross product of this vector and `_v`.

Parameters

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

Returns

the cross product

10.162.3.2 `double gazebo::math::Vector2d::Distance (const Vector2d & _pt) const`

Calc distance to the given point.

Parameters

<code>in</code>	<code>_pt</code>	The point to measure to
-----------------	------------------	-------------------------

Returns

the distance

10.162.3.3 `bool gazebo::math::Vector2d::IsFinite () const`

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.162.3.4 `void gazebo::math::Vector2d::Normalize ()`

Normalize the vector length.

10.162.3.5 `bool gazebo::math::Vector2d::operator!=(const Vector2d & _v) const`

Not equal to operator.

Returns

true if elements are of diffent values (tolerence 1e-6)

10.162.3.6 `const Vector2d gazebo::math::Vector2d::operator* (const Vector2d & _v) const`

Multiplication operators.

Parameters

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

Returns

the result

10.162.3.7 `const Vector2d gazebo::math::Vector2d::operator* (double _v) const`

Multiplication operators.

Parameters

<code>in</code>	<code>_v</code>	the scaling factor
-----------------	-----------------	--------------------

Returns

a scaled vector

10.162.3.8 `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

10.162.3.9 `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

10.162.3.10 `Vector2d gazebo::math::Vector2d::operator+ (const Vector2d & _v) const`

Addition operator.

Parameters

<code>in</code>	<code>_v</code>	vector to add
-----------------	-----------------	---------------

Returns

sum vector

10.162.3.11 `const Vector2d& gazebo::math::Vector2d::operator+=(const Vector2d & _v)`

Addition assignment operator.

Parameters

in	_v	the vector to add
----	----	-------------------

10.162.3.12 `Vector2d gazebo::math::Vector2d::operator-(const Vector2d & _v) const`

Subtraction operator.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

the subtracted vector

10.162.3.13 `const Vector2d& gazebo::math::Vector2d::operator-= (const Vector2d & _v)`

Subtraction assignment operator.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

this

10.162.3.14 `const Vector2d gazebo::math::Vector2d::operator/ (const Vector2d & _v) const`

Division operator.

Remarks

this is an element wise division

Parameters

in	_v	a vector
----	----	----------

Returns

a result

10.162.3.15 `const Vector2d gazebo::math::Vector2d::operator/ (double _v) const`

Division operator.

Parameters

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

Returns

a vector

10.162.3.16 `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

10.162.3.17 `const Vector2d& gazebo::math::Vector2d::operator/= (double _v)`

Division operator.

Parameters

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

Returns

a vector

10.162.3.18 `Vector2d& gazebo::math::Vector2d::operator= (const Vector2d & _v)`

Assignment operator.

Parameters

in	_v	a value for x and y element
----	----	-----------------------------

Returns

this

10.162.3.19 `const Vector2d& gazebo::math::Vector2d::operator= (double _v)`

Assignment operator.

Parameters

in	_v	the value for x and y element
----	----	-------------------------------

Returns

this

10.162.3.20 `bool gazebo::math::Vector2d::operator== (const Vector2d & _v) const`

Equal to operator.

Parameters

in	_v	the vector to compare to
----	----	--------------------------

Returns

true if the elements of the 2 vectors are equal within a tolerance (1e-6)

10.162.3.21 `double gazebo::math::Vector2d::operator[] (unsigned int _index) const`

Array subscript operator.

Parameters

in	_index	the index
----	--------	-----------

Returns

the value, or 0 if _index is out of bounds

10.162.3.22 `void gazebo::math::Vector2d::Set (double _x, double _y)`

Set the contents of the vector.

Parameters

in	<code>_x</code>	value along x
in	<code>_y</code>	value along y

10.162.4 Friends And Related Function Documentation

10.162.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector2d & _pt)` [*friend*]

Stream extraction operator.

Parameters

in	<code>_out</code>	output stream
in	<code>_pt</code>	Vector2d (p. 782) to output

Returns

The stream

10.162.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector2d & _pt)` [*friend*]

Stream extraction operator.

Parameters

in	<code>_in</code>	input stream
in	<code>_pt</code>	Vector3 (p. 799) to read values into

Returns

The stream

10.162.5 Member Data Documentation

10.162.5.1 `double gazebo::math::Vector2d::x`

x data

10.162.5.2 `double gazebo::math::Vector2d::y`

y data

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

- **Vector2d.hh**

10.163 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.

10.163.1 Detailed Description

Generic integer x, y vector.

10.163.2 Constructor & Destructor Documentation

10.163.2.1 gazebo::math::Vector2i::Vector2i ()

Constructor.

10.163.2.2 gazebo::math::Vector2i::Vector2i (const int & _x, const int & _y)

Constructor.

Parameters

in	_x	value along x
in	_y	value along y

10.163.2.3 gazebo::math::Vector2i::Vector2i (const Vector2i & _pt)

Copy onstructor.

Parameters

in	<code>_pt</code>	a point
----	------------------	---------

10.163.2.4 virtual gazebo::math::Vector2i::~~Vector2i () [virtual]

Destructor.

10.163.3 Member Function Documentation

10.163.3.1 Vector2i gazebo::math::Vector2i::Cross (const Vector2i & `_pt`) const

Return the cross product of this vector and `_pt`.

Parameters

in	<code>_pt</code>	the other vector
----	------------------	------------------

Returns

the product

10.163.3.2 int gazebo::math::Vector2i::Distance (const Vector2i & `_pt`) const

Calc distance to the given point.

Parameters

in	<code>_pt</code>	a point
----	------------------	---------

Returns

the distance

10.163.3.3 bool gazebo::math::Vector2i::IsFinite () const

See if a point is finite (e.g., not nan)

Returns

the result

10.163.3.4 void gazebo::math::Vector2i::Normalize ()

Normalize the vector length.

10.163.3.5 bool gazebo::math::Vector2i::operator!=(const Vector2i & `_v`) const

Equality operators.

Parameters

	_v	the vector to compare with
--	----	----------------------------

Returns

true if component have different values, false otherwise

10.163.3.6 `const Vector2i gazebo::math::Vector2i::operator* (const Vector2i & _v) const`

Multiplication operator.

Remarks

this is an element wise multiplication

Parameters

in	_v	the vector
----	----	------------

Returns

the result

10.163.3.7 `const Vector2i gazebo::math::Vector2i::operator* (int _v) const`

Multiplication operator.

Parameters

in	_v	the scaling factor
----	----	--------------------

Returns

the result

10.163.3.8 `const Vector2i& gazebo::math::Vector2i::operator*=(const Vector2i & _v)`

Multiplication operators.

Remarks

this is an element wise multiplication

Parameters

in	_v	the vector
----	----	------------

Returns

this

10.163.3.9 `const Vector2i& gazebo::math::Vector2i::operator*=(int _v)`

Multiplication operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

this

10.163.3.10 `Vector2i gazebo::math::Vector2i::operator+(const Vector2i & _v) const`

Addition operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to add
-----------------	-----------------	-------------------

Returns

the sum vector

10.163.3.11 `const Vector2i& gazebo::math::Vector2i::operator+=(const Vector2i & _v)`

Addition assignment operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to add
-----------------	-----------------	-------------------

Returns

this

10.163.3.12 `Vector2i gazebo::math::Vector2i::operator-(const Vector2i & _v) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

the result vector

10.163.3.13 `const Vector2i& gazebo::math::Vector2i::operator-= (const Vector2i & _v)`

Subtraction operators.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

this

10.163.3.14 `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

10.163.3.15 `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

10.163.3.16 `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

10.163.3.17 `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

10.163.3.18 `Vector2i& gazebo::math::Vector2i::operator= (const Vector2i & _v)`

Assignment operator.

Parameters

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

Returns

this

10.163.3.19 `const Vector2i& gazebo::math::Vector2i::operator= (int _value)`

Assignment operator.

Parameters

<i>in</i>	<i>_value</i>	the value for x and y
-----------	---------------	-----------------------

Returns

this

10.163.3.20 `bool gazebo::math::Vector2i::operator==(const Vector2i & _v) const`

Equality operator.

Parameters

	<i>_v</i>	the vector to compare with
--	-----------	----------------------------

Returns

true if component have the same values, false otherwise

10.163.3.21 `int gazebo::math::Vector2i::operator[](unsigned int _index) const`

Array subscript operator.

Parameters

<i>in</i>	<i>_index</i>	the array index
-----------	---------------	-----------------

10.163.3.22 `void gazebo::math::Vector2i::Set (int _x, int _y)`

Set the contents of the vector.

Parameters

<i>in</i>	<i>_x</i>	value along x
<i>in</i>	<i>_y</i>	value along y

10.163.4 Friends And Related Function Documentation

10.163.4.1 `std::ostream& operator<<(std::ostream & _out, const gazebo::math::Vector2i & _pt)` [*friend*]

Stream insertion operator.

Parameters

<i>in</i>	<i>_out</i>	output stream
<i>in</i>	<i>pt</i>	Vector2i (p. 790) to output

Returns

the stream

10.163.4.2 `std::istream& operator>> (std::istream & in, gazebo::math::Vector2i & pt)` [*friend*]

Stream extraction operator.

Parameters

<code>in</code>	<code><i>_in</i></code>	input stream
<code>in</code>	<code><i>_pt</i></code>	Vector3 (p. 799) to read values into

Returns

The stream

10.163.5 Member Data Documentation

10.163.5.1 `int gazebo::math::Vector2i::x`

x data

10.163.5.2 `int gazebo::math::Vector2i::y`

y data

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

- **Vector2i.hh**

10.164 gazebo::math::Vector3 Class Reference

The **Vector3** (p. 799) class represents the generic vector containing 3 elements.

```
#include <math/gzmath.hh>
```

Public Member Functions

- **Vector3** ()
Constructor.
- **Vector3** (const double &*x*, const double &*y*, const double &*z*)
Constructor.
- **Vector3** (const **Vector3** &*v*)
Copy constructor.
- virtual **~Vector3** ()
Destructor.
- void **Correct** ()
Corrects any nan values.

- **Vector3 Cross** (const **Vector3** &_pt) const
Return the cross product of this vector and pt.
- double **Distance** (const **Vector3** &_pt) const
Calc distance to the given point.
- double **Distance** (double _x, double _y, double _z) const
Calc distance to the given point.
- double **Dot** (const **Vector3** &_pt) const
Return the dot product of this vector and pt.
- bool **Equal** (const **Vector3** &_v) const
Equality test.
- **Vector3 GetAbs** () const
Get the absolute value of the vector.
- double **GetDistToLine** (const **Vector3** &_pt1, const **Vector3** &_pt2)
Get distance to a line.
- double **GetLength** () const
Returns the length (magnitude) of the vector \ return the length.
- double **GetMax** () const
Get the maximum value in the vector.
- double **GetMin** () const
Get the minimum value in the vector.
- **Vector3 GetPerpendicular** () const
Return a vector that is perpendicular to this one.
- **Vector3 GetRounded** () const
Get a rounded version of this vector.
- double **GetSquaredLength** () const
Return the square of the length (magnitude) of the vector.
- double **GetSum** () const
Return the sum of the values.
- bool **IsFinite** () const
See if a point is finite (e.g., not nan)
- **Vector3 Normalize** ()
Normalize the vector length.
- bool **operator!=** (const **Vector3** &_v) const
Not equal to operator.
- **Vector3 operator*** (const **Vector3** &_p) const
Multiplication operator.
- **Vector3 operator*** (double _v) const
Multiplication operators.
- const **Vector3** & **operator*= **(const Vector3 &_v)****
Multiplication operators.
- const **Vector3** & **operator*= **(double _v)****
Multiplication operator.
- **Vector3 operator+** (const **Vector3** &_v) const
Addition operator.
- const **Vector3** & **operator+= **(const Vector3 &_v)****
Addition assignment operator.
- **Vector3 operator-** (const **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

- 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.

10.164.1 Detailed Description

The **Vector3** (p. 799) 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.

10.164.2 Constructor & Destructor Documentation

10.164.2.1 gazebo::math::Vector3::Vector3 ()

Constructor.

Referenced by operator-().

10.164.2.2 gazebo::math::Vector3::Vector3 (const double & _x, const double & _y, const double & _z)

Constructor.

Parameters

in	<code>_x</code>	value along x
in	<code>_y</code>	value along y
in	<code>_z</code>	value along z

10.164.2.3 gazebo::math::Vector3::Vector3 (const Vector3 & _v)

Copy constructor.

Parameters

in	<code>_v</code>	a vector
----	-----------------	----------

10.164.2.4 virtual gazebo::math::Vector3::~Vector3 () [virtual]

Destructor.

10.164.3 Member Function Documentation

10.164.3.1 void gazebo::math::Vector3::Correct () [inline]

Corrects any nan values.

References x, y, and z.

Referenced by gazebo::math::Pose::Correct().

10.164.3.2 Vector3 gazebo::math::Vector3::Cross (const Vector3 & _pt) const

Return the cross product of this vector and pt.

Returns

the product

10.164.3.3 double gazebo::math::Vector3::Distance (const Vector3 & _pt) const

Calc distance to the given point.

Parameters

in	_pt	the point
----	-----	-----------

Returns

the distance

10.164.3.4 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

10.164.3.5 double gazebo::math::Vector3::Dot (const Vector3 & _pt) const

Return the dot product of this vector and pt.

Returns

the product

10.164.3.6 `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

10.164.3.7 `Vector3 gazebo::math::Vector3::GetAbs () const`

Get the absolute value of the vector.

Returns

a vector with positive elements

10.164.3.8 `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

10.164.3.9 `double gazebo::math::Vector3::GetLength () const`

Returns the length (magnitude) of the vector \ return the length.

10.164.3.10 `double gazebo::math::Vector3::GetMax () const`

Get the maximum value in the vector.

Returns

the maximum element

10.164.3.11 `double gazebo::math::Vector3::GetMin () const`

Get the minimum value in the vector.

Returns

the minimum element

10.164.3.12 `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

10.164.3.13 `Vector3 gazebo::math::Vector3::GetPerpendicular () const`

Return a vector that is perpendicular to this one.

Returns

an orthogonal vector

10.164.3.14 `Vector3 gazebo::math::Vector3::GetRounded () const`

Get a rounded version of this vector.

Returns

a rounded vector

10.164.3.15 `double gazebo::math::Vector3::GetSquaredLength () const`

Return the square of the length (magnitude) of the vector.

Returns

the squared length

10.164.3.16 `double gazebo::math::Vector3::GetSum () const`

Return the sum of the values.

Returns

the sum

10.164.3.17 `bool gazebo::math::Vector3::IsFinite () const`

See if a point is finite (e.g., not nan)

10.164.3.18 `Vector3 gazebo::math::Vector3::Normalize ()`

Normalize the vector length.

Returns

unit length vector

10.164.3.19 `bool gazebo::math::Vector3::operator!=(const Vector3 & _v) const`

Not equal to operator.

Parameters

in	_v	The vector to compare against
----	----	-------------------------------

Returns

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.164.3.20 `Vector3 gazebo::math::Vector3::operator*(const Vector3 & _p) const`

Multiplication operator.

Remarks

this is an element wise multiplication, not a cross product

Parameters

in	_v	
----	----	--

10.164.3.21 `Vector3 gazebo::math::Vector3::operator*(double _v) const`

Multiplication operators.

Parameters

in	_v	the scaling factor
----	----	--------------------

Returns

a scaled vector

10.164.3.22 `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

10.164.3.23 `const Vector3& gazebo::math::Vector3::operator*=(double _v)`

Multiplication operator.

Parameters

in	_v	scaling factor
----	----	----------------

Returns

this

10.164.3.24 `Vector3 gazebo::math::Vector3::operator+(const Vector3 & _v) const`

Addition operator.

Parameters

in	_v	vector to add
----	----	---------------

Returns

the sum vector

10.164.3.25 `const Vector3& gazebo::math::Vector3::operator+=(const Vector3 & _v)`

Addition assignment operator.

Parameters

<code>in</code>	<code>_v</code>	vector to add
-----------------	-----------------	---------------

10.164.3.26 `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.

10.164.3.27 `const Vector3& gazebo::math::Vector3::operator-=(const Vector3 & _pt)`

Subtraction operators.

Parameters

<code>in</code>	<code>_pt</code>	subtrahend
-----------------	------------------	------------

10.164.3.28 `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

10.164.3.29 `const Vector3 gazebo::math::Vector3::operator/(double _v) const`

Division operator.

Remarks

this is an element wise division

Returns

a vector

10.164.3.30 `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

10.164.3.31 `const Vector3& gazebo::math::Vector3::operator/= (double _v)`

Division operator.

Remarks

this is an element wise division

Returns

this

10.164.3.32 `Vector3& gazebo::math::Vector3::operator= (const Vector3 & _v)`

Assignment operator.

Parameters

<i>in</i>	<i>_v</i>	a new value
-----------	-----------	-------------

Returns

this

10.164.3.33 `Vector3& gazebo::math::Vector3::operator= (double _value)`

Assignment operator.

Parameters

<i>in</i>	<i>_value</i>	assigned to all elements
-----------	---------------	--------------------------

Returns

this

10.164.3.34 `bool gazebo::math::Vector3::operator==(const Vector3 & _pt) const`

Equal to operator.

Parameters

<code>in</code>	<code>_pt</code>	The vector to compare against
-----------------	------------------	-------------------------------

Returns

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.164.3.35 `double gazebo::math::Vector3::operator[] (unsigned int index) const`

[] operator

10.164.3.36 `Vector3 gazebo::math::Vector3::Round ()`

Round to near whole number, return the result.

Returns

the result

10.164.3.37 `void gazebo::math::Vector3::Round (int _precision)`

Round all values to `_precision` decimal places.

Parameters

<code>in</code>	<code>_precision</code>	the decimal places
-----------------	-------------------------	--------------------

10.164.3.38 `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.

10.164.3.39 void gazebo::math::Vector3::SetToMax (const Vector3 & _v)

Set this vector's components to the maximum of itself and the passed in vector.

Parameters

<i>in</i>	<i>_v</i>	the maximum clamping vector
-----------	-----------	-----------------------------

10.164.3.40 void gazebo::math::Vector3::SetToMin (const Vector3 & _v)

Set this vector's components to the minimum of itself and the passed in vector.

Parameters

<i>in</i>	<i>_v</i>	the minimum clamping vector
-----------	-----------	-----------------------------

10.164.4 Friends And Related Function Documentation

10.164.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::math::Vector3 & *_pt*) [friend]

Stream insertion operator.

Parameters

<i>_out</i>	output stream
<i>_pt</i>	Vector3 (p. 799) to output

Returns

the stream

10.164.4.2 std::istream& operator>> (std::istream & *_in*, gazebo::math::Vector3 & *_pt*) [friend]

Stream extraction operator.

Parameters

<i>_in</i>	input stream
<i>_pt</i>	vector3 to read values into

Returns

the stream

10.164.5 Member Data Documentation

10.164.5.1 double gazebo::math::Vector3::x

X location.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), operator-(), gazebo::math::Quaternion::RotateVector(), and Set().

10.164.5.2 double gazebo::math::Vector3::y

Y location.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), operator-(), gazebo::math::Quaternion::RotateVector(), and Set().

10.164.5.3 double gazebo::math::Vector3::z

Z location.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), operator-(), gazebo::math::Quaternion::RotateVector(), and Set().

10.164.5.4 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**

10.165 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.

10.165.1 Detailed Description

double Generic x, y, z, w vector

10.165.2 Constructor & Destructor Documentation

10.165.2.1 gazebo::math::Vector4::Vector4 ()

Constructor.

10.165.2.2 gazebo::math::Vector4::Vector4 (const double & _x, const double & _y, const double & _z, const double & _w)

Constructor with component values.

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

10.165.2.3 gazebo::math::Vector4::Vector4 (const Vector4 & _v)

Copy constructor.

Parameters

<code>in</code>	<code>_v</code>	vector
-----------------	-----------------	--------

10.165.2.4 virtual gazebo::math::Vector4::~~Vector4 () [virtual]

Destructor.

10.165.3 Member Function Documentation

10.165.3.1 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

10.165.3.2 double gazebo::math::Vector4::GetLength () const

Returns the length (magnitude) of the vector.

10.165.3.3 double gazebo::math::Vector4::GetSquaredLength () const

Return the square of the length (magnitude) of the vector.

Returns

the length

10.165.3.4 bool gazebo::math::Vector4::IsFinite () const

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.165.3.5 void gazebo::math::Vector4::Normalize ()

Normalize the vector length.

10.165.3.6 bool gazebo::math::Vector4::operator!= (const Vector4 & *_pt*) const

Not equal to operator.

Parameters

in	<i>_pt</i>	the other vector
----	------------	------------------

Returns

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.165.3.7 `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

10.165.3.8 `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`

10.165.3.9 `const Vector4 gazebo::math::Vector4::operator* (double _v) const`

Multiplication operators.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

a scaled vector

10.165.3.10 `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

10.165.3.11 `const Vector4& gazebo::math::Vector4::operator*= (double _v)`

Multiplication assignment operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

this

10.165.3.12 `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

10.165.3.13 `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

10.165.3.14 **Vector4** gazebo::math::Vector4::operator- (const Vector4 & _v) const

Subtraction operator.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

a vector

10.165.3.15 **const Vector4&** gazebo::math::Vector4::operator-= (const Vector4 & _v)

Subtraction assignment operators.

Parameters

in	_v	the vector to subtract
----	----	------------------------

Returns

this vector

10.165.3.16 **const Vector4** gazebo::math::Vector4::operator/ (const Vector4 & _v) const

Division assignment operator.

Remarks

Performs element wise division, which has limited use.

Parameters

in	_v	the vector to perform element wise division with
----	----	--

Returns

a result vector

10.165.3.17 **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

10.165.3.18 `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

10.165.3.19 `const Vector4& gazebo::math::Vector4::operator/= (double _v)`

Division operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

a vector

10.165.3.20 `Vector4& gazebo::math::Vector4::operator= (const Vector4 & _v)`

Assignment operator.

Parameters

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

Returns

a reference to this vector

10.165.3.21 `Vector4& gazebo::math::Vector4::operator=(double _value)`

Assignment operator.

Parameters

in	<code>_value</code>	
----	---------------------	--

10.165.3.22 `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

10.165.3.23 `double gazebo::math::Vector4::operator[](unsigned int _index) const`

Array subscript operator.

Parameters

in	<code>_index</code>	
----	---------------------	--

10.165.3.24 `void gazebo::math::Vector4::Set (double _x = 0, double _y = 0, double _z = 0, double _w = 0)`

Set the contents of the vector.

Parameters

in	<code>_x</code>	value along x axis
in	<code>_y</code>	value along y axis
in	<code>_z</code>	value along z axis
in	<code>_w</code>	value along w axis

10.165.4 Friends And Related Function Documentation

10.165.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector4 & _pt)` [friend]

Stream insertion operator.

Parameters

in	<code>_out</code>	output stream
in	<code>_pt</code>	Vector4 (p. 812) to output

Returns

The stream

10.165.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector4 & _pt)` [[friend](#)]

Stream extraction operator.

Parameters

<code><i>in</i></code>	<code><i>_in</i></code>	input stream
<code><i>in</i></code>	<code><i>_pt</i></code>	Vector4 (p. 812) to read values into

Returns

the stream

10.165.5 Member Data Documentation

10.165.5.1 `double gazebo::math::Vector4::w`

W value.

10.165.5.2 `double gazebo::math::Vector4::x`

X value.

10.165.5.3 `double gazebo::math::Vector4::y`

Y value.

10.165.5.4 `double gazebo::math::Vector4::z`

Z value.

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

- **Vector4.hh**

10.166 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.

10.166.1 Detailed Description

Handle video encoding and decoding using libavcodec.

10.166.2 Constructor & Destructor Documentation

10.166.2.1 gazebo::common::Video::Video ()

Constructor.

10.166.2.2 virtual gazebo::common::Video::~~Video () [virtual]

Destructor.

10.166.3 Member Function Documentation

10.166.3.1 int gazebo::common::Video::GetHeight () const

Get the height of the video in pixels.

Returns

the height

10.166.3.2 bool gazebo::common::Video::GetNextFrame (unsigned char **_buffer)

Get the next frame of the video.

Parameters

out	<i>_img</i>	Image (p. 349) in which the frame is stored
-----	-------------	--

Returns

false if HAVE_FFmpeg is not defined, true otherwise

10.166.3.3 `int gazebo::common::Video::GetWidth () const`

Get the width of the video in pixels.

Returns

the width

10.166.3.4 `bool gazebo::common::Video::Load (const std::string & _filename)`

Load a video file.

Parameters

<code>in</code>	<code>_filename</code>	Full path of the video file
-----------------	------------------------	-----------------------------

Returns

false if HAVE_FFmpeg is not defined or if a video stream can't be found

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

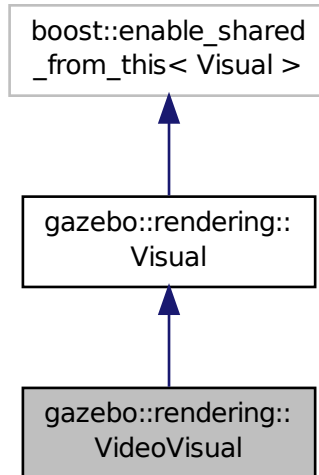
- [Video.hh](#)

10.167 gazebo::rendering::VideoVisual Class Reference

A visual element that displays a video as a texture.

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

Inheritance diagram for gazebo::rendering::VideoVisual:



Public Member Functions

- **VideoVisual** (const std::string &_name, **VisualPtr** _parent)
Constructor.
- virtual ~**VideoVisual** ()
Destructor.

Additional Inherited Members

10.167.1 Detailed Description

A visual element that displays a video as a texture.

10.167.2 Constructor & Destructor Documentation

10.167.2.1 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.

10.167.2.2 virtual gazebo::rendering::VideoVisual::~~VideoVisual () [virtual]

Destructor.

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

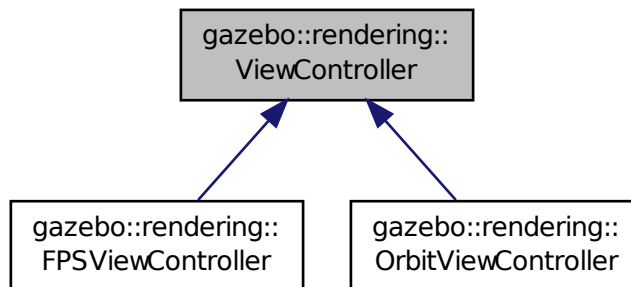
- **VideoVisual.hh**

10.168 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. 149).*

Protected Attributes

- **UserCameraPtr camera**

Pointer to the camera to control.

- bool **enabled**

True if enabled.

- std::string **typeString**

Type of view controller.

10.168.1 Detailed Description

Base class for view controllers.

10.168.2 Constructor & Destructor Documentation

10.168.2.1 gazebo::rendering::ViewController::ViewController (UserCameraPtr `_camera`)

Constructor.

Parameters

in	<code>_camera</code>	The user camera to controll.
----	----------------------	------------------------------

10.168.2.2 virtual gazebo::rendering::ViewController::~~ViewController () [virtual]

Destructor.

10.168.3 Member Function Documentation

10.168.3.1 std::string gazebo::rendering::ViewController::GetTypeString () const

Get the type of view controller.

Returns

The view controller type string.

10.168.3.2 virtual void gazebo::rendering::ViewController::HandleKeyPressEvent (const std::string & `_key`) [pure virtual]

Handle a key press event.

Parameters

in	_key	The key that was pressed.
----	------	---------------------------

Implemented in **gazebo::rendering::OrbitViewController** (p.519), and **gazebo::rendering::FPSViewController** (p.306).

10.168.3.3 virtual void gazebo::rendering::ViewController::HandleKeyReleaseEvent (const std::string & _key) [pure virtual]

Handle a key release event.

Parameters

in	_key	The key that was released.
----	------	----------------------------

Implemented in **gazebo::rendering::OrbitViewController** (p.520), and **gazebo::rendering::FPSViewController** (p.307).

10.168.3.4 virtual void gazebo::rendering::ViewController::HandleMouseEvent (const common::MouseEvent & _event) [pure virtual]

Handle a mouse event.

Parameters

in	_event	The mouse position.
----	--------	---------------------

Implemented in **gazebo::rendering::OrbitViewController** (p.520), and **gazebo::rendering::FPSViewController** (p.307).

10.168.3.5 virtual void gazebo::rendering::ViewController::Init () [pure virtual]

Initialize the view controller.

Implemented in **gazebo::rendering::OrbitViewController** (p.520), and **gazebo::rendering::FPSViewController** (p.307).

10.168.3.6 virtual void gazebo::rendering::ViewController::Init (const math::Vector3 & _focalPoint) [virtual]

Initialize with a focus point.

Parameters

in	_focalPoint	The point to look at.
----	-------------	-----------------------

Reimplemented in **gazebo::rendering::OrbitViewController** (p.520).

10.168.3.7 void gazebo::rendering::ViewController::SetEnabled (bool _value)

Set whether the controller is enabled.

Parameters

in	<code>_value</code>	True if the controller is enabled.
----	---------------------	------------------------------------

10.168.3.8 `virtual void gazebo::rendering::ViewController::Update ()` [pure virtual]

Update the controller, which should update the position of the **Camera** (p. 149).

Implemented in **`gazebo::rendering::OrbitViewController`** (p.521), and **`gazebo::rendering::FPSViewController`** (p.307).

10.168.4 Member Data Documentation

10.168.4.1 `UserCameraPtr gazebo::rendering::ViewController::camera` [protected]

Pointer to the camera to control.

10.168.4.2 `bool gazebo::rendering::ViewController::enabled` [protected]

True if enabled.

10.168.4.3 `std::string gazebo::rendering::ViewController::typeString` [protected]

Type of view controller.

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

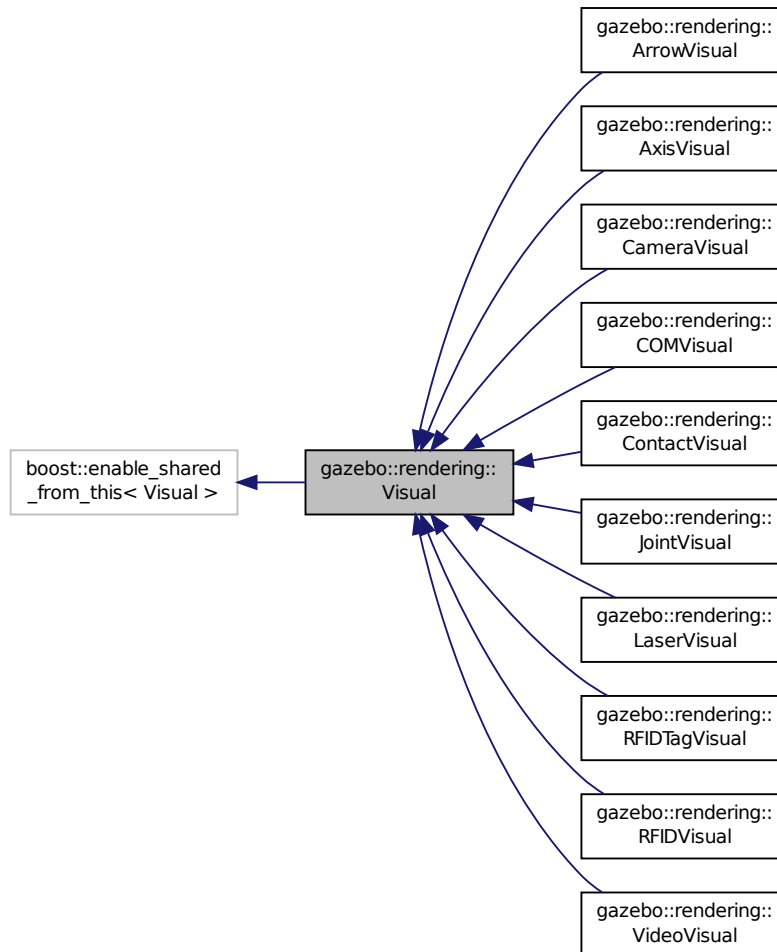
- **`ViewController.hh`**

10.169 `gazebo::rendering::Visual` Class Reference

A renderable object.

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


Inheritance diagram for gazebo::rendering::Visual:



Public Member Functions

- **Visual** (const std::string &_name, **VisualPtr** _parent, bool _useRTShader=true)
Constructor.
- **Visual** (const std::string &_name, **ScenePtr** _scene, bool _useRTShader=true)
Constructor.
- virtual ~**Visual** ()
Destructor.
- void **AttachAxes** ()
Attach visualization axes.
- void **AttachLineVertex** (**DynamicLines** *_line, unsigned int _index)
Attach a vertex of a line to the position of the visual.
- Ogre::MovableObject * **AttachMesh** (const std::string &_meshName, const std::string &_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 **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. 98).*
- **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. 98).*

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. 98).*

10.169.1 Detailed Description

A renderable object.

10.169.2 Constructor & Destructor Documentation

10.169.2.1 `gazebo::rendering::Visual::Visual (const std::string & _name, VisualPtr _parent, bool _useRTShader = true)`

Constructor.

Parameters

in	<i>_name</i>	Name of the visual.
in	<i>_parent</i>	Parent of the visual.
in	<i>_useRTShader</i>	True if the visual should use the real-time shader system (RTShader).

10.169.2.2 `gazebo::rendering::Visual::Visual (const std::string & _name, ScenePtr _scene, bool _useRTShader = true)`

Constructor.

Parameters

in	<code>_name</code>	Name of the visual.
in	<code>_scene</code>	Scene (p. 632) containing the visual.
in	<code>_useRTShader</code>	True if the visual should use the real-time shader system (RTShader).

10.169.2.3 `virtual gazebo::rendering::Visual::~Visual () [virtual]`

Destructor.

10.169.3 Member Function Documentation

10.169.3.1 `void gazebo::rendering::Visual::AttachAxes ()`

Attach visualization axes.

10.169.3.2 `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.

10.169.3.3 `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.

10.169.3.4 `void gazebo::rendering::Visual::AttachObject (Ogre::MovableObject * _obj)`

Attach a renewable object to the visual.

Parameters

in	<code>_obj</code>	A movable object to attach to the visual.
----	-------------------	---

10.169.3.5 void gazebo::rendering::Visual::AttachVisual (VisualPtr _vis)

Attach a visual to this visual.

Parameters

in	_vis	Visual (p. 828) to attach.
----	------	----------------------------

10.169.3.6 void gazebo::rendering::Visual::ClearParent ()

Clear parents.

10.169.3.7 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. 828).
in	_newParent	Parent of the cloned Visual (p. 828).

Returns

The visual.

10.169.3.8 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 pointer to the new dynamic line.

10.169.3.9 void gazebo::rendering::Visual::DeleteDynamicLine (DynamicLines * _line)

Delete a dynamic line.

Parameters

in	_line	Pointer to the line to delete.
----	-------	--------------------------------

10.169.3.10 void gazebo::rendering::Visual::DetachObjects ()

Detach all objects.

10.169.3.11 `void gazebo::rendering::Visual::DetachVisual (VisualPtr _vis)`

Detach a visual.

Parameters

<code>in</code>	<code>_vis</code>	Visual (p. 828) to detach.
-----------------	-------------------	-----------------------------------

10.169.3.12 `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.
-----------------	--------------------	-------------------------------

10.169.3.13 `void gazebo::rendering::Visual::DisableTrackVisual ()`

Disable tracking of a visual.

10.169.3.14 `void gazebo::rendering::Visual::EnableTrackVisual (VisualPtr _vis)`

Set one visual to track/follow another.

Parameters

<code>in</code>	<code>_vis</code>	Visual (p. 828) to track.
-----------------	-------------------	----------------------------------

10.169.3.15 `void gazebo::rendering::Visual::Fini ()`

Helper for the destructor.

10.169.3.16 `unsigned int gazebo::rendering::Visual::GetAttachedObjectCount () const`

Return the number of attached movable objects.

Returns

The number of attached movable objects.

10.169.3.17 `math::Box gazebo::rendering::Visual::GetBoundingBox () const`

Get the bounding box for the visual.

Returns

The bounding box in world coordinates.

10.169.3.18 `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. 837).

Parameters

<code>in</code>	<code>_index</code>	Index of the child to retrieve.
-----------------	---------------------	---------------------------------

Returns

Pointer to the child visual, NULL if index is invalid.

10.169.3.19 `unsigned int gazebo::rendering::Visual::GetChildCount ()`

Get the number of attached visuals.

Returns

The number of children.

10.169.3.20 `std::string gazebo::rendering::Visual::GetMaterialName () const`

Get the name of the material.

Returns

The name of the visual applied to this visual.

10.169.3.21 `std::string gazebo::rendering::Visual::GetName () const`

Get the name of the visual.

Returns

The name of the visual.

10.169.3.22 `std::string gazebo::rendering::Visual::GetNormalMap () const`

Get the normal map.

Returns

The name of the normal map material.

10.169.3.23 `VisualPtr gazebo::rendering::Visual::GetParent () const`

Get the parent visual, if one exists.

Returns

Pointer to the parent visual, NULL if no parent.

10.169.3.24 `math::Pose gazebo::rendering::Visual::GetPose () const`

Get the pose of the visual.

Returns

The **Visual** (p. 828)'s pose.

10.169.3.25 `math::Vector3 gazebo::rendering::Visual::GetPosition () const`

Get the position of the visual.

Returns

The visual's position.

10.169.3.26 `VisualPtr gazebo::rendering::Visual::GetRootVisual ()`

Get the root visual.

Returns

The root visual, which is one level below the world visual.

10.169.3.27 `math::Quaternion gazebo::rendering::Visual::GetRotation () const`

Get the rotation of the visual.

Returns

The visual's rotation.

10.169.3.28 `math::Vector3 gazebo::rendering::Visual::GetScale ()`

Get the scale.

Returns

The scaling factor.

10.169.3.29 `ScenePtr gazebo::rendering::Visual::GetScene () const`

Get current.

Returns

Pointer to the scene.

10.169.3.30 `Ogre::SceneNode* gazebo::rendering::Visual::GetSceneNode () const`

Return the scene Node of this visual entity.

Returns

The **Ogre** (p. 98) scene node.

10.169.3.31 `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".

10.169.3.32 `float gazebo::rendering::Visual::GetTransparency ()`

Get the transparency.

Returns

The transparency.

10.169.3.33 `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. 1008)

GZ_VISIBILITY_GUI (p. 1008)

GZ_VISIBILITY_NOT_SELECTABLE (p. 1008)

10.169.3.34 `bool gazebo::rendering::Visual::GetVisible () const`

Get whether the visual is visible.

Returns

True if the visual is visible.

10.169.3.35 `math::Pose gazebo::rendering::Visual::GetWorldPose () const`

Get the global pose of the node.

Returns

The pose in the world coordinate frame.

10.169.3.36 `bool gazebo::rendering::Visual::HasAttachedObject (const std::string & _name)`

Returns true if an object with `_name` is attached.

Parameters

in	<code>_name</code>	Name of an object to find.
----	--------------------	----------------------------

10.169.3.37 `void gazebo::rendering::Visual::Init ()`

Helper for the constructor.

10.169.3.38 `void gazebo::rendering::Visual::InsertMesh (const std::string & _meshName)`

Insert a mesh into **Ogre** (p. 98).

Parameters

in	<code>_meshName</code>	Name of the mesh to insert.
----	------------------------	-----------------------------

10.169.3.39 `static void gazebo::rendering::Visual::InsertMesh (const common::Mesh * _mesh) [static]`

Insert a mesh into **Ogre** (p. 98).

Parameters

in	<code>_mesh</code>	Pointer to the mesh to insert.
----	--------------------	--------------------------------

10.169.3.40 `bool gazebo::rendering::Visual::IsPlane () const`

Return true if the visual is a plane.

Returns

True if a plane.

10.169.3.41 `bool gazebo::rendering::Visual::IsStatic () const`

Return true if the visual is a static geometry.

Returns

True if the visual is static.

10.169.3.42 `void gazebo::rendering::Visual::Load (sdf::ElementPtr _sdf)`

Load the visual with a set of parameters.

Parameters

in	_sdf	Load from an SDF element.
----	------	---------------------------

10.169.3.43 `virtual void gazebo::rendering::Visual::Load () [virtual]`

Load the visual with default parameters.

Reimplemented in **gazebo::rendering::ArrowVisual** (p. 120), and **gazebo::rendering::AxisVisual** (p. 122).

10.169.3.44 `void gazebo::rendering::Visual::LoadFromMsg (ConstVisualPtr & _msg)`

Load from a message.

Parameters

in	_msg	A visual message.
----	------	-------------------

10.169.3.45 `void gazebo::rendering::Visual::LoadPlugin (const std::string & _filename, const std::string & _name, sdf::ElementPtr _sdf)`

Load a plugin.

Parameters

_filename	The filename of the plugin
_name	A unique name for the plugin
_sdf	The SDF to pass into the plugin.

10.169.3.46 `void gazebo::rendering::Visual::MakeStatic ()`

Make the visual objects static renderables.

10.169.3.47 `void gazebo::rendering::Visual::MoveToPosition (const math::Pose & _pose, double _time)`

Move to a pose and over a given time.

Parameters

in	_pose	Pose the visual will end at.
in	_time	Time it takes the visual to move to the pose.

10.169.3.48 `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.

10.169.3.49 `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
--	--------------------	---

10.169.3.50 `void gazebo::rendering::Visual::SetAmbient (const common::Color & _color)`

Set the ambient color of the visual.

Parameters

in	<code>_color</code>	The ambient color.
----	---------------------	--------------------

10.169.3.51 `void gazebo::rendering::Visual::SetCastShadows (bool _shadows)`

Set whether the visual should cast shadows.

Parameters

in	<code>_shadows</code>	True to enable shadows.
----	-----------------------	-------------------------

10.169.3.52 `void gazebo::rendering::Visual::SetDiffuse (const common::Color & _color)`

Set the diffuse color of the visual.

Parameters

in	<code>_color</code>	Set the diffuse color.
----	---------------------	------------------------

10.169.3.53 `virtual void gazebo::rendering::Visual::SetEmissive (const common::Color & _color)` [virtual]

Set the emissive value.

Parameters

in	<code>_color</code>	The emissive color.
----	---------------------	---------------------

Reimplemented in **`gazebo::rendering::LaserVisual`** (p. 392).

10.169.3.54 `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	<code>_highlighted</code>	True to enable the highlighting.
----	---------------------------	----------------------------------

10.169.3.55 `void gazebo::rendering::Visual::SetMaterial (const std::string & _materialName, bool _unique = true)`

Set the material.

Parameters

in	<code>_materialName</code>	The name of the material.
in	<code>_unique</code>	True to make the material unique, which allows the material to change without changing materials that originally had the same name.

10.169.3.56 `void gazebo::rendering::Visual::SetName (const std::string & _name)`

Set the name of the visual.

Parameters

in	<code>_name</code>	Name of the visual
----	--------------------	--------------------

10.169.3.57 `void gazebo::rendering::Visual::SetNormalMap (const std::string & _nmap)`

Set the normal map.

Parameters

in	<code>_nmap</code>	Name of the normal map material.
----	--------------------	----------------------------------

10.169.3.58 `void gazebo::rendering::Visual::SetPose (const math::Pose & _pose)`

Set the pose of the visual.

Parameters

in	<code>_pose</code>	The new pose of the visual.
----	--------------------	-----------------------------

10.169.3.59 `void gazebo::rendering::Visual::SetPosition (const math::Vector3 & _pos)`

Set the position of the visual.

Parameters

in	_pos	The position to set the visual to.
----	------	------------------------------------

10.169.3.60 `void gazebo::rendering::Visual::SetRibbonTrail (bool _value, const common::Color & _initialColor, const common::Color & _changeColor)`

True on or off a ribbon trail.

Parameters

in	_value	True to enable ribbon trail.
in	_initialColor	The initial color of the ribbon trail.
in	_changeColor	Color to change too as the trail grows.

10.169.3.61 `void gazebo::rendering::Visual::SetRotation (const math::Quaternion & _rot)`

Set the rotation of the visual.

Parameters

in	_rot	The rotation of the visual.
----	------	-----------------------------

10.169.3.62 `void gazebo::rendering::Visual::SetScale (const math::Vector3 & _scale)`

Set the scale.

Parameters

in	_scale	The scaling factor for the visual.
----	--------	------------------------------------

10.169.3.63 `void gazebo::rendering::Visual::SetScene (ScenePtr _scene)`

Set current scene.

Parameters

in	_scene	Pointer to the scene.
----	--------	-----------------------

10.169.3.64 `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".
----	-------	---

10.169.3.65 void gazebo::rendering::Visual::SetSkeletonPose (const msgs::PoseAnimation & *_pose*)

Set animation skeleton pose.

Parameters

in	_pose	Skelton message
----	-------	-----------------

10.169.3.66 void gazebo::rendering::Visual::SetSpecular (const common::Color & *_color*)

Set the specular color of the visual.

Parameters

in	_color	Specular color.
----	--------	-----------------

10.169.3.67 void gazebo::rendering::Visual::SetTransparency (float *_trans*)

Set the transparency.

Parameters

in	_trans	The transparency, between 0 and 1 where 0 is no transparency.
----	--------	---

10.169.3.68 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. 1008)

GZ_VISIBILITY_GUI (p. 1008)

GZ_VISIBILITY_NOT_SELECTABLE (p. 1008)

10.169.3.69 void gazebo::rendering::Visual::SetVisible (bool *_visible*, bool *_cascade* = true)

Set whether the visual is visible.

Parameters

in	<code>_visible</code>	set this node visible.
in	<code>_cascade</code>	setting this parameter in children too.

10.169.3.70 `void gazebo::rendering::Visual::SetWorldPose (const math::Pose _pose)`

Set the world pose of the visual.

Parameters

in	<code>_pose</code>	Pose of the visual in the world coordinate frame.
----	--------------------	---

10.169.3.71 `void gazebo::rendering::Visual::SetWorldPosition (const math::Vector3 & _pos)`

Set the world linear position of the visual.

Parameters

in	<code>_pose</code>	Position in the world coordinate frame.
----	--------------------	---

10.169.3.72 `void gazebo::rendering::Visual::SetWorldRotation (const math::Quaternion & _rot)`

Set the world orientation of the visual.

Parameters

in	<code>_rot</code>	Rotation in the world coordinate frame.
----	-------------------	---

10.169.3.73 `void gazebo::rendering::Visual::ShowBoundingBox ()`

Display the bounding box visual.

10.169.3.74 `void gazebo::rendering::Visual::ShowCollision (bool _show)`

Display the collision visuals.

Parameters

in	<code>_show</code>	True to show visuals labeled as collision objects.
----	--------------------	--

10.169.3.75 `void gazebo::rendering::Visual::ShowCOM (bool _show)`

Display Center of Mass visuals.

Parameters

in	<code>_show</code>	True to show center of mass visualizations.
----	--------------------	---

10.169.3.76 void gazebo::rendering::Visual::ShowJoints (bool *_show*)

Display joint visuals.

Parameters

in	<i>_show</i>	True to show joint visualizations.
----	--------------	------------------------------------

10.169.3.77 void gazebo::rendering::Visual::ShowSkeleton (bool *_show*)

Display the skeleton visuals.

Parameters

in	<i>_show</i>	True to show skeleton visuals.
----	--------------	--------------------------------

10.169.3.78 void gazebo::rendering::Visual::ToggleVisible ()

Toggle whether this visual is visible.

10.169.3.79 void gazebo::rendering::Visual::Update ()

Update the visual.

10.169.3.80 void gazebo::rendering::Visual::UpdateFromMsg (ConstVisualPtr & *_msg*)

Update a visual based on a message.

Parameters

in	<i>_msg</i>	The visual message.
----	-------------	---------------------

10.169.4 Member Data Documentation

10.169.4.1 VisualPtr gazebo::rendering::Visual::parent [protected]

Parent visual.

10.169.4.2 ScenePtr gazebo::rendering::Visual::scene [protected]

Pointer to the visual's scene.

10.169.4.3 Ogre::SceneNode* gazebo::rendering::Visual::sceneNode [protected]

Pointer to the visual's scene node in **Ogre** (p. 98).

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

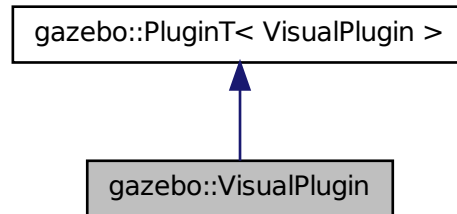
- **Visual.hh**

10.170 gazebo::VisualPlugin Class Reference

A plugin loaded within the gzserver on startup.

```
#include <Plugin.hh>
```

Inheritance diagram for gazebo::VisualPlugin:



Public Member Functions

- **VisualPlugin** ()
- virtual void **Init** ()
Initialize the plugin.
- virtual void **Load** (**rendering::VisualPtr** _visual, **sdf::ElementPtr** _sdf)=0
Load function.
- virtual void **Reset** ()
Override this method for custom plugin reset behavior.

Additional Inherited Members

10.170.1 Detailed Description

A plugin loaded within the gzserver on startup.

See [reference](#).

10.170.2 Constructor & Destructor Documentation

10.170.2.1 gazebo::VisualPlugin::VisualPlugin () [inline]

References [gazebo::PluginT< VisualPlugin >::type](#), and [gazebo::VISUAL_PLUGIN](#).

10.170.3 Member Function Documentation

10.170.3.1 `virtual void gazebo::VisualPlugin::Init () [inline],[virtual]`

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.170.3.2 `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.

10.170.3.3 `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:

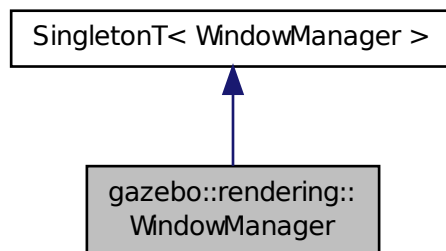
- `common/Plugin.hh`

10.171 gazebo::rendering::WindowManager Class Reference

Class to manage render windows.

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

Inheritance diagram for gazebo::rendering::WindowManager:



Public Member Functions

- int **CreateWindow** (const std::string &_ogreHandle, uint32_t _width, uint32_t _height)

Create a window.

- void **Fini** ()
Shutdown all the windows.
- float **GetAvgFPS** (uint32_t _id)
Get the average FPS.
- uint32_t **GetTriangleCount** (uint32_t _id)
Get the triangle count.
- Ogre::RenderWindow * **GetWindow** (uint32_t _id)
Get the render window associated with the given id.
- void **Moved** (uint32_t _id)
*Tells **Ogre** (p. 98) 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

10.171.1 Detailed Description

Class to manage render windows.

10.171.2 Member Function Documentation

10.171.2.1 int gazebo::rendering::WindowManager::CreateWindow (const std::string & _ogreHandle, uint32_t _width, uint32_t _height)

Create a window.

Parameters

in	<i>_ogreHandle</i>	String representing the ogre window handle.
in	<i>_width</i>	Width of the window in pixels.
in	<i>_height</i>	Height of the window in pixels.

10.171.2.2 void gazebo::rendering::WindowManager::Fini ()

Shutdown all the windows.

10.171.2.3 float gazebo::rendering::WindowManager::GetAvgFPS (uint32_t _id)

Get the average FPS.

Parameters

in	<i>_id</i>	ID of the window.
----	------------	-------------------

Returns

The frames per second.

10.171.2.4 `uint32_t gazebo::rendering::WindowManager::GetTriangleCount (uint32_t _id)`

Get the triangle count.

Parameters

<code>in</code>	<code>_id</code>	ID of the window.
-----------------	------------------	-------------------

Returns

The triangle count.

10.171.2.5 `Ogre::RenderWindow* gazebo::rendering::WindowManager::GetWindow (uint32_t _id)`

Get the render window associated with the given id.

Parameters

<code>in</code>	<code>_id</code>	ID of the window.
-----------------	------------------	-------------------

Returns

Pointer to the render window, NULL if the id is invalid.

10.171.2.6 `void gazebo::rendering::WindowManager::Moved (uint32_t _id)`

Tells **Ogre** (p. 98) the window has moved, and needs updating.

Parameters

<code>in</code>	<code>_id</code>	ID of the window.
-----------------	------------------	-------------------

10.171.2.7 `void gazebo::rendering::WindowManager::Resize (uint32_t _id, int _width, int _height)`

Resize a window.

Parameters

<code>in</code>	<code>_id</code>	Id of the window to resize.
<code>in</code>	<code>_width</code>	New width of the window.
<code>in</code>	<code>_height</code>	New height of the window.

10.171.2.8 void gazebo::rendering::WindowManager::SetCamera (int *_windowId*, CameraPtr *_camera*)

Attach a camera to a window.

Parameters

in	<i>_windowId</i>	Id of the window to add the camera to.
in	<i>_camera</i>	Pointer to the camera to attach.

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

- **WindowManager.hh**

10.172 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.

10.172.1 Detailed Description

Draws a wireframe box.

10.172.2 Constructor & Destructor Documentation

10.172.2.1 gazebo::rendering::WireBox::WireBox (VisualPtr *_parent*, const math::Box &*_box*) [explicit]

Constructor.

Parameters

in	<i>_box</i>	Dimension of the box to draw.
----	-------------	-------------------------------

10.172.2.2 gazebo::rendering::WireBox::~~WireBox ()

Destructor.

10.172.3 Member Function Documentation

10.172.3.1 void gazebo::rendering::WireBox::Init (const math::Box & _box)

Builds the wireframe line list.

Parameters

in	_box	Box to build a wireframe from.
----	------	--------------------------------

10.172.3.2 void gazebo::rendering::WireBox::SetVisible (bool _visible)

Set the visibility of the box.

Parameters

in	_visible	True to make the box visible, False to hide.
----	----------	--

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

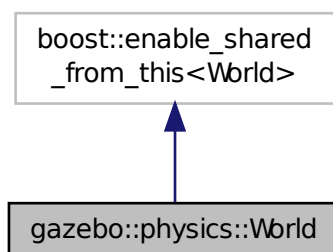
- **WireBox.hh**

10.173 gazebo::physics::World Class Reference

The world provides access to all other object within a simulated environment.

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

Inheritance diagram for gazebo::physics::World:



Public Member Functions

- **World** (const std::string &_name="")
Constructor.
- **~World** ()

- Destructor.*
- void **Clear** ()
Remove all entities from the world.
 - void **DisableAllModels** ()
Disable all links in all the models.
 - void **EnableAllModels** ()
Enable all links in all the models.
 - void **EnablePhysicsEngine** (bool _enable)
enable/disable physics engine during World::Update.
 - void **Fini** ()
Finalize the world.
 - **BasePtr GetByName** (const std::string &_name)
Get an element by name.
 - bool **GetEnablePhysicsEngine** ()
check if physics engine is enabled/disabled.
 - **EntityPtr GetEntity** (const std::string &_name)
*Get a pointer to an **Entity** (p. 265) based on a name.*
 - **EntityPtr GetEntityBelowPoint** (const math::Vector3 &_pt)
Get the nearest entity below a point.
 - **EntityPtr GetEntityByName** (const std::string &_name) **GAZEBO_DEPRECATED**
Deprecated.
 - **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.
 - **ModelPtr GetModelByName** (const std::string &name) **GAZEBO_DEPRECATED**
Deprecated.
 - 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. 265).*
 - 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. 737).*

- **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 **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. 265) 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. 275) on the **physics::Link** (p. 398) in **World::Update**.

10.173.1 Detailed Description

The world provides access to all other object within a simulated environment.

The **World** (p. 853) is the container for all models and their components (links, joints, sensors, plugins, etc), and **World-Plugin** (p. 864) instances. Many core function are also handled in the **World** (p. 853), including physics update, model updates, and message processing.

10.173.2 Constructor & Destructor Documentation

10.173.2.1 `gazebo::physics::World::World (const std::string & _name = " ") [explicit]`

Constructor.

Constructor for the **World** (p. 853). Must specify a unique name.

Parameters

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

10.173.2.2 `gazebo::physics::World::~~World ()`

Destructor.

10.173.3 Member Function Documentation

10.173.3.1 `void gazebo::physics::World::Clear ()`

Remove all entities from the world.

10.173.3.2 `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.

10.173.3.3 `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.

10.173.3.4 `void gazebo::physics::World::EnablePhysicsEngine (bool _enable) [inline]`

enable/disable physics engine during World::Update.

Parameters

<code>in</code>	<code><i>_enable</i></code>	True to enable the physics engine.
-----------------	-----------------------------	------------------------------------

10.173.3.5 `void gazebo::physics::World::Fini ()`

Finalize the world.

Call this function to tear-down the world.

10.173.3.6 `BasePtr gazebo::physics::World::GetByName (const std::string & _name)`

Get an element by name.

Searches the list of entities, and return a pointer to the model with a matching *_name*.

Parameters

<code>in</code>	<code><i>_name</i></code>	The name of the Model (p. 460) to find.
-----------------	---------------------------	--

Returns

A pointer to the entity, or NULL if no entity was found.

10.173.3.7 `bool gazebo::physics::World::GetEnablePhysicsEngine () [inline]`

check if physics engine is enabled/disabled.

Parameters

<code><i>True</i></code>	if the physics engine is enabled.
--------------------------	-----------------------------------

10.173.3.8 `EntityPtr gazebo::physics::World::GetEntity (const std::string & _name)`

Get a pointer to an **Entity** (p. 265) based on a name.

This function is the same as GetByName, but limits the search to only Entities.

Parameters

<code>in</code>	<code><i>_name</i></code>	The name of the Entity (p. 265) to find.
-----------------	---------------------------	---

Returns

A pointer to the **Entity** (p. 265), or NULL if no **Entity** (p. 265) was found.

10.173.3.9 EntityPtr gazebo::physics::World::GetEntityBelowPoint (const math::Vector3 & _pt)

Get the nearest entity below a point.

Projects a Ray down (-Z axis) starting at the given point. The first entity hit by the Ray is returned.

Parameters

in	<code>_pt</code>	The 3D point to search below
----	------------------	------------------------------

Returns

A pointer to nearest **Entity** (p. 265), NULL if none is found.

10.173.3.10 EntityPtr gazebo::physics::World::GetEntityByName (const std::string & _name)

Deprecated.

10.173.3.11 ModelPtr gazebo::physics::World::GetModel (unsigned int _index) const

Get a model based on an index.

Get a **Model** (p. 460) using an index, where index must be greater than zero and less than **World::GetModelCount()** (p. 859)

Parameters

in	<code>_index</code>	The index of the model [0..GetModelCount)
----	---------------------	---

Returns

A pointer to the **Model** (p. 460). NULL if `_index` is invalid.

10.173.3.12 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	<code>_name</code>	The name of the Model (p. 460) to find.
----	--------------------	--

Returns

A pointer to the **Model** (p. 460), or NULL if no model was found.

10.173.3.13 ModelPtr gazebo::physics::World::GetModelBelowPoint (const math::Vector3 & _pt)

Get the nearest model below a point.

This function makes use of **World::GetEntityBelowPoint** (p. 858).

Parameters

<code>in</code>	<code>_pt</code>	The 3D point to search below.
-----------------	------------------	-------------------------------

Returns

A pointer to nearest **Model** (p. 460), NULL if none is found.

10.173.3.14 **ModelPtr** gazebo::physics::World::GetModelByName (const std::string & name)

Deprecated.

10.173.3.15 unsigned int gazebo::physics::World::GetModelCount () const

Get the number of models.

Returns

The number of models in the **World** (p. 853).

10.173.3.16 **Model_V** gazebo::physics::World::GetModels () const

Get a list of all the models.

Returns

A list of all the Models in the world.

10.173.3.17 std::string gazebo::physics::World::GetName () const

Get the name of the world.

Returns

The name of the world.

10.173.3.18 **common::Time** gazebo::physics::World::GetPauseTime () const

Get the amount of time simulation has been paused.

Returns

The pause time.

10.173.3.19 **PhysicsEnginePtr** gazebo::physics::World::GetPhysicsEngine () const

Return the physics engine.

Get a pointer to the physics engine used by the world.

Returns

Pointer to the physics engine.

10.173.3.20 `common::Time gazebo::physics::World::GetRealTime () const`

Get the real time (elapsed time).

Returns

The real time.

10.173.3.21 `EntityPtr gazebo::physics::World::GetSelectedEntity () const`

Get the selected **Entity** (p. 265).

The selected entity is set via the GUI.

Returns

A point to the **Entity** (p. 265), NULL if nothing is selected.

10.173.3.22 `boost::mutex* gazebo::physics::World::GetSetWorldPoseMutex () const` `[inline]`

Get the set world pose mutex.

Returns

Pointer to the mutex.

10.173.3.23 `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. 737).

Returns

The current simulation time

10.173.3.24 `common::Time gazebo::physics::World::GetStartTime () const`

Get the wall time simulation was started.

Returns

The start time.

10.173.3.25 `void gazebo::physics::World::Init ()`

Initialize the world.

This is called after Load.

10.173.3.26 `void gazebo::physics::World::InsertModelFile (const std::string & _sdfFilename)`

Insert a model from an SDF file.

Spawns a model into the world base on and SDF file.

Parameters

<code>in</code>	<code>_sdfFilename</code>	The name of the SDF file (including path).
-----------------	---------------------------	--

10.173.3.27 `void gazebo::physics::World::InsertModelSDF (const sdf::SDF & _sdf)`

Insert a model using SDF.

Spawns a model into the world base on and SDF object.

Parameters

<code>in</code>	<code>_sdf</code>	A reference to an SDF object.
-----------------	-------------------	-------------------------------

10.173.3.28 `void gazebo::physics::World::InsertModelString (const std::string & _sdfString)`

Insert a model from an SDF string.

Spawns a model into the world base on and SDF string.

Parameters

<code>in</code>	<code>_sdfString</code>	A string containing valid SDF markup.
-----------------	-------------------------	---------------------------------------

10.173.3.29 `bool gazebo::physics::World::IsPaused () const`

Returns the state of the simulation true if paused.

Returns

True if paused.

10.173.3.30 `void gazebo::physics::World::Load (sdf::ElementPtr _sdf)`

Load the world using SDF parameters.

Load a world from and SDF pointer.

Parameters

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

10.173.3.31 `void gazebo::physics::World::LoadPlugin (const std::string & _filename, const std::string & _name, sdf::ElementPtr _sdf)`

Load a plugin.

Parameters

<code>in</code>	<code>_filename</code>	The filename of the plugin.
<code>in</code>	<code>_name</code>	A unique name for the plugin.
<code>in</code>	<code>_sdf</code>	The SDF to pass into the plugin.

10.173.3.32 `void gazebo::physics::World::PrintEntityTree ()`

Print **Entity** (p. 265) tree.

Prints all the entities to stdout.

10.173.3.33 `void gazebo::physics::World::RemovePlugin (const std::string & _name)`

Remove a running plugin.

Parameters

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

10.173.3.34 `void gazebo::physics::World::Reset ()`

Reset time and model poses, configurations in simulation.

10.173.3.35 `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. 128).

Parameters

<code>in</code>	<code>_type</code>	The type of reset.
-----------------	--------------------	--------------------

10.173.3.36 `void gazebo::physics::World::ResetTime ()`

Reset simulation time back to zero.

10.173.3.37 `void gazebo::physics::World::Run ()`

Run the world in a thread.

Run the update loop.

10.173.3.38 void gazebo::physics::World::Save (const std::string & *_filename*)

Save a world to a file.

Save the current world and its state to a file.

Parameters

in	<i>_filename</i>	Name of the file to save into.
----	------------------	--------------------------------

10.173.3.39 void gazebo::physics::World::SetPaused (bool *_p*)

Set whether the simulation is paused.

Parameters

in	<i>_p</i>	True pauses the simulation. False runs the simulation.
----	-----------	--

10.173.3.40 void gazebo::physics::World::SetSimTime (const common::Time & *_t*)

Set the sim time.

Parameters

in	<i>_t</i>	The new simulation time
----	-----------	-------------------------

10.173.3.41 void gazebo::physics::World::SetState (const WorldState & *_state*)

Set the current world state.

Parameters

<i>_state</i>	The state to set the World (p. 853) to.	
---------------	--	--

10.173.3.42 void gazebo::physics::World::StepWorld (int *_steps*)

Step callback.

Parameters

in	<i>_steps</i>	The number of steps the World (p. 853) should take.
----	---------------	--

10.173.3.43 void gazebo::physics::World::Stop ()

Stop the world.

Stop the update loop.

10.173.3.44 `std::string gazebo::physics::World::StripWorldName (const std::string & _name) const`

Return a version of the name with "<world_name>::" removed.

Parameters

in	_name	Usually the name of an entity.
----	-------	--------------------------------

Returns

The stripped world name.

10.173.3.45 `void gazebo::physics::World::UpdateStateSDF ()`

Update the state SDF value from the current state.

10.173.4 Member Data Documentation

10.173.4.1 `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. 275) on the **physics::Link** (p. 398) in `World::Update`.

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

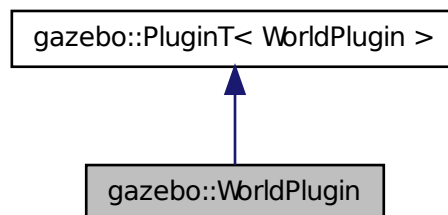
- **World.hh**

10.174 gazebo::WorldPlugin Class Reference

A plugin with access to **physics::World** (p. 853).

```
#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

10.174.1 Detailed Description

A plugin with access to **physics::World** (p. 853).

See [reference](#).

10.174.2 Constructor & Destructor Documentation

10.174.2.1 gazebo::WorldPlugin::WorldPlugin () [inline]

Constructor.

References [gazebo::PluginT< WorldPlugin >::type](#), and [gazebo::WORLD_PLUGIN](#).

10.174.2.2 virtual gazebo::WorldPlugin::~~WorldPlugin () [inline],[virtual]

Destructor.

10.174.3 Member Function Documentation

10.174.3.1 virtual void gazebo::WorldPlugin::Init () [inline],[virtual]

10.174.3.2 virtual void gazebo::WorldPlugin::Load (**physics::WorldPtr** _world, **sdf::ElementPtr** _sdf) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

in	<code>_world</code>	Pointer the World
in	<code>_sdf</code>	Pointer the the SDF element of the plugin.

10.174.3.3 virtual void gazebo::WorldPlugin::Reset () [inline],[virtual]

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

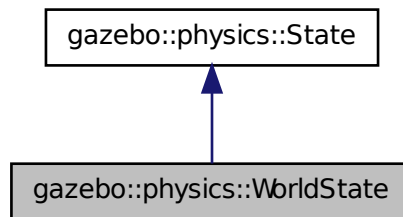
- `common/Plugin.hh`

10.175 gazebo::physics::WorldState Class Reference

Store state information of a `physics::World` (p. 853) 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.
- **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. 866) has a **ModelState** (p. 477) 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

10.175.1 Detailed Description

Store state information of a **physics::World** (p. 853) object.

Instances of this class contain the state of a **World** (p. 853) at a specific time. **World** (p. 853) state includes the state of all models, and their children.

10.175.2 Constructor & Destructor Documentation

10.175.2.1 gazebo::physics::WorldState::WorldState ()

Default constructor.

10.175.2.2 gazebo::physics::WorldState::WorldState (const **WorldPtr** _world) [explicit]

Constructor.

Generate a **WorldState** (p. 866) from an instance of a **World** (p. 853).

Parameters

in	<code>_world</code>	Pointer to a world
----	---------------------	--------------------

10.175.2.3 gazebo::physics::WorldState::WorldState (const sdf::ElementPtr _sdf) [explicit]

Constructor.

Build a **WorldState** (p. 866) from SDF data

Parameters

in	<code>_sdf</code>	SDF data to load a world state from.
----	-------------------	--------------------------------------

10.175.2.4 virtual gazebo::physics::WorldState::~~WorldState () [virtual]

Destructor.

10.175.3 Member Function Documentation

10.175.3.1 ModelState gazebo::physics::WorldState::GetModelState (unsigned int *_index*) const

Get a model state.

Get the state of a **Model** (p. 460) based on an index. The min index is and the max is **WorldState::GetModelStateCount()** (p. 868).

Parameters

in	<i>_index</i>	Index of the model.
----	---------------	---------------------

Returns

State (p. 702) of the requested **Model** (p. 460).

10.175.3.2 ModelState gazebo::physics::WorldState::GetModelState (const std::string & *_modelName*) const

Get a model state by model name.

Parameters

in	<i>_modelName</i>	Name of the model state to get.
----	-------------------	---------------------------------

Returns

The model state.

Exceptions

<i>common::Exception</i> (p. 303)	When the <i>_modelName</i> doesn't exist.
---	---

10.175.3.3 unsigned int gazebo::physics::WorldState::GetModelStateCount () const

Get the number of model states.

Returns the number of models in this instance.

Returns

Number of models.

10.175.3.4 const std::vector<ModelState>& gazebo::physics::WorldState::GetModelStates () const

Get the model states.

Returns

A vector of model states.

10.175.3.5 `bool gazebo::physics::WorldState::HasModelState (const std::string & _modelName) const`

Return true if **WorldState** (p. 866) has a **ModelState** (p. 477) 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. 477) exists.

10.175.3.6 `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.

10.175.3.7 `virtual void gazebo::physics::WorldState::Load (const sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Set a **WorldState** (p. 866) from an SDF element containing **WorldState** (p. 866) info.

Parameters

<code>in</code>	<code><i>_elem</i></code>	Pointer to the WorldState (p. 866) SDF element.
-----------------	---------------------------	--

Reimplemented from **gazebo::physics::State** (p. 704).

10.175.3.8 `WorldState gazebo::physics::WorldState::operator+ (const WorldState & _state) const`

Addition operator.

Parameters

<code>in</code>	<code><i>_pt</i></code>	A state to add.
-----------------	-------------------------	-----------------

Returns

The resulting state.

10.175.3.9 **WorldState** gazebo::physics::WorldState::operator- (const WorldState & *_state*) const

Subtraction operator.

Parameters

<i>in</i>	<i>_pt</i>	A state to subtract.
-----------	------------	----------------------

Returns

The resulting state.

10.175.3.10 **WorldState&** gazebo::physics::WorldState::operator= (const WorldState & *_state*)

Assignment operator.

Parameters

<i>in</i>	<i>_state</i>	State (p. 702) value
-----------	---------------	-----------------------------

Returns

Reference to this

10.175.4 Friends And Related Function Documentation

10.175.4.1 **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>	World (p. 853) state to output

Returns

the stream

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

- **WorldState.hh**

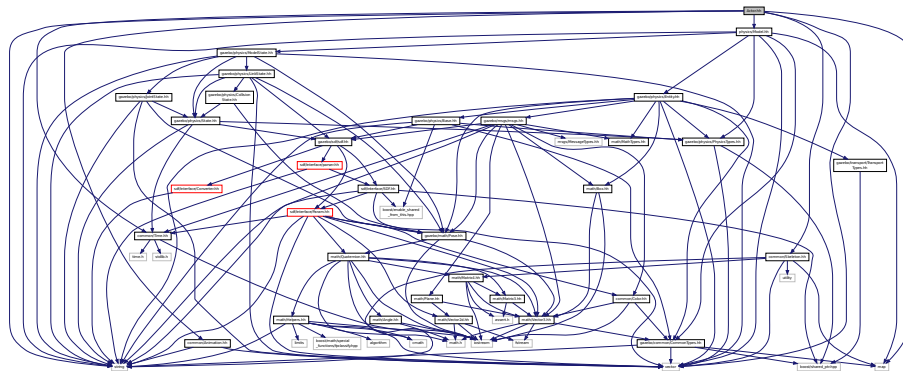
Chapter 11

File Documentation

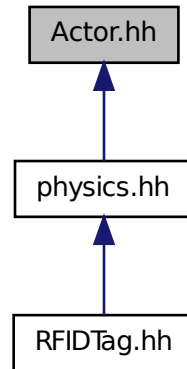
11.1 Actor.hh File Reference

```
#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. 101) class enables GPU based mesh model / skeleton scriptable animation.
- struct **gazebo::physics::TrajectoryInfo**

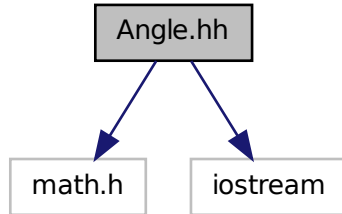
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::physics**
namespace for physics

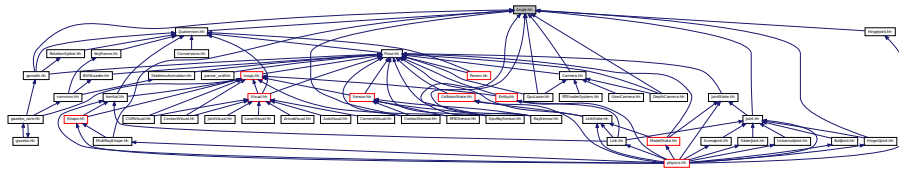
11.2 Angle.hh File Reference

```
#include <math.h>  
#include <iostream>
```

Include dependency graph for Angle.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Angle**
An angle and related functions.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

Macros

- #define **GZ_DTOR**(d) ((d) * M_PI / 180)
Converts degrees to radians.
- #define **GZ_NORMALIZE**(a) (atan2(sin(a), cos(a)))
Macro tha normalizes an angle in the range -Pi to Pi.
- #define **GZ_RTOD**(r) ((r) * 180 / M_PI)
Macro that converts radians to degrees.

11.2.1 Macro Definition Documentation

11.2.1.1 `#define GZ_DTOR(d) ((d) * M_PI / 180)`

Converts degrees to radians.

Parameters

<i>in</i>	<i>degrees</i>	
-----------	----------------	--

Returns

radians

11.2.1.2 `#define GZ_NORMALIZE(a) (atan2(sin(a), cos(a)))`

Macro tha normalizes an angle in the range -Pi to Pi.

Parameters

<i>in</i>	<i>angle</i>	
-----------	--------------	--

Returns

the angle, in range

11.2.1.3 `#define GZ_RTOD(r) ((r) * 180 / M_PI)`

Macro that converts radians to degrees.

Parameters

<i>in</i>	<i>radians</i>	
-----------	----------------	--

Returns

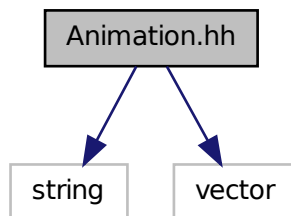
degrees

11.3 Animation.hh File Reference

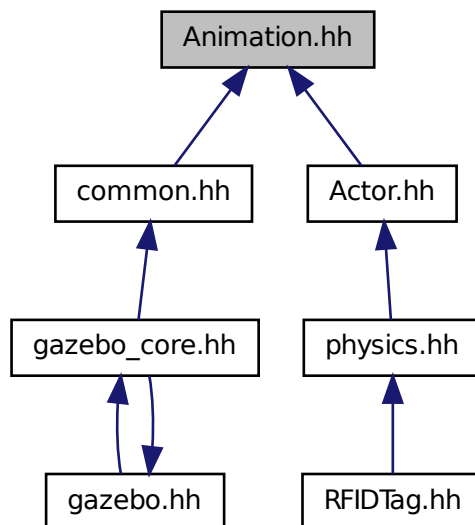
```
#include <string>
```

```
#include <vector>
```

Include dependency graph for Animation.hh:



This graph shows which files directly or indirectly include this file:



Classes

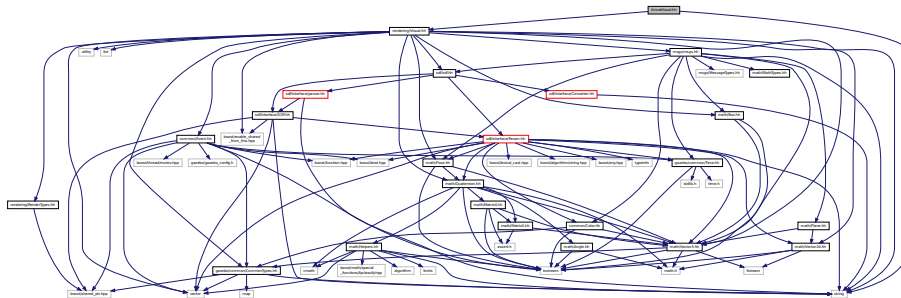
- class **gazebo::common::Animation**
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.
- class **gazebo::common::NumericAnimation**
A numeric animation.
- class **gazebo::common::PoseAnimation**
A pose animation.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::math**
Math namespace.

11.4 ArrowVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
Include dependency graph for ArrowVisual.hh:
```



Classes

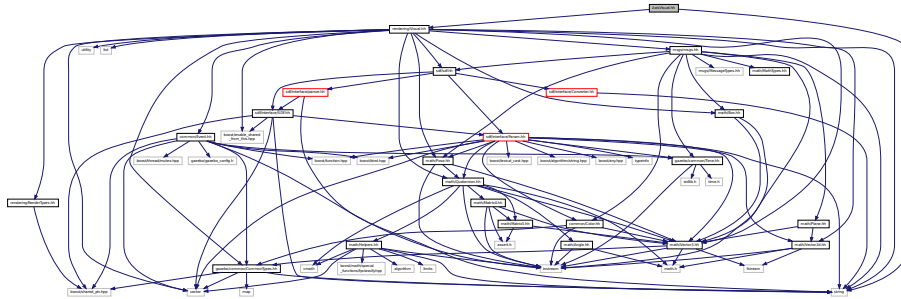
- class **gazebo::rendering::ArrowVisual**
Basic arrow visualization.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **ogre**

11.5 AxisVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
Include dependency graph for AxisVisual.hh:
```



Classes

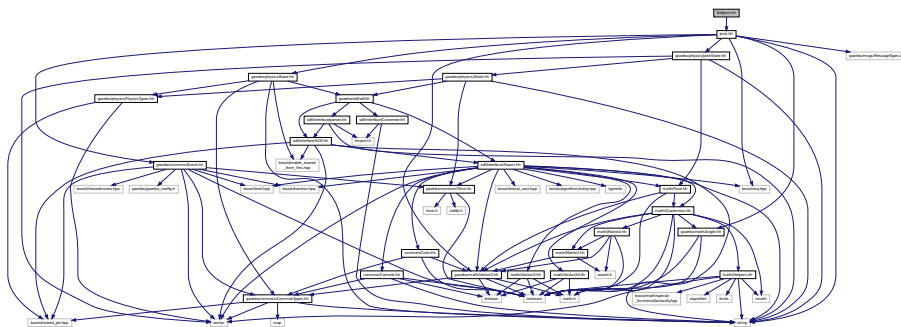
- class **gazebo::rendering::AxisVisual**
Basic axis visualization.

Namespaces

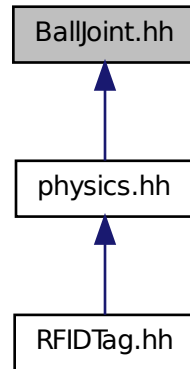
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.6 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. 125) class for a ball joint.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

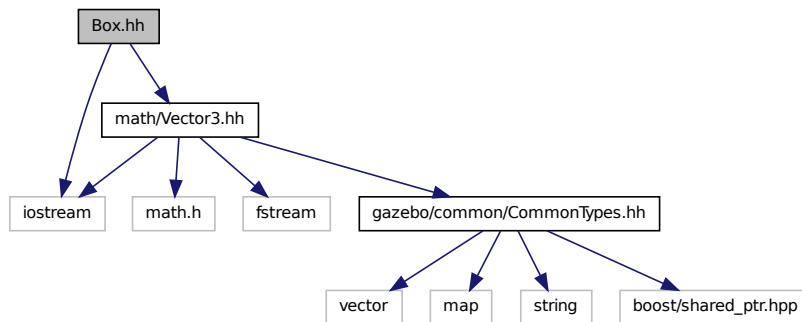
- namespace **gazebo::physics**

namespace for physics

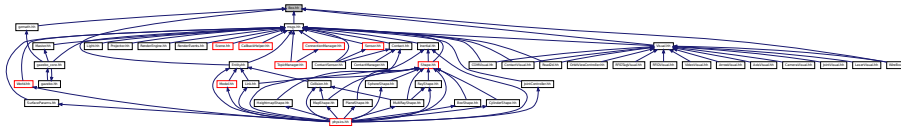
11.7 Base.hh File Reference

```
#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 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.

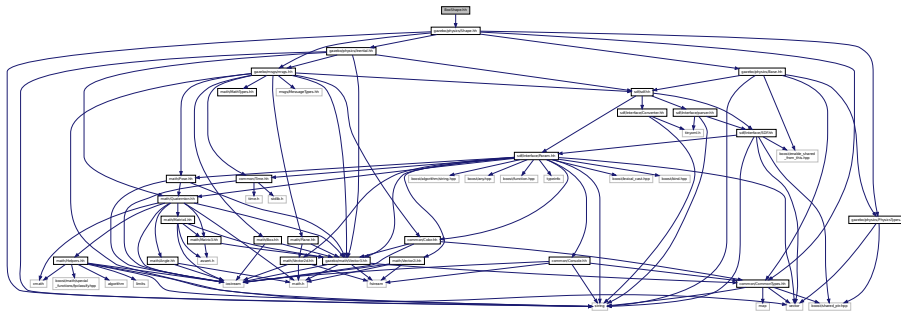
- namespace **gazebo::math**

Math namespace.

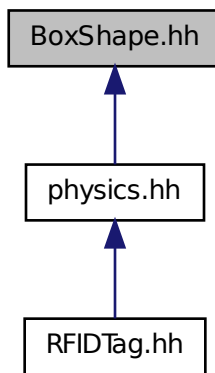
11.9 BoxShape.hh File Reference

```
#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.

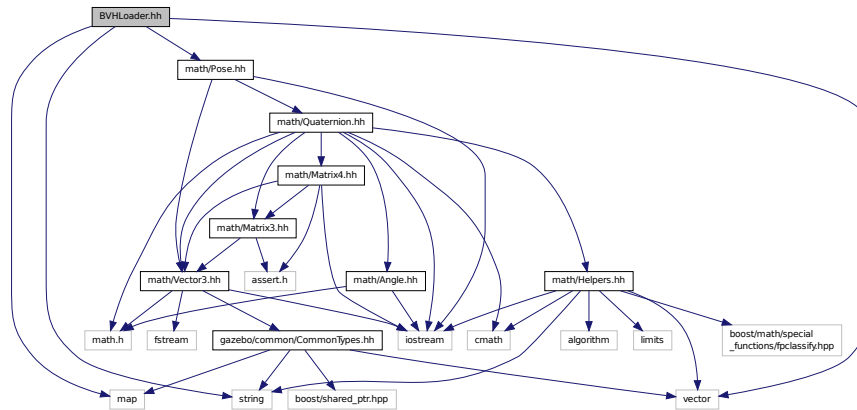
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

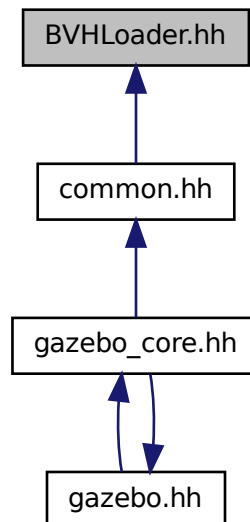
11.10 BVHLoader.hh File Reference

```
#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`

11.10.1 Macro Definition Documentation

11.10.1.1 `#define X_POSITION 0`

11.10.1.2 `#define X_ROTATION 3`

11.10.1.3 `#define Y_POSITION 1`

11.10.1.4 `#define Y_ROTATION 4`

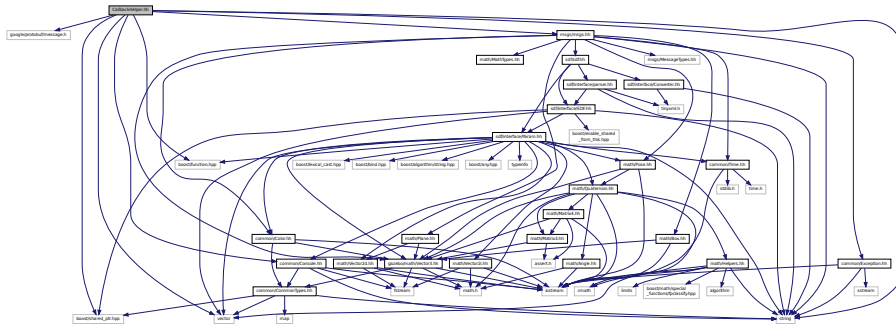
11.10.1.5 `#define Z_POSITION 2`

11.10.1.6 `#define Z_ROTATION 5`

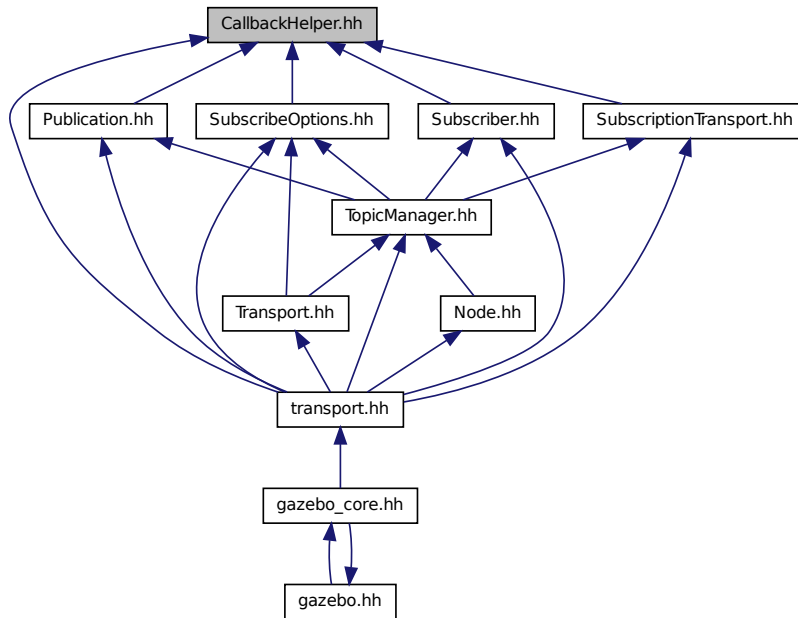
11.11 CallbackHelper.hh File Reference

```
#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 helper class to handle callbacks when messages arrive.
- class **gazebo::transport::CallbackHelperT < M >**
Callback helper Template.
- class **gazebo::transport::DebugCallbackHelper**
CallbackHelper (p. 144) subclass with debug facilities.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::transport**

Typedefs

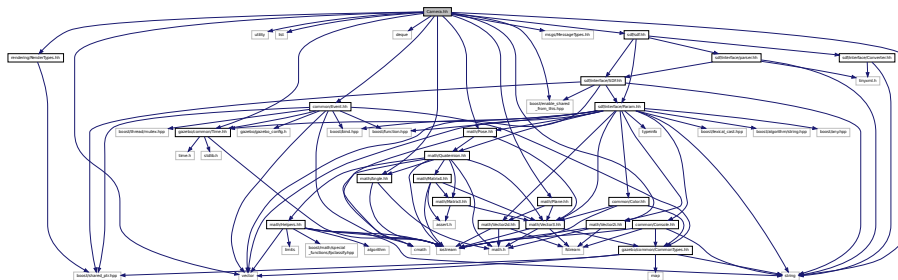
- typedef CallbackHelper * **gazebo::transport::CallbackHelperPtr**

*boost shared pointer to **transport::CallbackHelper** (p. 144)*

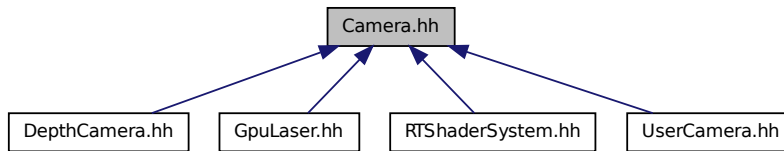
11.12 Camera.hh File Reference

```
#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**

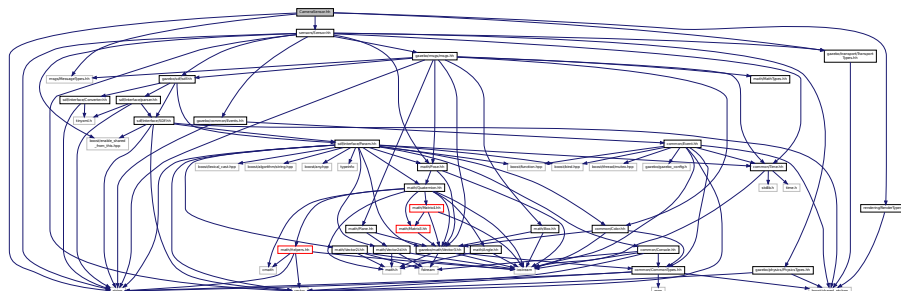
11.13 CameraSensor.hh File Reference

```

#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**

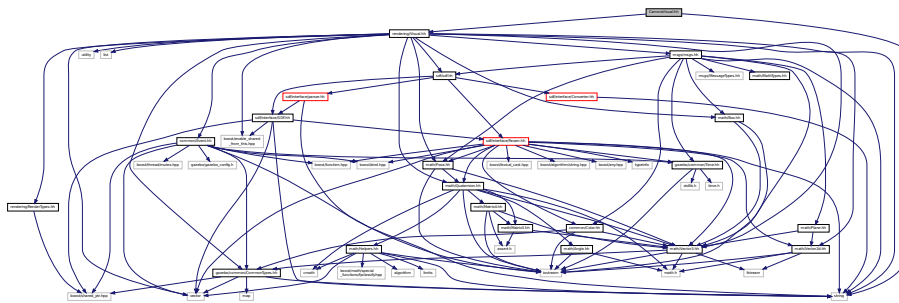
Basic camera sensor.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.14 CameraVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
Include dependency graph for CameraVisual.hh:
```



Classes

- class **gazebo::rendering::CameraVisual**
Basic camera visualization.

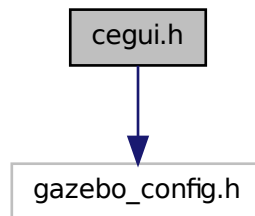
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

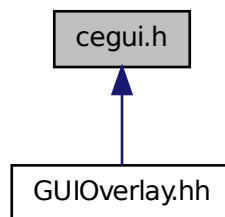
11.15 cegui.h File Reference

```
#include "gazebo_config.h"
```

Include dependency graph for cegui.h:



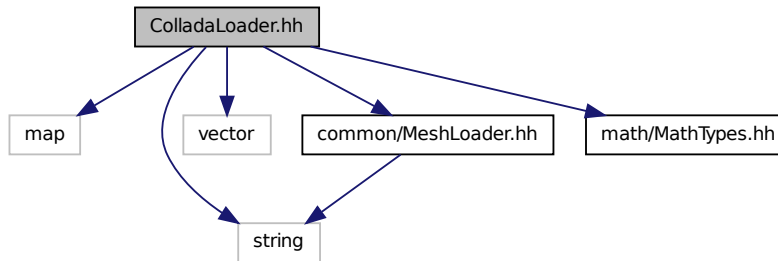
This graph shows which files directly or indirectly include this file:



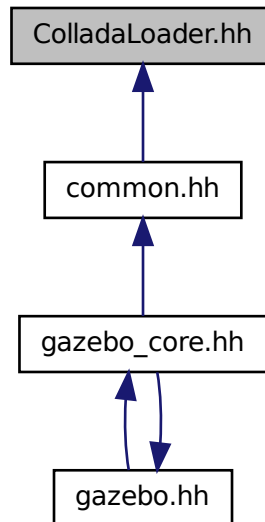
11.16 ColladaLoader.hh File Reference

```
#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.

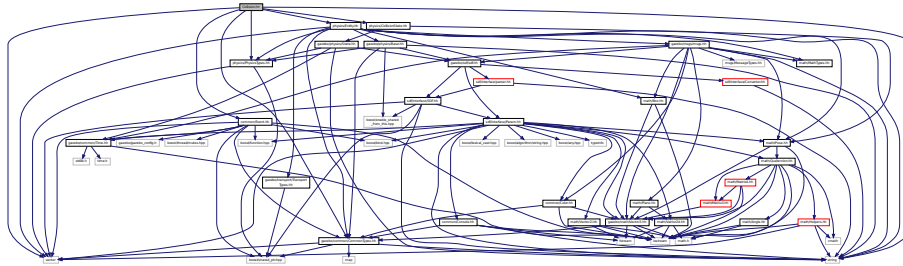
- namespace **gazebo::common**

Common namespace.

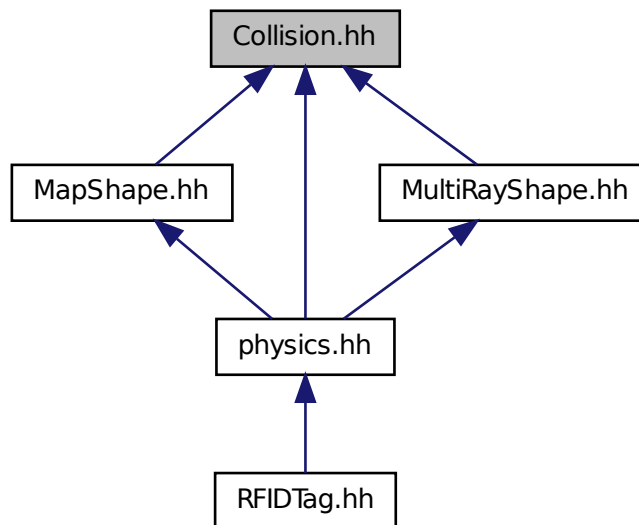
11.17 Collision.hh File Reference

```
#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. 125) class for all collision entities.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

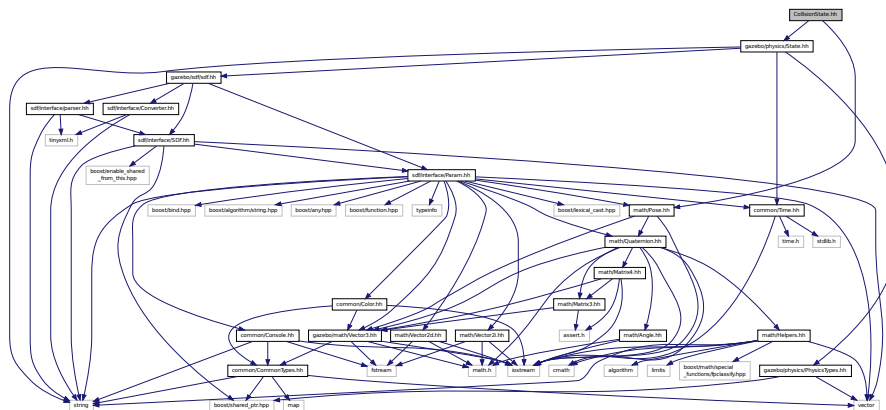
namespace for physics

11.18 CollisionState.hh File Reference

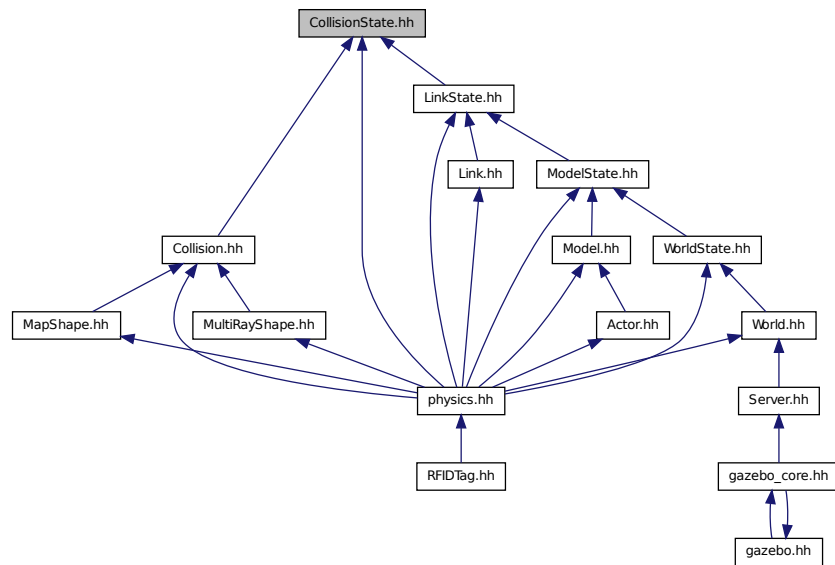
```
#include "gazebo/physics/State.hh"
```

```
#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. 180) object.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

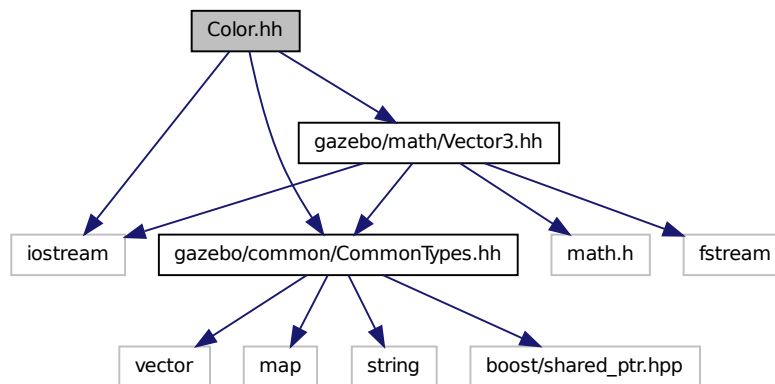
11.19 Color.hh File Reference

```

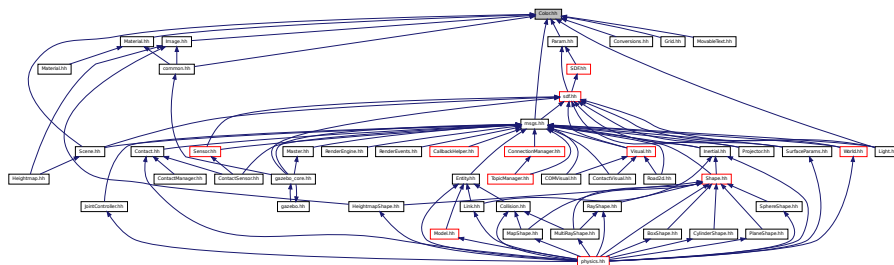
#include <iostream>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/math/Vector3.hh"

```


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.

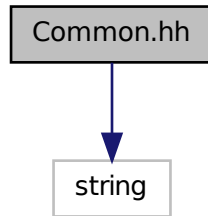
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

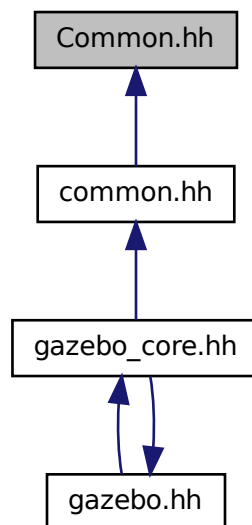
11.20 Common.hh File Reference

```
#include <string>
```

Include dependency graph for Common.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

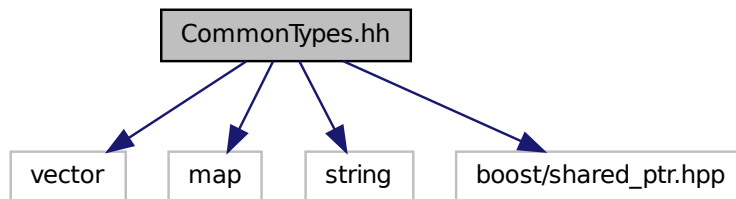
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

Functions

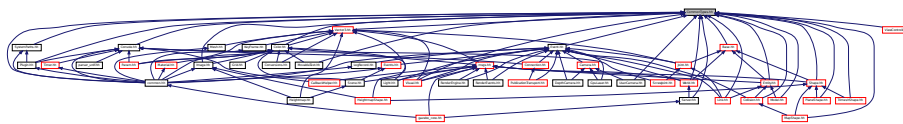
- void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)
*add path prefix to **common::SystemPaths** (p. 726)*
- std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath=true)
*search for file in **common::SystemPaths** (p. 726)*
- std::string **gazebo::common::find_file_path** (const std::string &_file)
*search for a file in **common::SystemPaths** (p. 726)*

11.21 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. 277) 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**

11.21.1 Macro Definition Documentation

11.21.1.1 #define **GAZEBO_DEPRECATED**

11.21.1.2 #define **GAZEBO_FORCEINLINE**

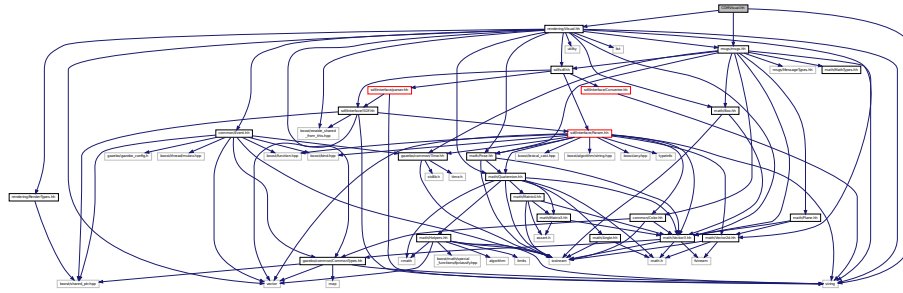
11.21.1.3 #define **NULL** 0

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().

11.22 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.

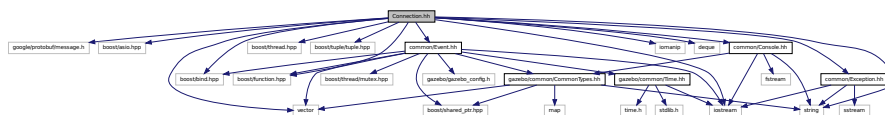
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **ogre**

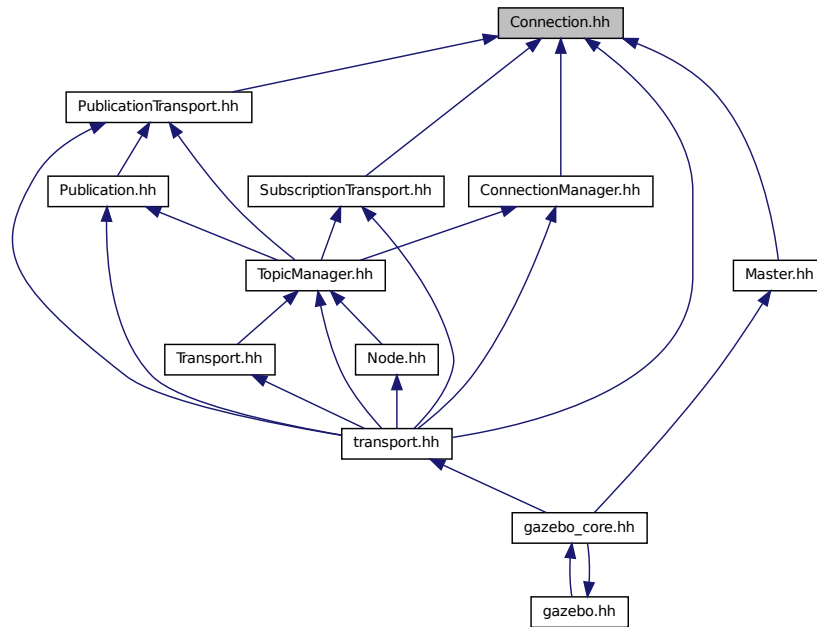
11.23 Connection.hh File Reference

```
#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?

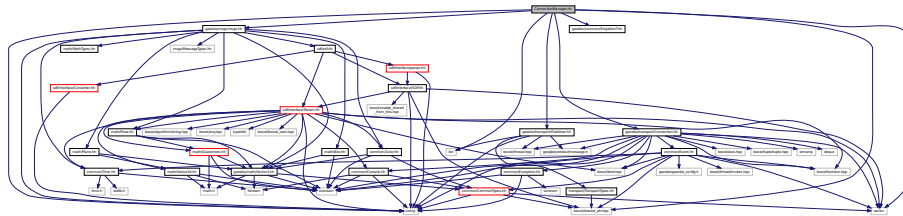
11.23.1 Macro Definition Documentation

11.23.1.1 #define HEADER_LENGTH 8

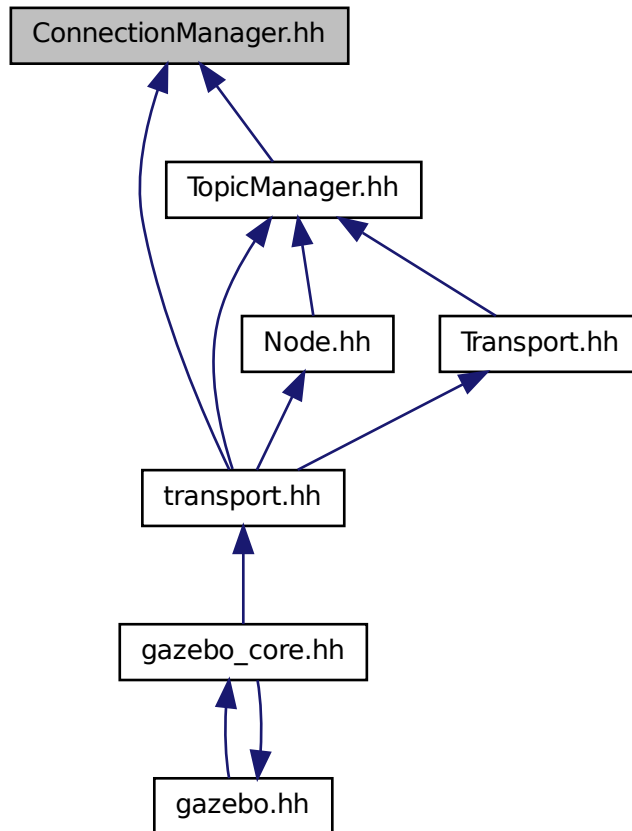
Referenced by gazebo::transport::Connection::AsyncRead().

11.24 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

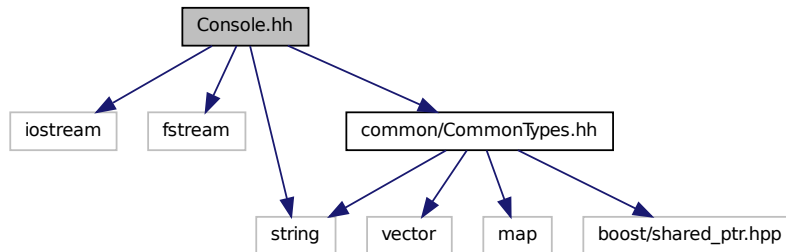
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

11.25 Console.hh File Reference

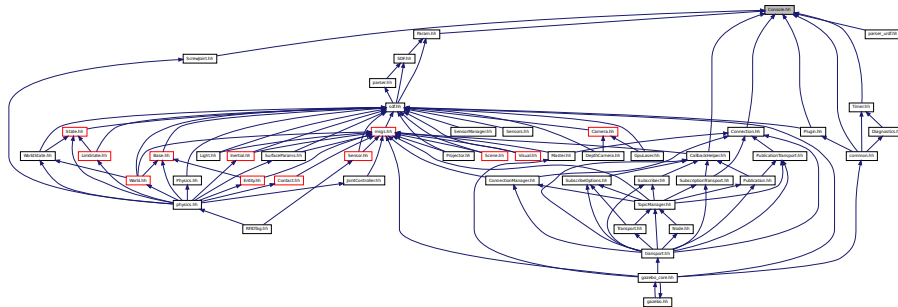
```
#include <iostream>
```



```
#include <fstream>
#include <string>
#include "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 gzmsg (gazebo::common::Console::Instance()->ColorMsg("Msg", 32))`

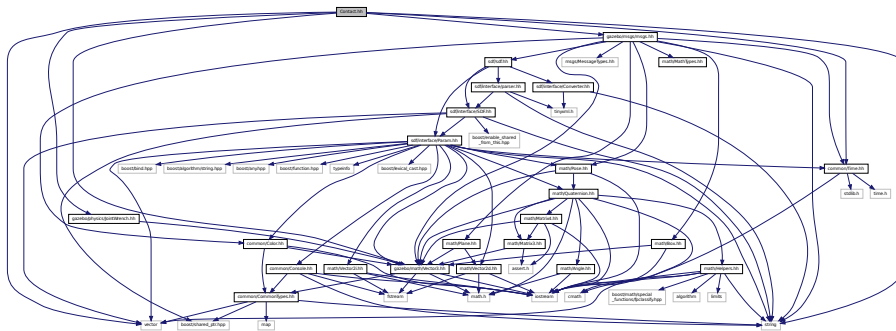
Output a message.

- `#define gzwarn`

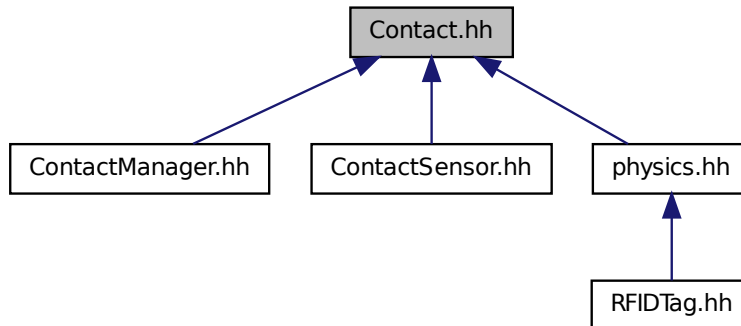
Output a warning message.

11.26 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 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

11.26.1 Macro Definition Documentation

11.26.1.1 #define MAX_COLLIDE_RETURNS 250

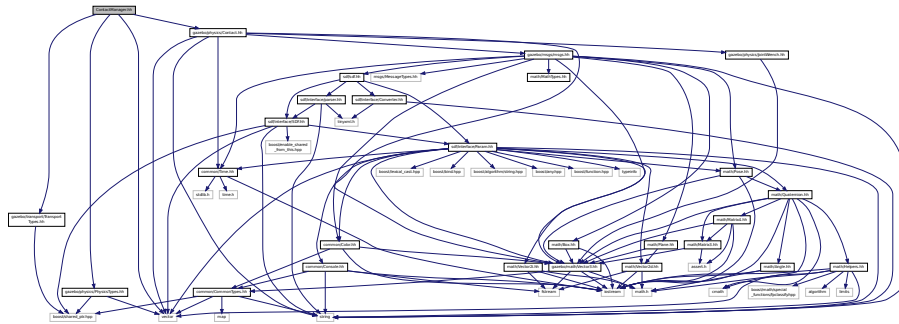
11.26.1.2 #define MAX_CONTACT_JOINTS 32

11.27 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.

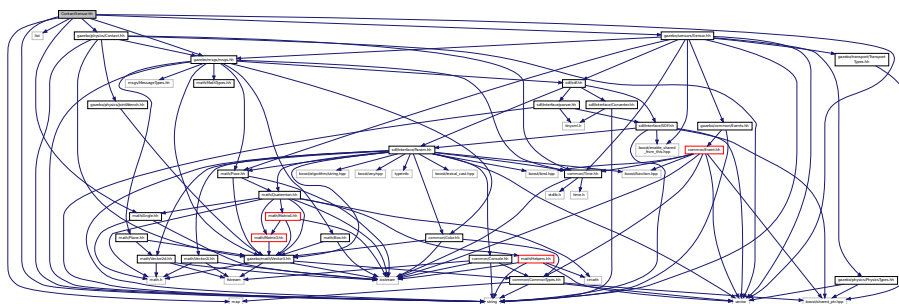
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.28 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

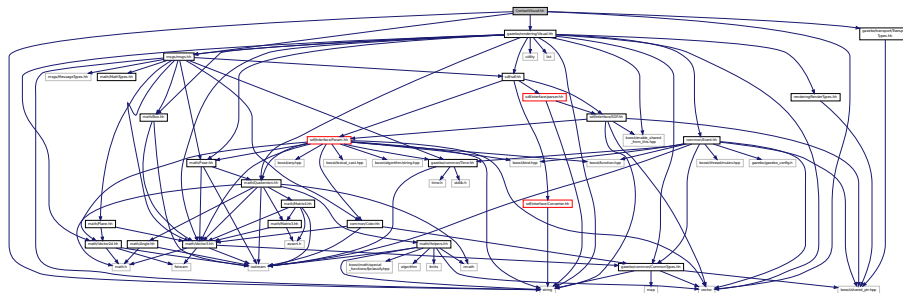
- class **gazebo::sensors::ContactSensor**
Contact sensor.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.29 ContactVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/messages/messages.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**

11.30 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. 230) *Conversions.hh* (p. 906) *rendering/Conversions.hh* (p. 906).

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

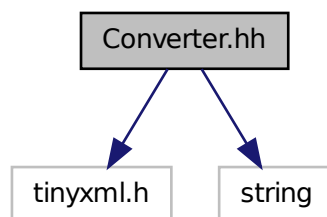
- namespace **gazebo::rendering**

Rendering namespace.

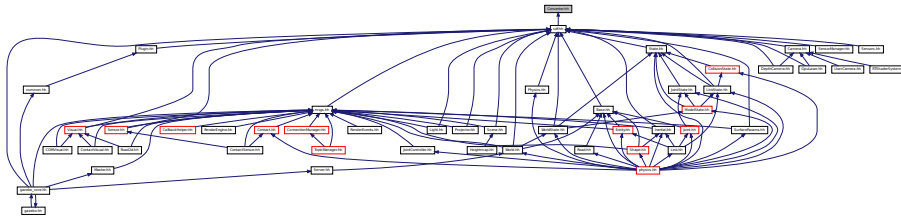
11.31 Converter.hh File Reference

```
#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. 649) to another.*

Namespaces

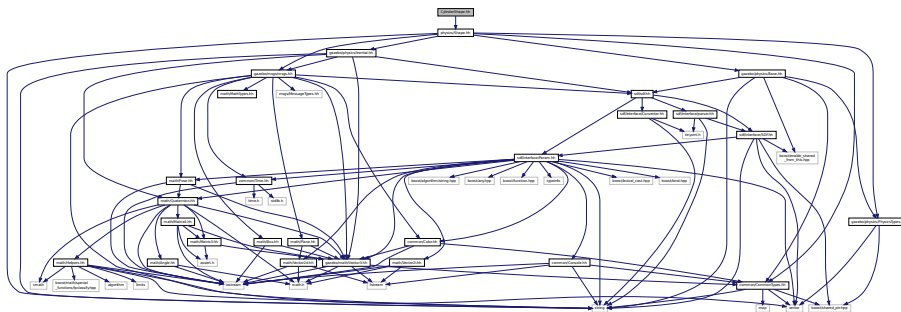
- namespace **sdf**

namespace for Simulation Description Format parser

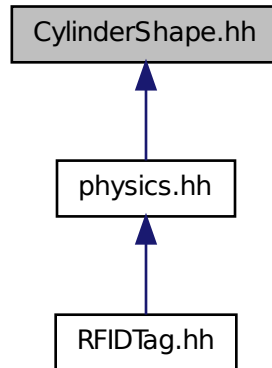
11.32 CylinderShape.hh File Reference

```
#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.

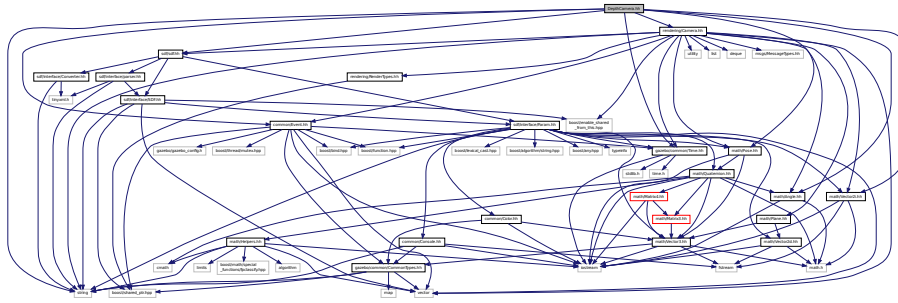
- namespace **gazebo::physics**

namespace for physics

11.33 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.

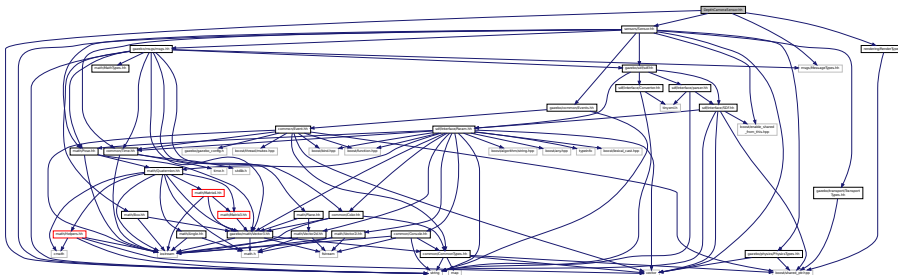
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.34 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**

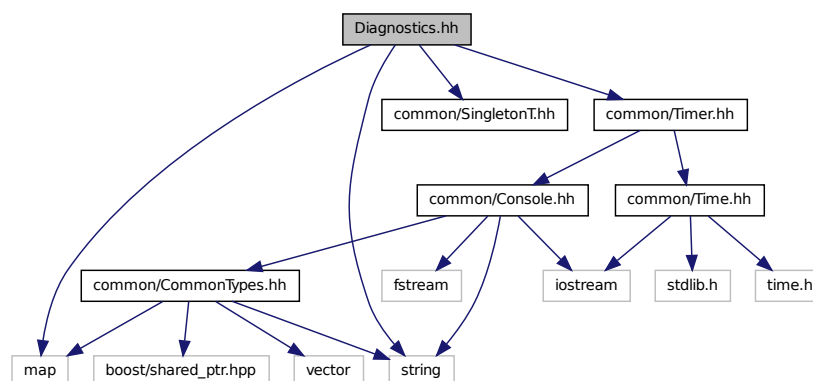
Forward declarations for the common classes.

- namespace **gazebo::sensors**

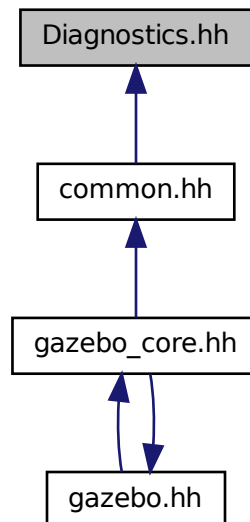
Sensors namespace.

11.35 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 diagnostic manager class.
- class **gazebo::common::DiagnosticTimer**
A 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**

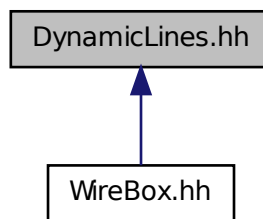
11.36 DynamicLines.hh File Reference

```
#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.

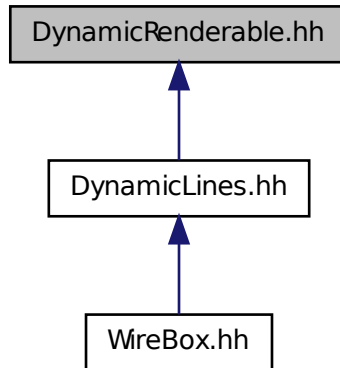
11.37 DynamicRenderable.hh File Reference

```
#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.

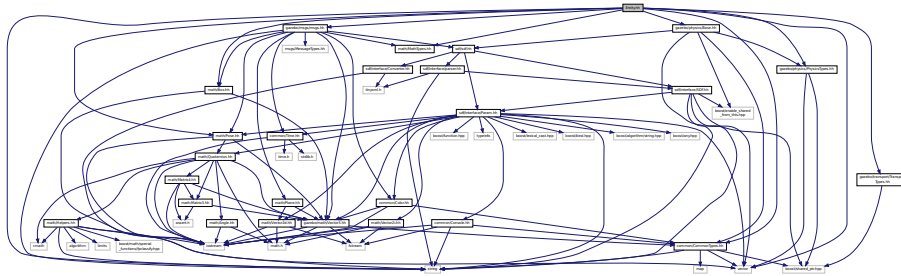
- namespace **gazebo::rendering**

Rendering namespace.

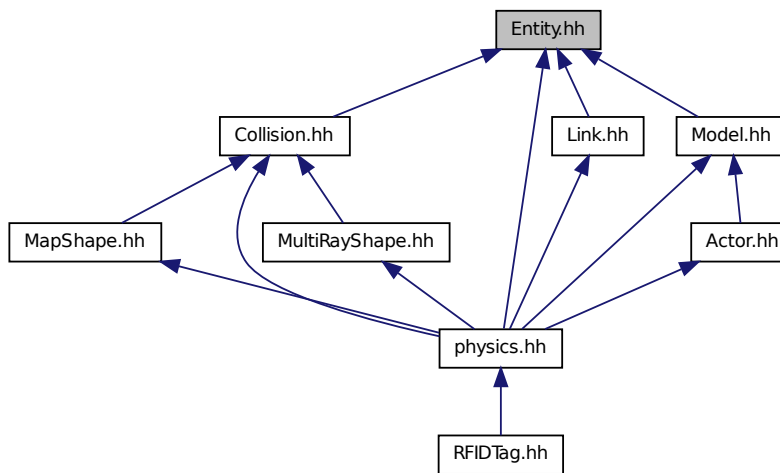
11.38 Entity.hh File Reference

```
#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. 125) class for all physics objects in Gazebo.

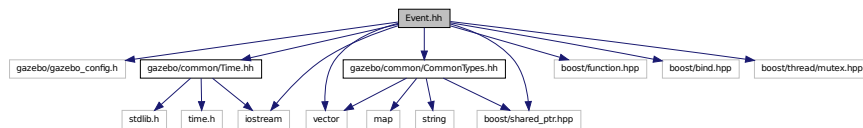
Namespaces

- namespace **boost**
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

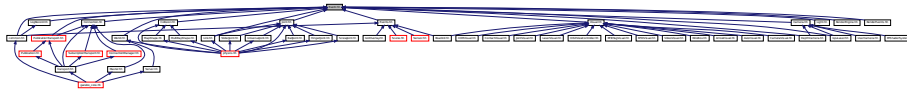
11.39 Event.hh File Reference

```
#include <gazebo/gazebo_config.h>
#include <gazebo/common/Time.hh>
#include <gazebo/common/CommonTypes.hh>
#include <boost/function.hpp>
#include <boost/bind.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/thread/mutex.hpp>
#include <iostream>
#include <vector>
```

Include dependency graph for Event.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::event::Connection**
A class that encapsulates a connection.
- class **gazebo::event::Event**
Base class for all events.
- class **gazebo::event::EventT < T >**
A class for event processing.

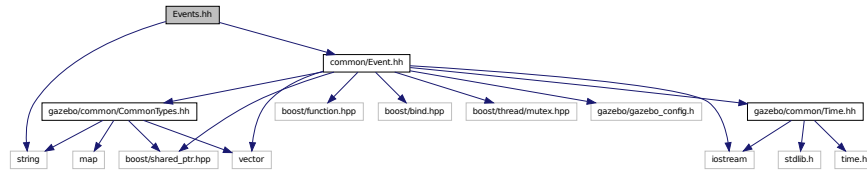
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::event**
Event (p. 277) namespace.

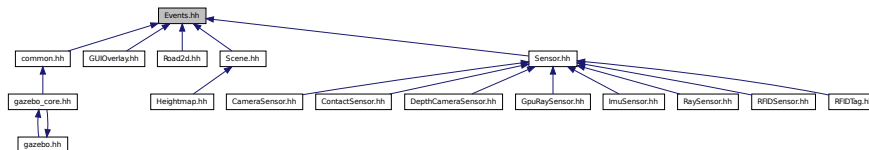
11.40 Events.hh File Reference

```
#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. 277) class to get notifications for simulator events.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::event**

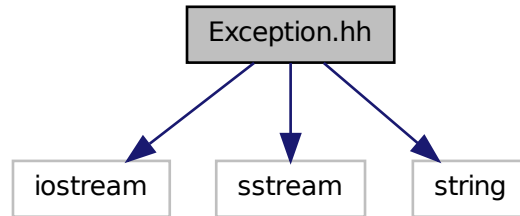
Event (p. 277) namespace.

11.41 Exception.hh File Reference

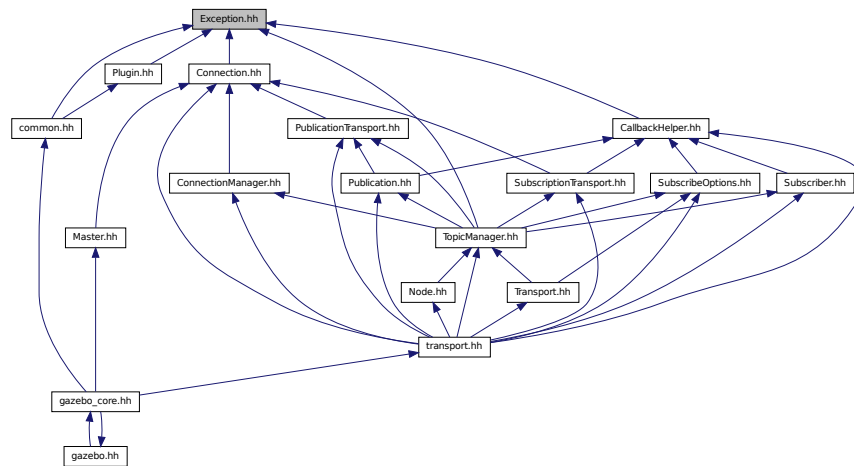
```

#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::Exception**

Class for generating exceptions.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

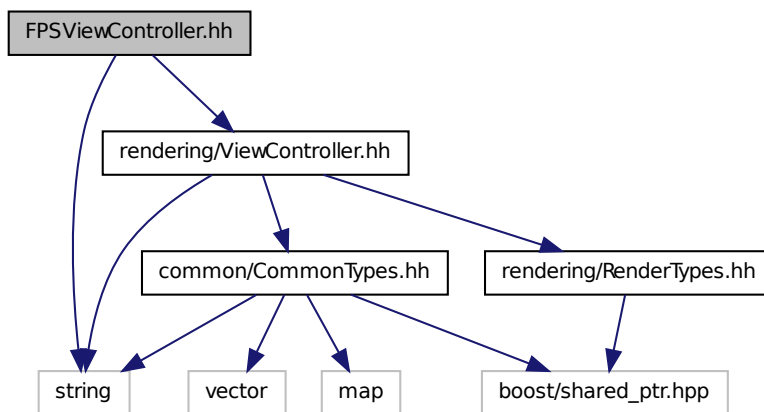
Macros

- `#define gzthrow(msg)`

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

11.42 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.

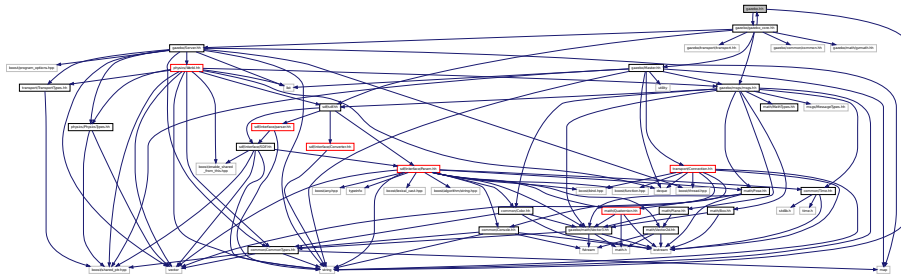
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

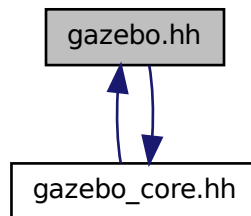
11.43 gazebo.hh File Reference

```
#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.

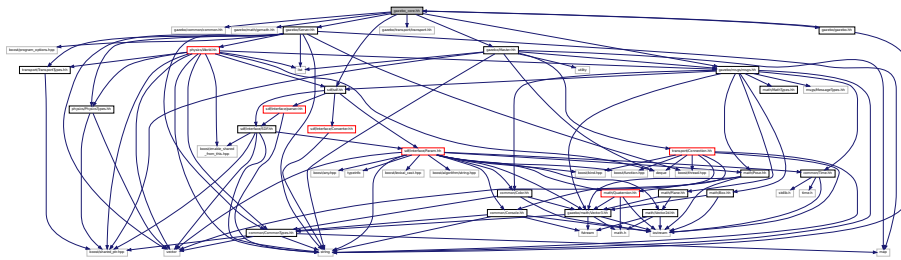
Functions

- 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** ()

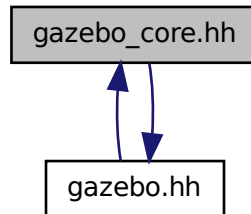
11.44 gazebo_core.hh File Reference

```
#include <gazebo/common/common.hh>
#include <gazebo/math/gzmath.hh>
#include <gazebo/messages/messages.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:



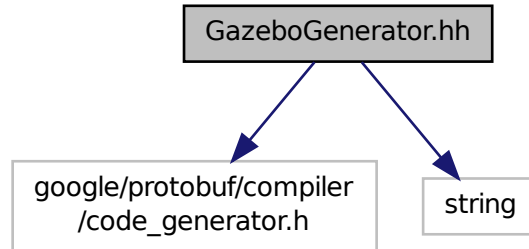
This graph shows which files directly or indirectly include this file:



11.45 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. 83).

Namespaces

- namespace **google**
- namespace **google::protobuf**
- namespace **google::protobuf::compiler**
- namespace **google::protobuf::compiler::cpp**

11.46 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.

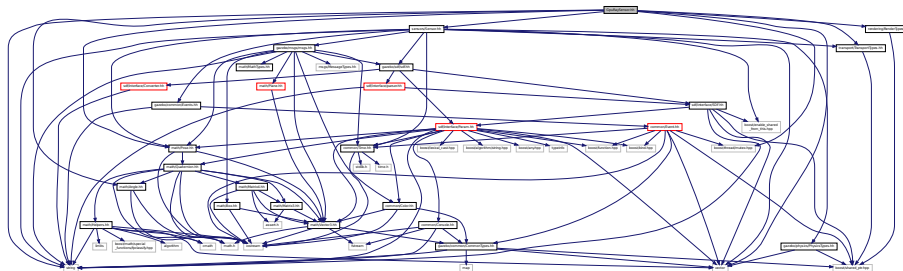
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.47 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.

11.48 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**

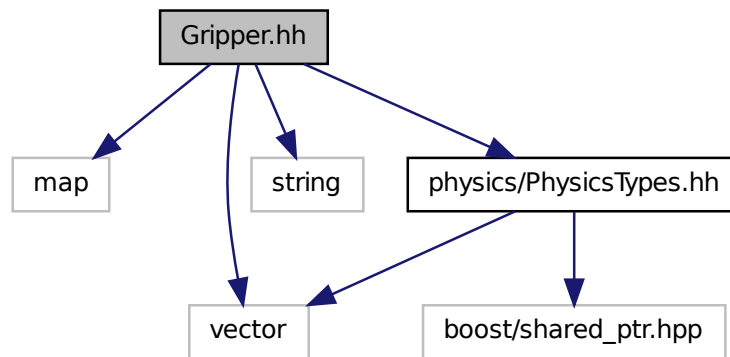
Rendering namespace.

- namespace **Ogre**

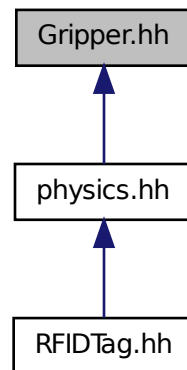
11.49 Gripper.hh File Reference

```
#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 gripper abstraction.

Namespaces

- namespace **gazebo**

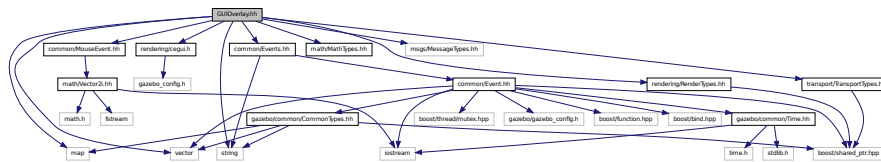
Forward declarations for the common classes.

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

11.50 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 class that creates a CEGUI overlay on a render window.

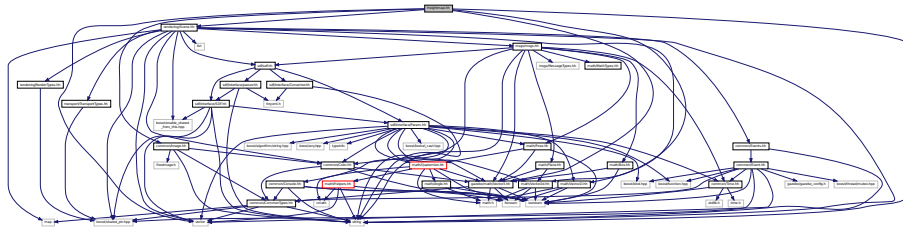
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.51 Heightmap.hh File Reference

```
#include <string>
#include <vector>
#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::Heightmap**
Rendering a terrain using heightmap information.

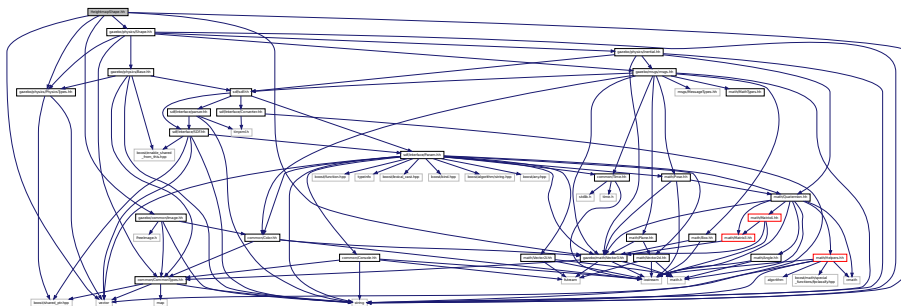
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

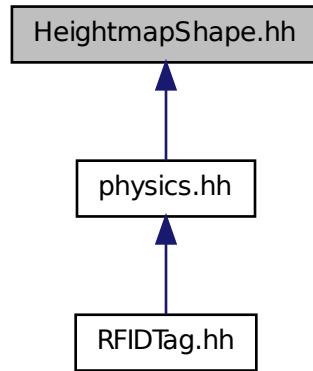
11.52 HeightmapShape.hh File Reference

```
#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. 341) collision shape builds a heightmap from an image.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

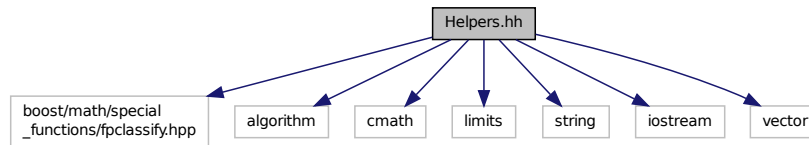
- namespace **gazebo::physics**

namespace for physics

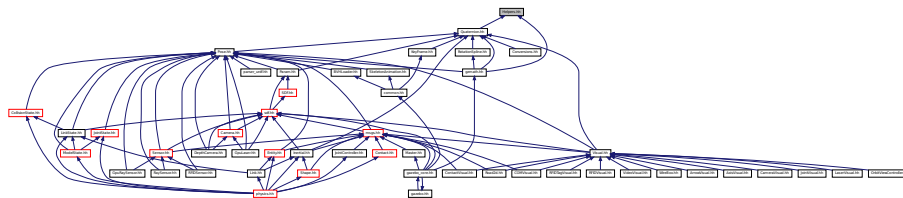
11.53 Helpers.hh File Reference

```
#include <boost/math/special_functions/fpclassify.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)

11.53.1 Macro Definition Documentation

11.53.1.1 #define GZ_DBL_MAX std::numeric_limits<double>::max()

11.53.1.2 #define GZ_DBL_MIN std::numeric_limits<double>::min()

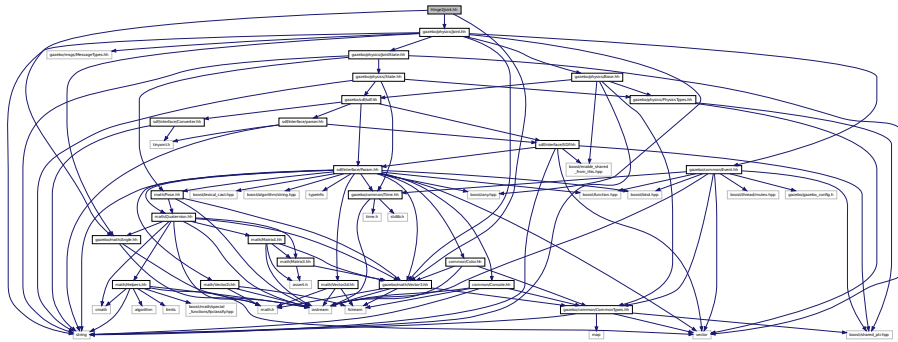
11.53.1.3 #define GZ_FLT_MAX std::numeric_limits<float>::max()

11.53.1.4 #define GZ_FLT_MIN std::numeric_limits<float>::min()

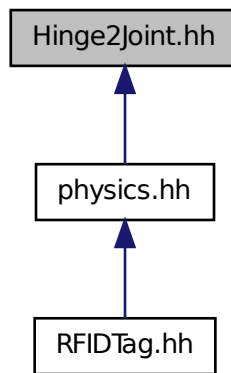
11.54 Hinge2Joint.hh File Reference

```
#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

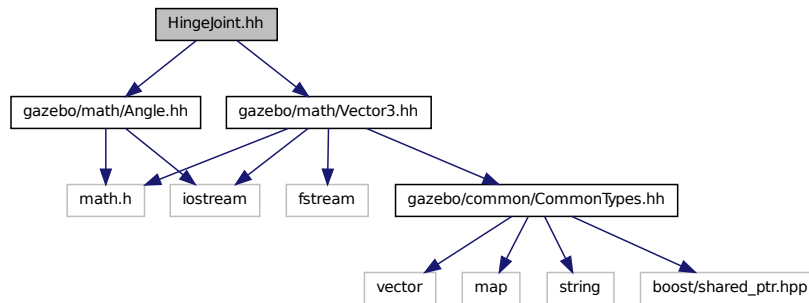
- class **gazebo::physics::Hinge2Joint**< T >
A two axis hinge joint.

Namespaces

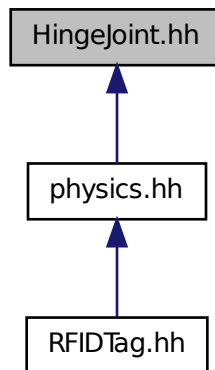
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.55 HingeJoint.hh File Reference

```
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
Include dependency graph for HingeJoint.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::HingeJoint**< T >
A single axis hinge joint.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

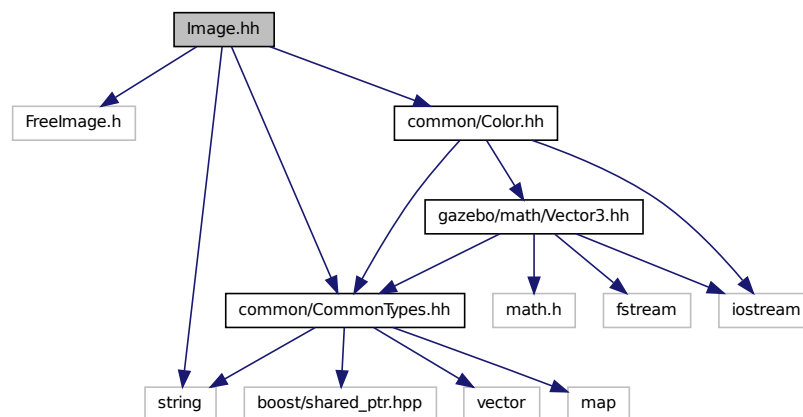
- namespace **gazebo::physics**

namespace for physics

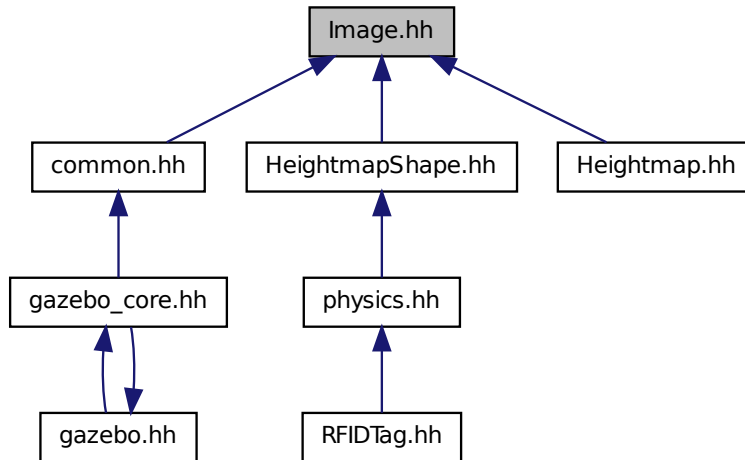
11.56 Image.hh File Reference

```
#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.

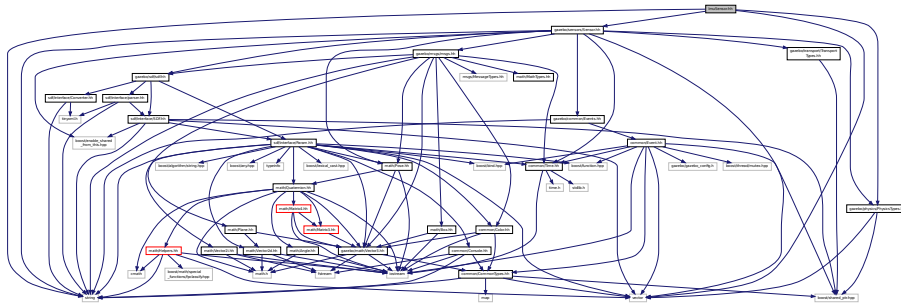
11.57 ImuSensor.hh File Reference

```

#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.

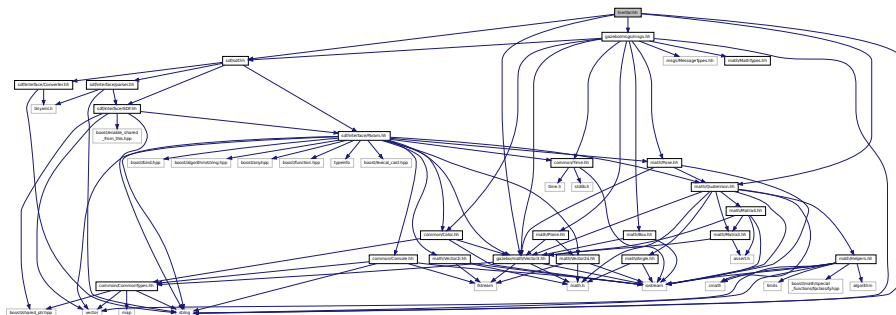
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

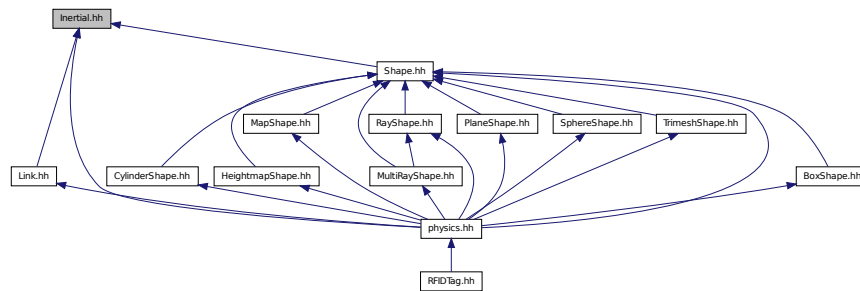
11.58 Inertial.hh File Reference

```
#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 class for inertial information about a link.

Namespaces

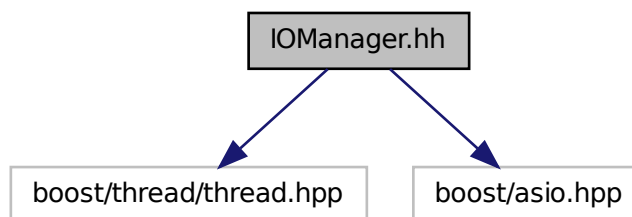
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.59 IOManager.hh File Reference

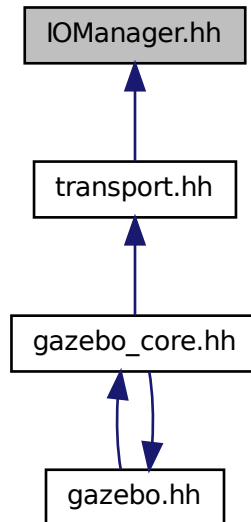
```
#include <boost/thread/thread.hpp>
```

```
#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**

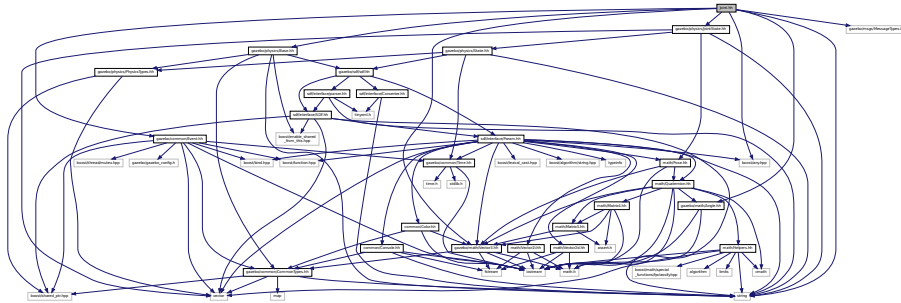
11.60 Joint.hh File Reference

```

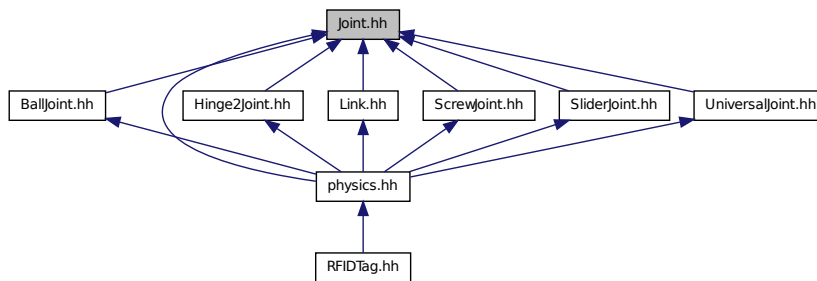
#include <string>
#include <boost/any.hpp>
#include "gazebo/common/Event.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/physics/JointState.hh"
#include "gazebo/physics/Base.hh"

```

Include dependency graph for Joint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Joint**
Base (p. 125) class for all joints.

Namespaces

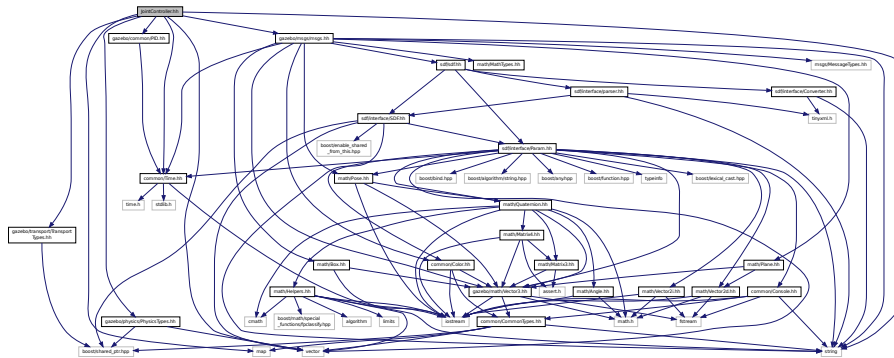
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.61 JointController.hh File Reference

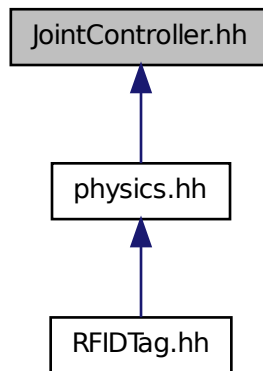
```
#include <map>
```

```
#include <string>
#include <vector>
#include "gazebo/common/PID.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo msgs/msgs.hh"
```

Include dependency graph for JointController.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::JointController**

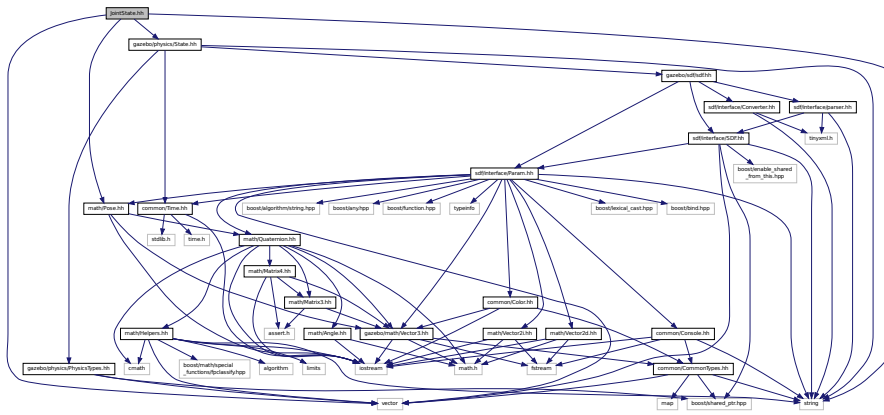
*A class for manipulating **physics::Joint** (p. 366).*

Namespaces

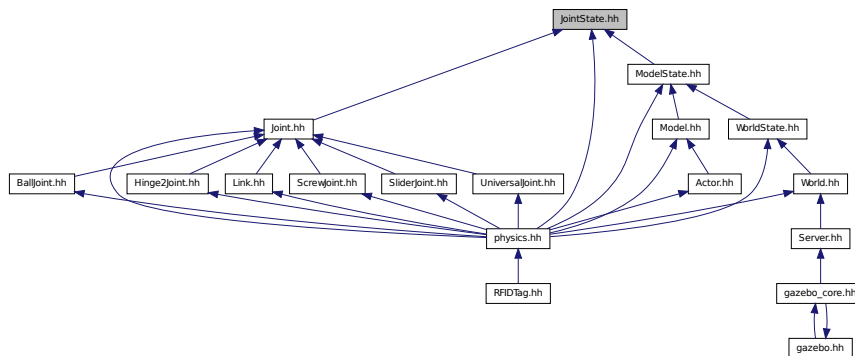
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.62 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**

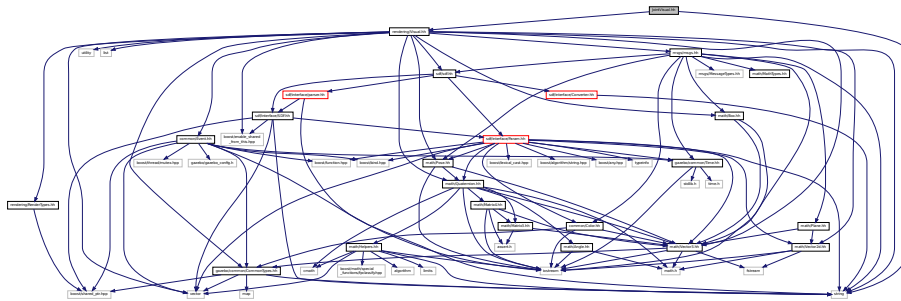
keeps track of state of a **physics::Joint** (p. 366)

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.63 JointVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
Include dependency graph for JointVisual.hh:
```



Classes

- class **gazebo::rendering::JointVisual**
Visualization for joints.

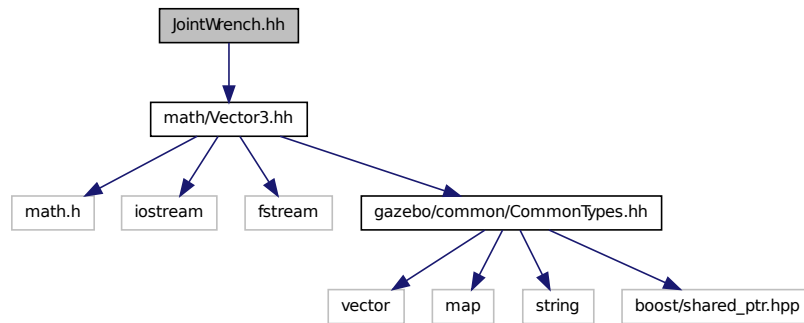
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

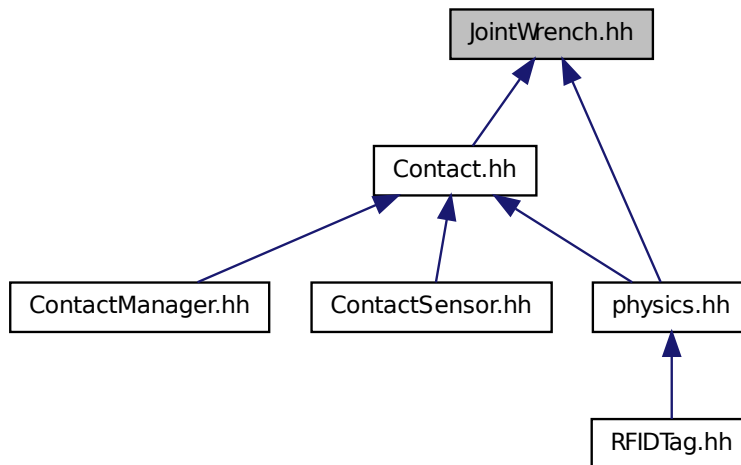
11.64 JointWrench.hh File Reference

```
#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.

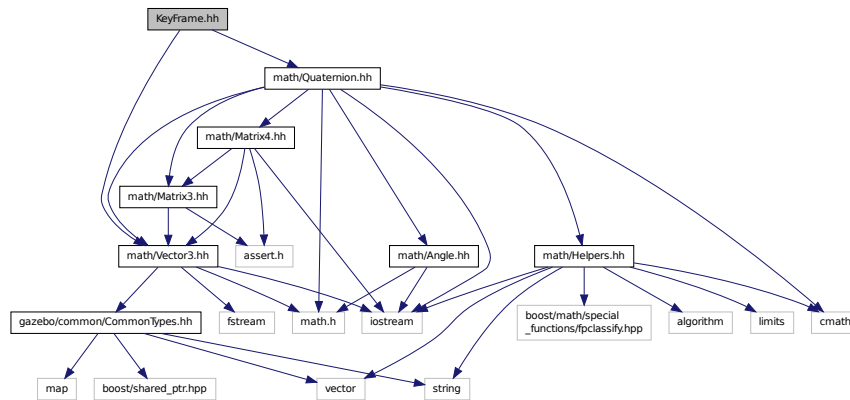
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.

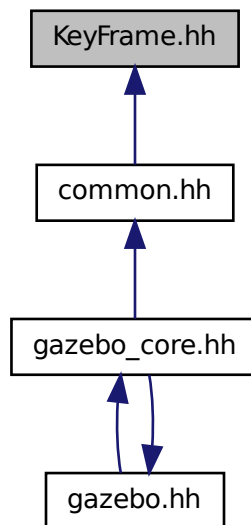
- namespace **gazebo::physics**
namespace for physics

11.65 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 key frame in an animation.
- class **gazebo::common::NumericKeyFrame**
A keyframe for a **NumericAnimation** (p. 514).
- class **gazebo::common::PoseKeyFrame**
A keyframe for a **PoseAnimation** (p. 564).

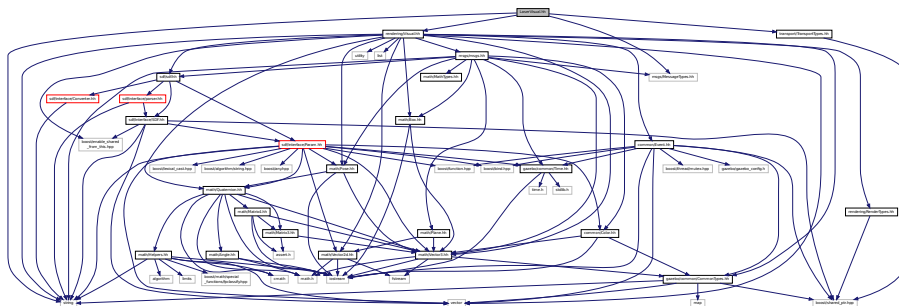
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.66 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.

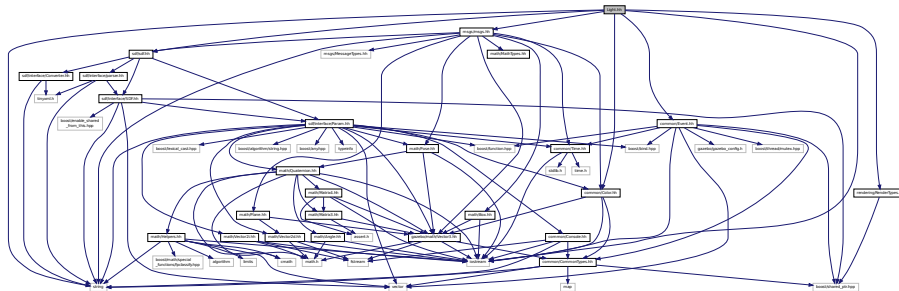
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.67 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 light source.

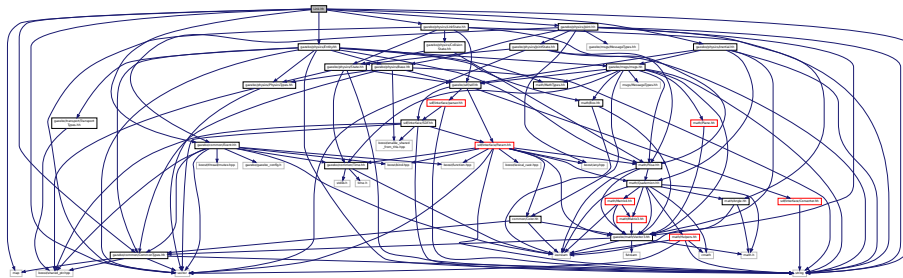
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

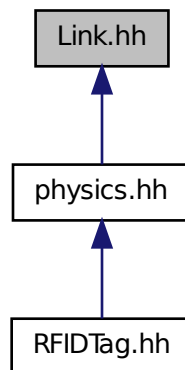
11.68 Link.hh File Reference

```
#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. 398) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

Namespaces

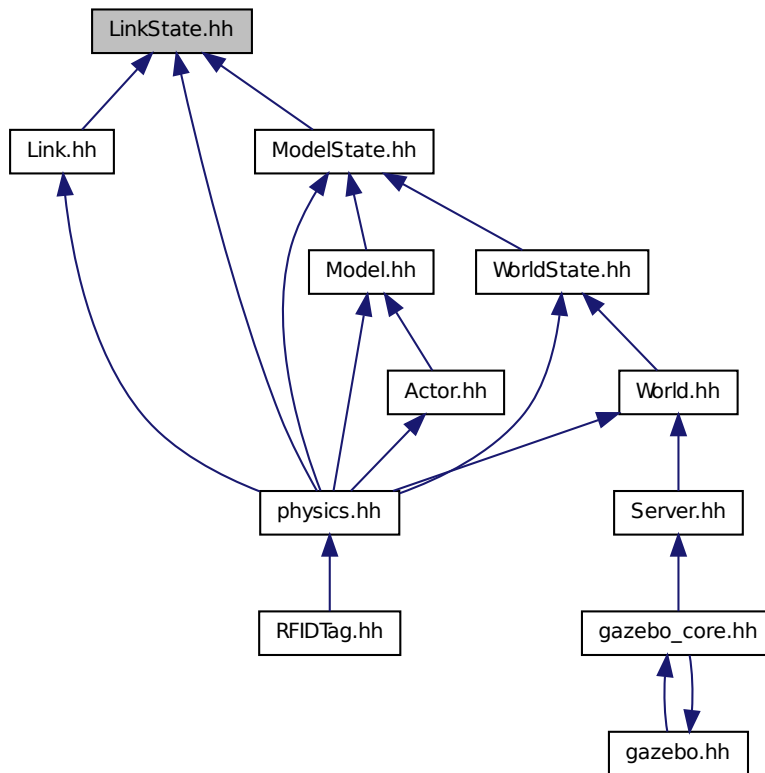
- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::LinkState**
Store state information of a **physics::Link** (p. 398) object.

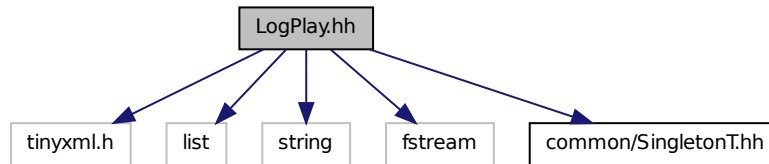
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

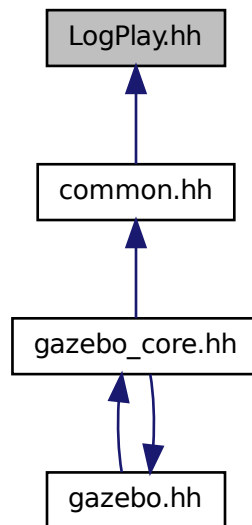
11.70 LogPlay.hh File Reference

```
#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.

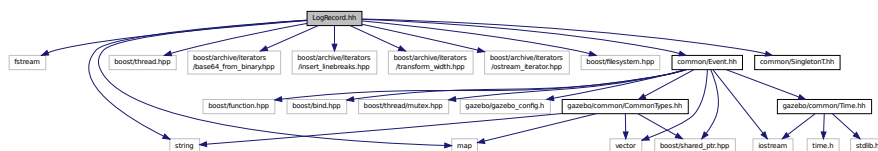
- namespace **gazebo::common**

Common namespace.

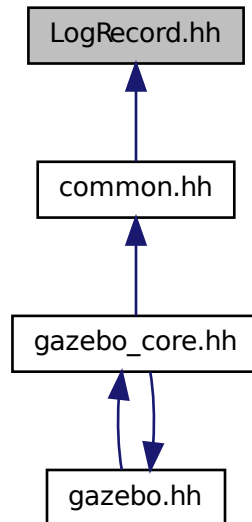
11.71 LogRecord.hh File Reference

```
#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"

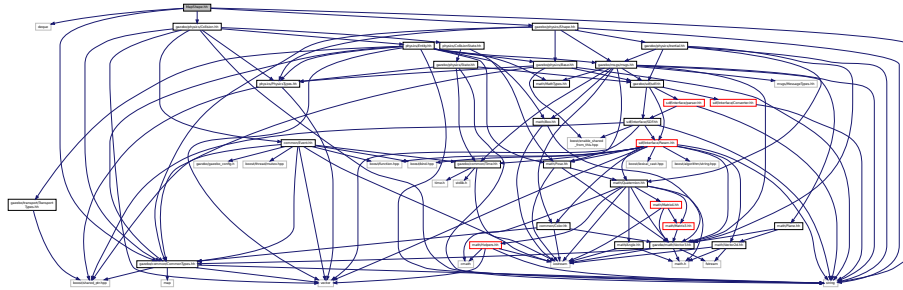
11.71.1 Macro Definition Documentation

- 11.71.1.1 #define GZ_LOG_VERSION "1.0"

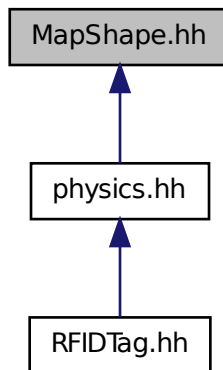
11.72 mainpage.html File Reference

11.73 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:



11.74 Master.hh File Reference

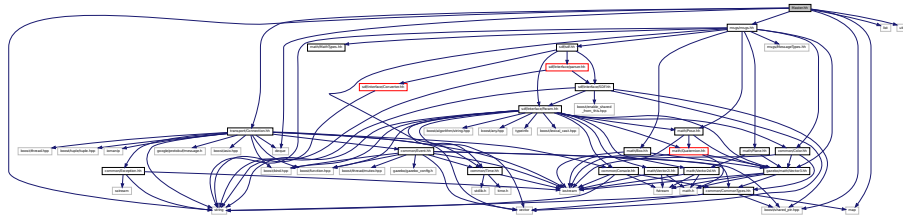
```
#include <string>
```

```

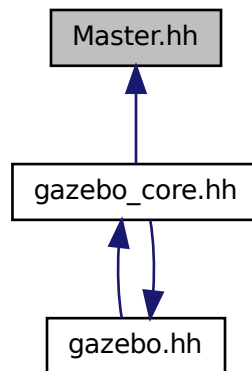
#include <list>
#include <deque>
#include <utility>
#include <map>
#include <boost/shared_ptr.hpp>
#include "msgs/msgs.hh"
#include "transport/Connection.hh"

```

Include dependency graph for Master.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::Master**

A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

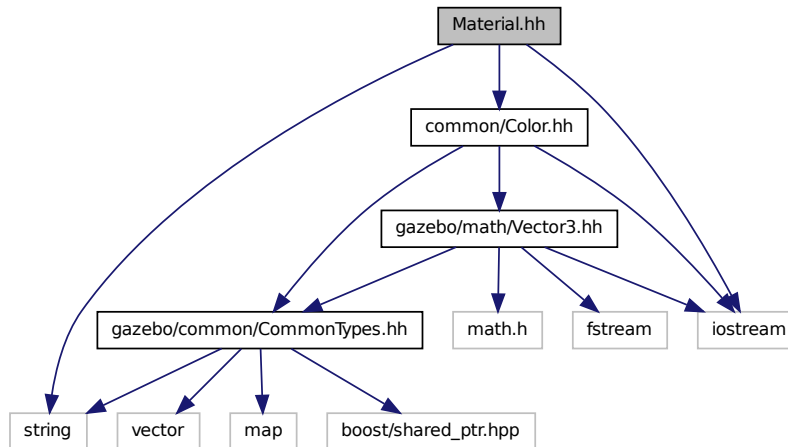
Namespaces

- namespace **gazebo**

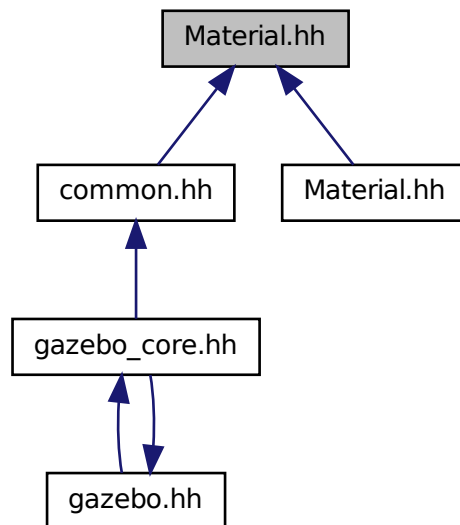
Forward declarations for the common classes.

11.75 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.

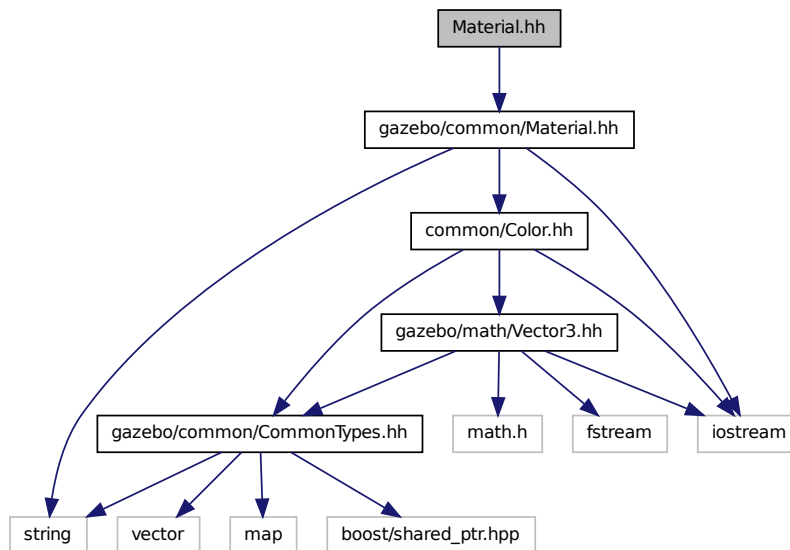
- namespace **gazebo::common**

Common namespace.

11.76 Material.hh File Reference

```
#include "gazebo/common/Material.hh"
```

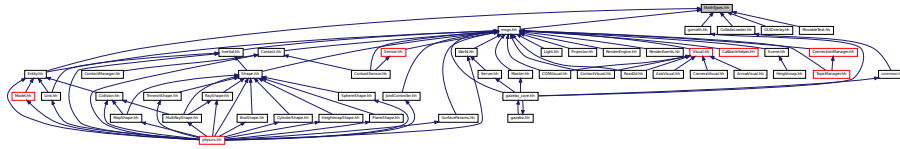
Include dependency graph for rendering/Material.hh:



11.77 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**

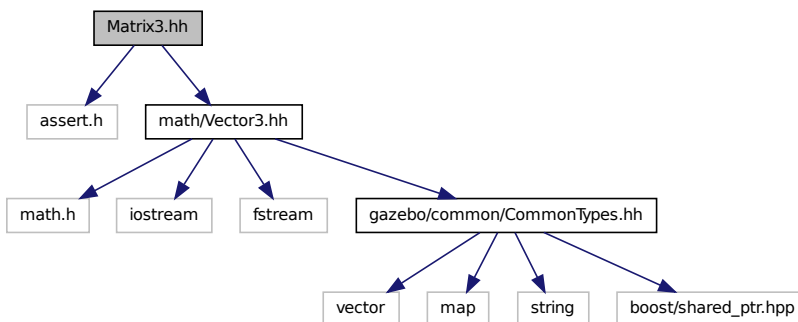
Math namespace.

11.77.1 Detailed Description

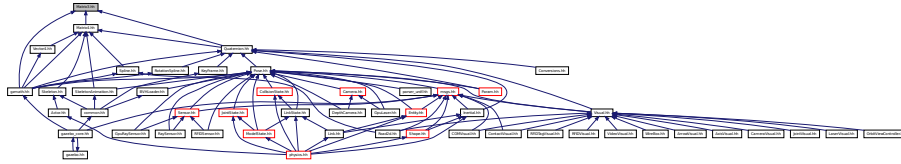
Forward declarations for the math classes.

11.78 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 3x3 matrix class.

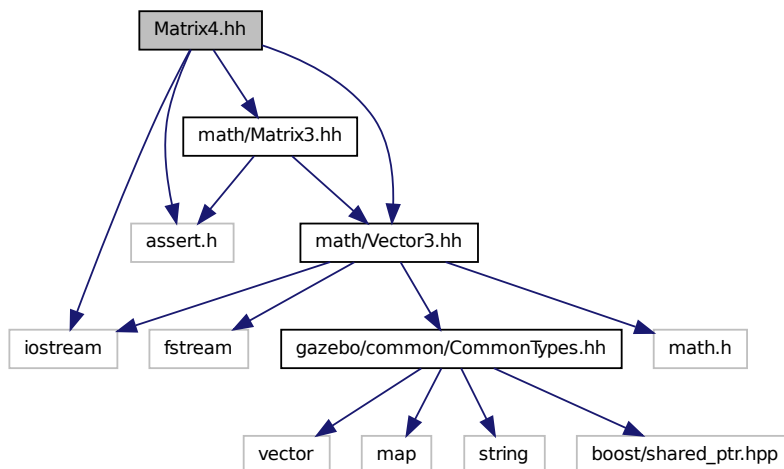
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

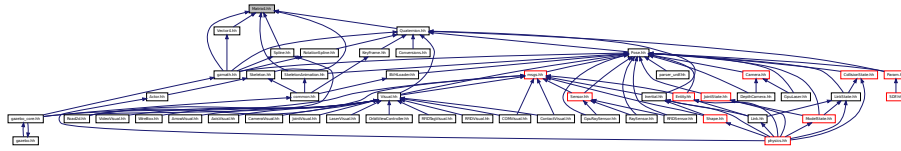
11.79 Matrix4.hh File Reference

```
#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 3x3 matrix class.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

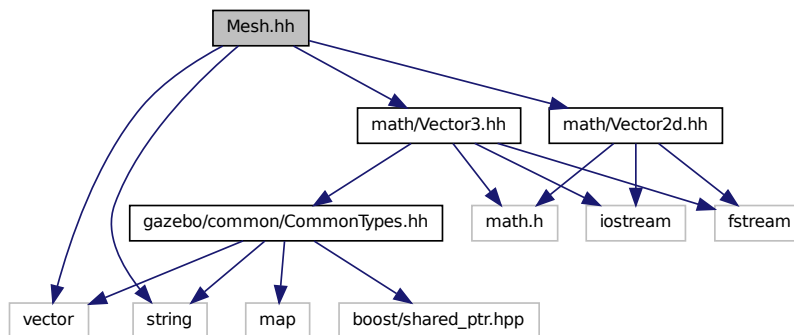
- namespace **gazebo::math**

Math namespace.

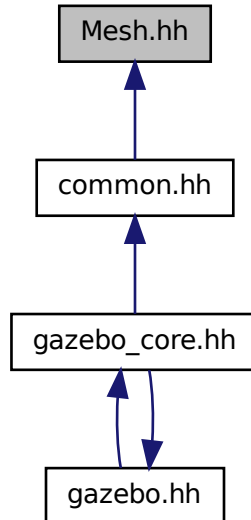
11.80 Mesh.hh File Reference

```
#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 3D mesh.
- struct **gazebo::common::NodeAssignment**
Vertex to node weighted assignment for skeleton animation visualization.
- class **gazebo::common::SubMesh**
A child mesh.

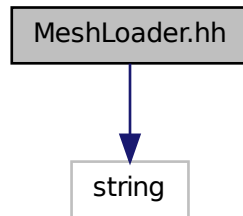
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

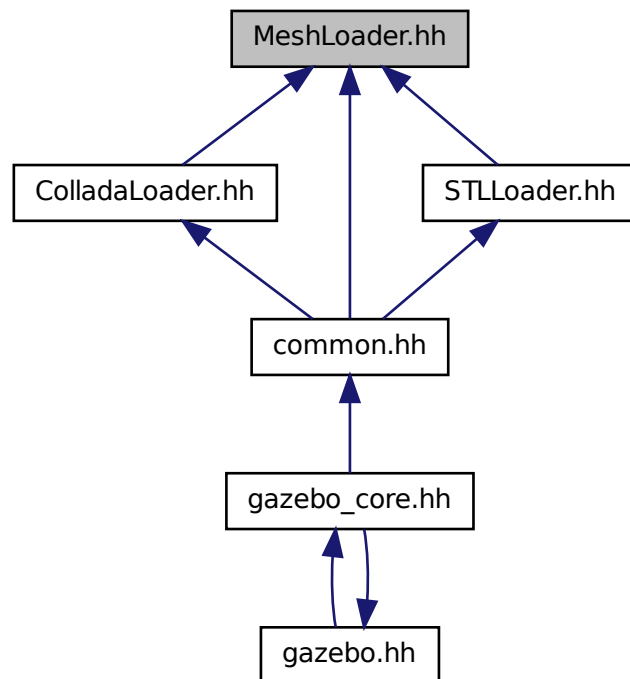
11.81 MeshLoader.hh File Reference

```
#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**

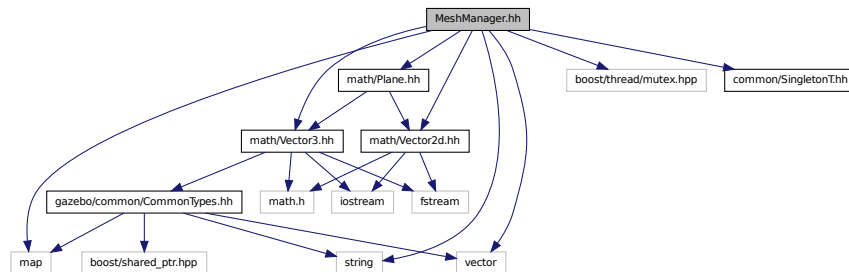
Forward declarations for the common classes.

- namespace **gazebo::common**

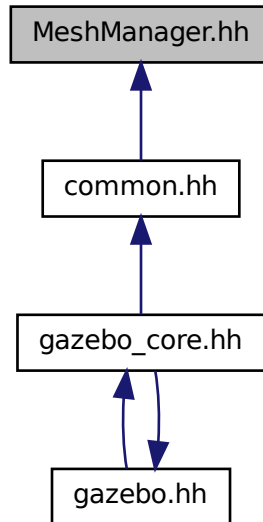
Common namespace.

11.82 MeshManager.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include <boost/thread/mutex.hpp>
#include "math/Vector3.hh"
#include "math/Vector2d.hh"
#include "math/Plane.hh"
#include "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.

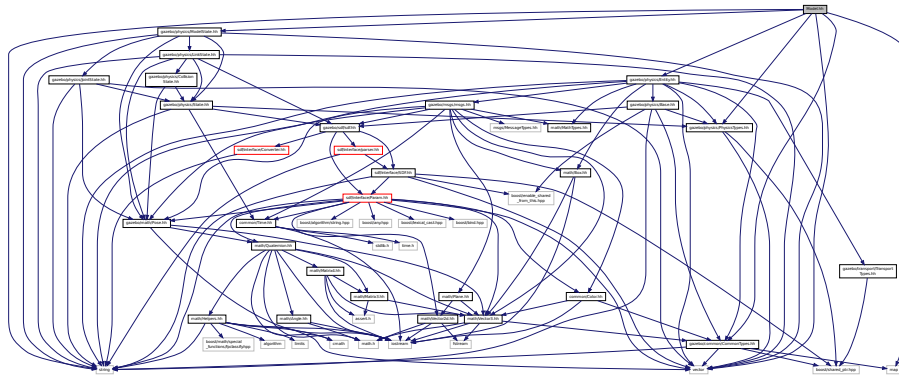
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

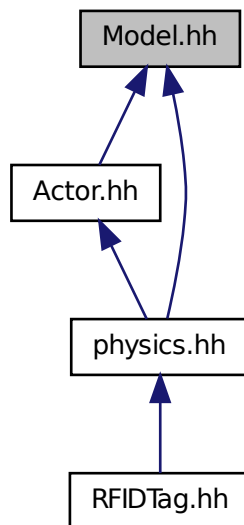
11.83 Model.hh File Reference

```
#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 model is a collection of links, joints, and plugins.

Namespaces

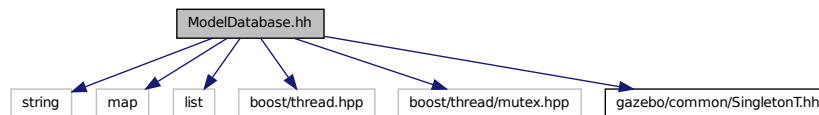
- namespace **boost**

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.84 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.

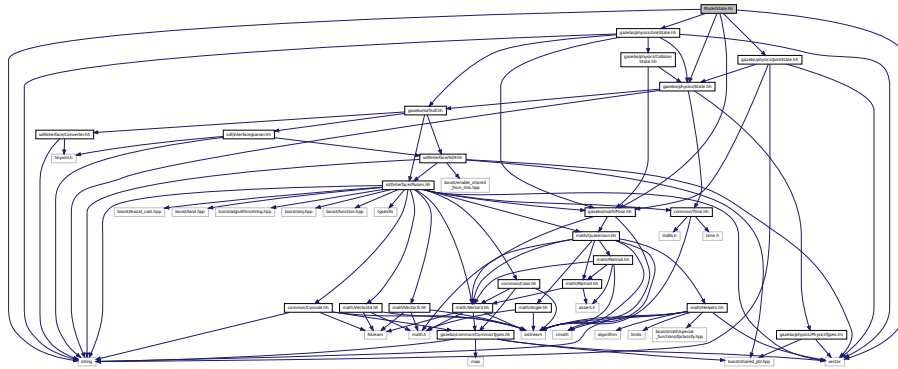
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

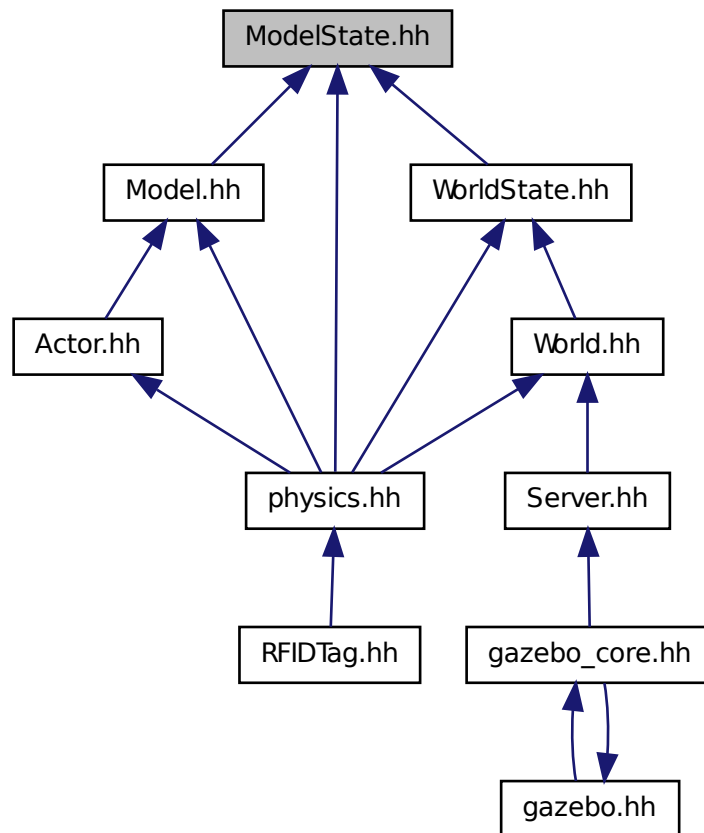
11.85 ModelState.hh File Reference

```
#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. 460) object.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

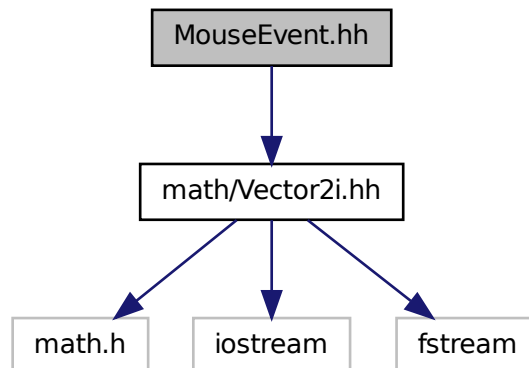
- namespace **gazebo::physics**

namespace for physics

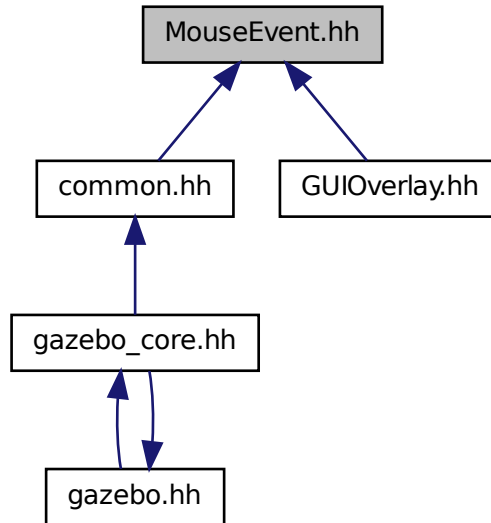
11.86 MouseEvent.hh File Reference

```
#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.

11.87 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.

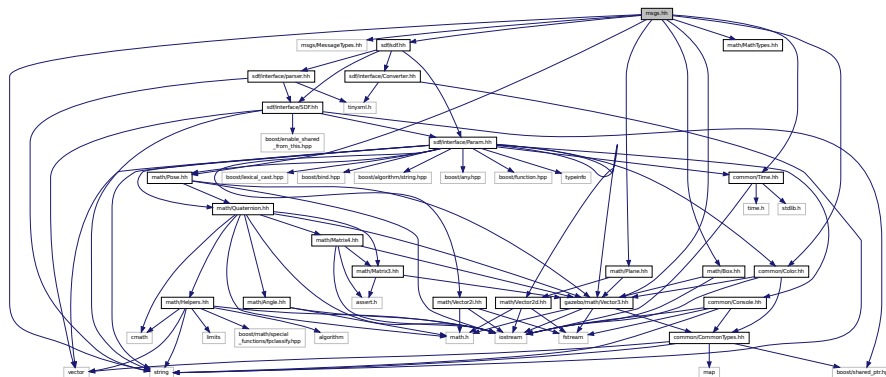
Namespaces

- namespace **boost**
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

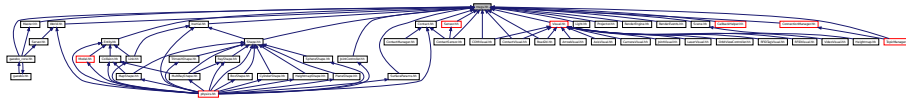
11.88 msgs.hh File Reference

```
#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. 799) to a `msgs::Vector3d`.*
- `msgs::Quaternion gazebo::msgs::Convert` (`const math::Quaternion &_q`)
*Convert a **math::Quaternion** (p. 581) to a `msgs::Quaternion`.*
- `msgs::Pose gazebo::msgs::Convert` (`const math::Pose &_p`)
*Convert a **math::Pose** (p. 556) to a `msgs::Pose`.*
- `msgs::Color gazebo::msgs::Convert` (`const common::Color &_c`)
*Convert a **common::Color** (p. 193) to a `msgs::Color`.*
- `msgs::Time gazebo::msgs::Convert` (`const common::Time &_t`)
*Convert a **common::Time** (p. 732) to a `msgs::Time`.*
- `msgs::PlaneGeom gazebo::msgs::Convert` (`const math::Plane &_p`)
*Convert a **math::Plane** (p. 547) 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. 581).*
- `math::Pose gazebo::msgs::Convert` (`const msgs::Pose &_p`)
*Convert a `msgs::Pose` to a **math::Pose** (p. 556).*
- `common::Color gazebo::msgs::Convert` (`const msgs::Color &_c`)
*Convert a `msgs::Color` to a **common::Color** (p. 193).*
- `common::Time gazebo::msgs::Convert` (`const msgs::Time &_t`)
*Convert a `msgs::Time` to a **common::Time** (p. 732).*
- `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`)

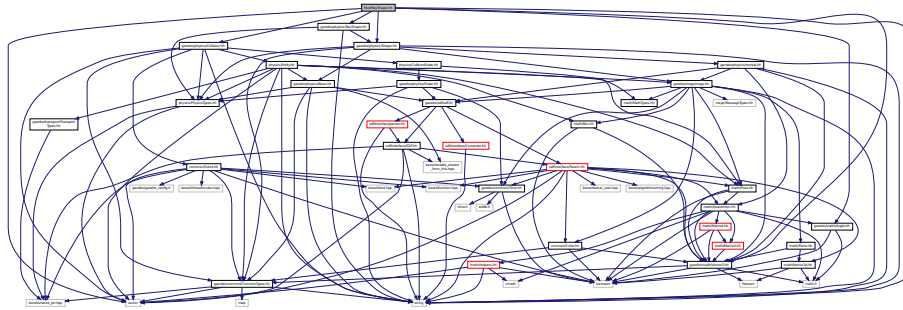
- Create a `msgs::GUI` from a GUI SDF element.*

 - 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. 349).
- void **gazebo::msgs::Set** (msgs::Image *_msg, const common::Image &_i)
Set a `msgs::Image` from a `common::Image` (p. 349).
- void **gazebo::msgs::Set** (msgs::Vector3d *_pt, const math::Vector3 &_v)
Set a `msgs::Vector3d` from a `math::Vector3` (p. 799).
- void **gazebo::msgs::Set** (msgs::Vector2d *_pt, const math::Vector2d &_v)
Set a `msgs::Vector2d` from a `math::Vector3` (p. 799).
- void **gazebo::msgs::Set** (msgs::Quaternion *_q, const math::Quaternion &_v)
Set a `msgs::Quaternion` from a `math::Quaternion` (p. 581).
- void **gazebo::msgs::Set** (msgs::Pose *_p, const math::Pose &_v)
Set a `msgs::Pose` from a `math::Pose` (p. 556).
- void **gazebo::msgs::Set** (msgs::Color *_c, const common::Color &_v)
Set a `msgs::Color` from a `common::Color` (p. 193).
- void **gazebo::msgs::Set** (msgs::Time *_t, const common::Time &_v)
Set a `msgs::Time` from a `common::Time` (p. 732).
- void **gazebo::msgs::Set** (msgs::PlaneGeom *_p, const math::Plane &_v)
Set a `msgs::Plane` from a `math::Plane` (p. 547).
- 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.

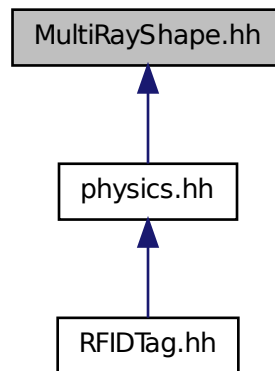
11.89 MultiRayShape.hh File Reference

```
#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**

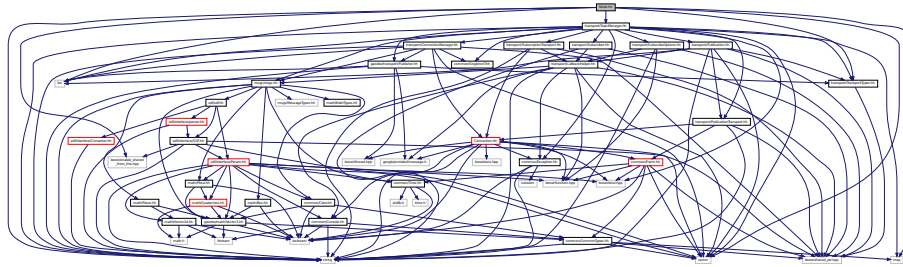
Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

Namespaces

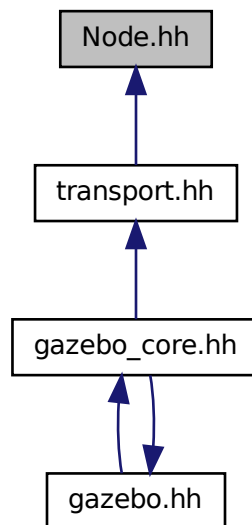
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.90 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 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**

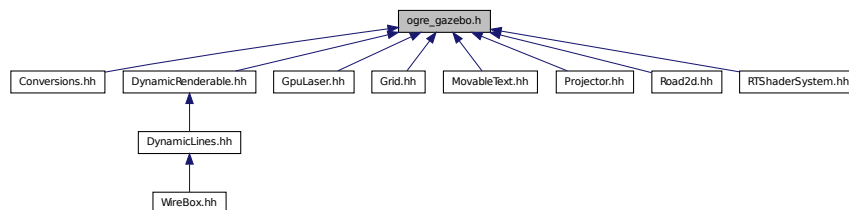
11.91 ogre_gazebo.h File Reference

```
#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 dependency graph for ogre_gazebo.h:

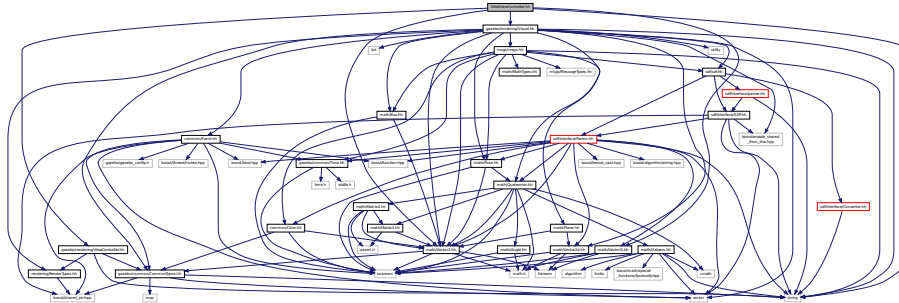


This graph shows which files directly or indirectly include this file:



11.92 OrbitViewController.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/rendering/ViewController.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2i.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.

11.93 Param.hh File Reference

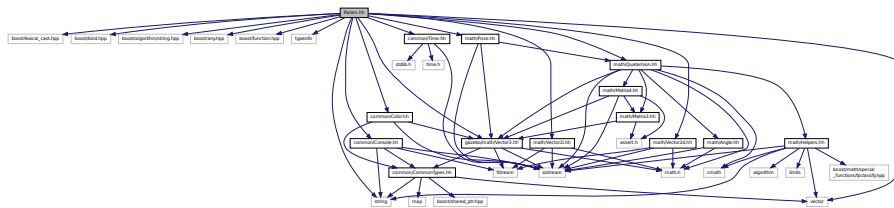
```
#include <boost/lexical_cast.hpp>
```

```

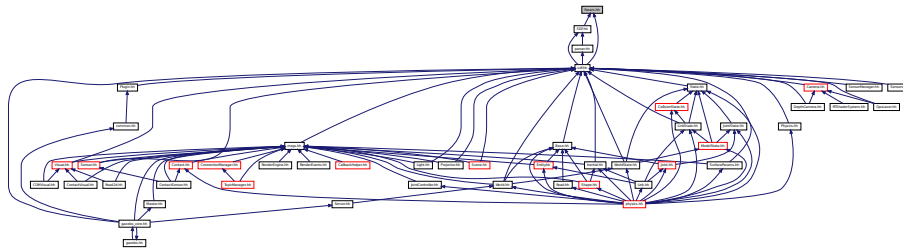
#include <boost/bind.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/any.hpp>
#include <boost/function.hpp>
#include <typeinfo>
#include <string>
#include <vector>
#include "common/Console.hh"
#include "common/Color.hh"
#include "common/Time.hh"
#include "math/Vector3.hh"
#include "math/Vector2i.hh"
#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 parameter class.
- class **sdf::ParamT< T >**
Templatized parameter class.

Namespaces

- namespace **sdf**
namespace for Simulation Description Format parser

- bool **sdf::initDoc** (TiXmlDocument *_xmlDoc, ElementPtr _sdf)
- bool **sdf::initFile** (const std::string &_filename, SDFPtr _sdf)
- bool **sdf::initFile** (const std::string &_filename, ElementPtr _sdf)
- bool **sdf::initString** (const std::string &_xmlString, SDFPtr _sdf)
- bool **sdf::initXml** (TiXmlElement *_xml, ElementPtr _sdf)
- bool **sdf::readDoc** (TiXmlDocument *_xmlDoc, SDFPtr _sdf, const std::string &_source)

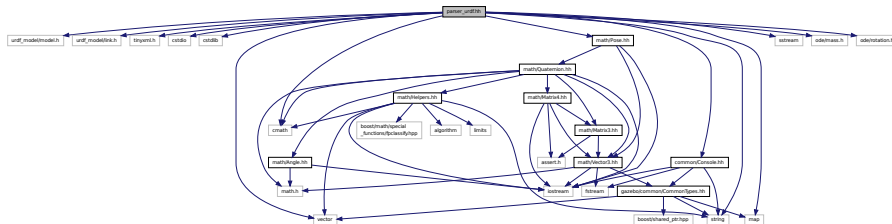
*Populate the **SDF** (p. 649) values from a TinyXML document.*
- bool **sdf::readDoc** (TiXmlDocument *_xmlDoc, ElementPtr _sdf, const std::string &_source)
- bool **sdf::readFile** (const std::string &_filename, SDFPtr _sdf)

*Populate the **SDF** (p. 649) values from a file.*
- bool **sdf::readString** (const std::string &_xmlString, SDFPtr _sdf)

*Populate the **SDF** (p. 649) values from a string.*
- bool **sdf::readString** (const std::string &_xmlString, ElementPtr _sdf)
- bool **sdf::readXml** (TiXmlElement *_xml, ElementPtr _sdf)

11.95 parser_urdf.hh File Reference

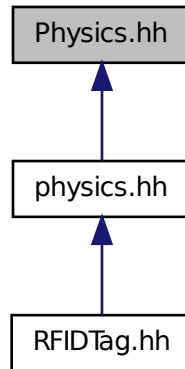
```
#include <urdf_model/model.h>
#include <urdf_model/link.h>
#include <tinyxml.h>
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <vector>
#include <string>
#include <sstream>
#include <map>
#include "ode/mass.h"
#include "ode/rotation.h"
#include "math/Pose.hh"
#include "common/Console.hh"
Include dependency graph for parser_urdf.hh:
```



Classes

- class **urdf2gazebo::GazeboExtension**
- class **urdf2gazebo::URDF2Gazebo**

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

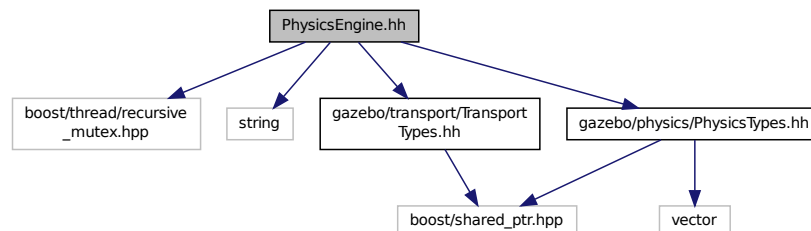
- WorldPtr **gazebo::physics::create_world** (const std::string &_name="")
Create a world given a name.
- bool **gazebo::physics::fini** ()
*Finalize transport by calling **gazebo::transport::fini** (p. 71).*
- WorldPtr **gazebo::physics::get_world** (const std::string &_name="")
Returns a pointer to 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. 730)'s and call **gazebo::transport::init** (p. 72).*
- void **gazebo::physics::load_world** (WorldPtr _world, sdf::ElementPtr _sdf)
*Load world from **sdf::Element** (p. 258) pointer.*
- void **gazebo::physics::load_worlds** (sdf::ElementPtr _sdf)
*load multiple worlds from single **sdf::Element** (p. 258) pointer*
- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)
*Pause world by calling **World::SetPaused** (p. 863).*

- 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. 862) 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. 863) given a pointer to it.*
- void **gazebo::physics::stop_worlds** ()
stop multiple worlds stored in static variable gazebo::g_worlds

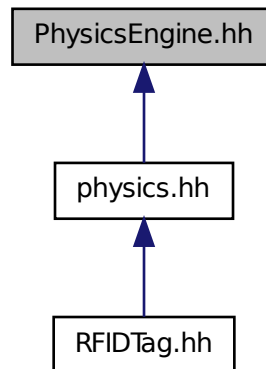
11.97 PhysicsEngine.hh File Reference

```
#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. 125) class for a physics engine.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

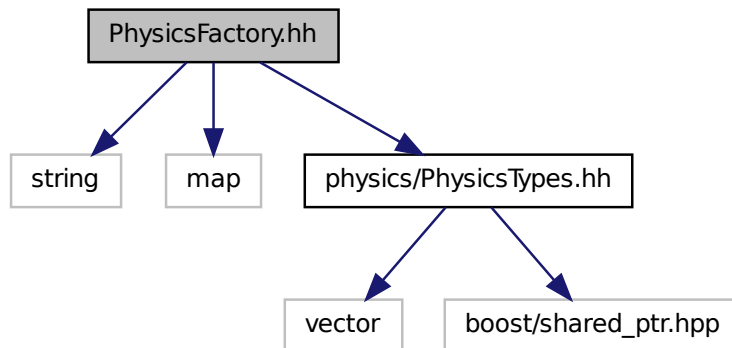
- namespace **gazebo::physics**

namespace for physics

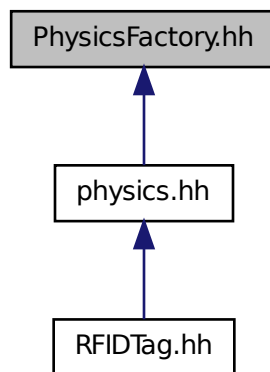
11.98 PhysicsFactory.hh File Reference

```
#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.

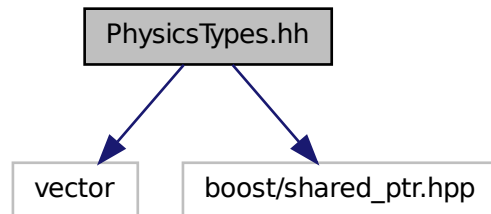
Typedefs

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

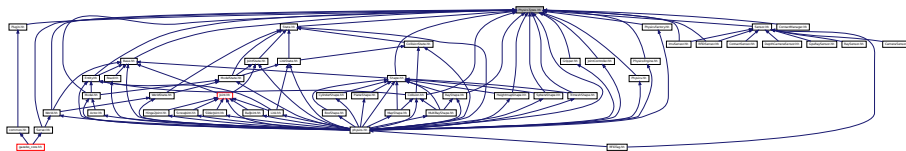
11.99 PhysicsTypes.hh File Reference

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 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**

11.99.1 Detailed Description

default namespace for gazebo

11.99.2 Macro Definition Documentation

11.99.2.1 #define GZ_ALL_COLLIDE 0xFFFFFFFF

Default collision bitmask.

Collision objects will collide with everything.

11.99.2.2 #define GZ_FIXED_COLLIDE 0x00000001

Collision object will collide only with fixed objects.

11.99.2.3 #define GZ_GHOST_COLLIDE 0x10000000

Collides with everything else but other ghost.

11.99.2.4 #define GZ_NONE_COLLIDE 0x00000000

Collision object will collide with nothing.

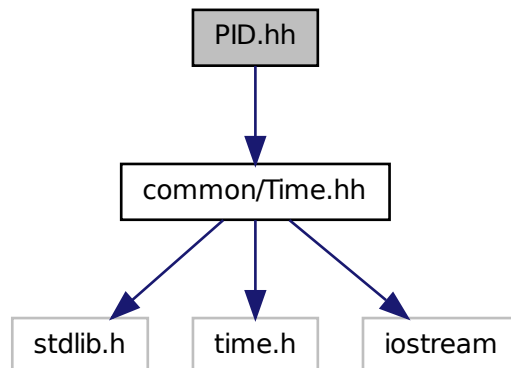
11.99.2.5 #define GZ_SENSOR_COLLIDE 0x00000003

Collision object will collide only with sensors.

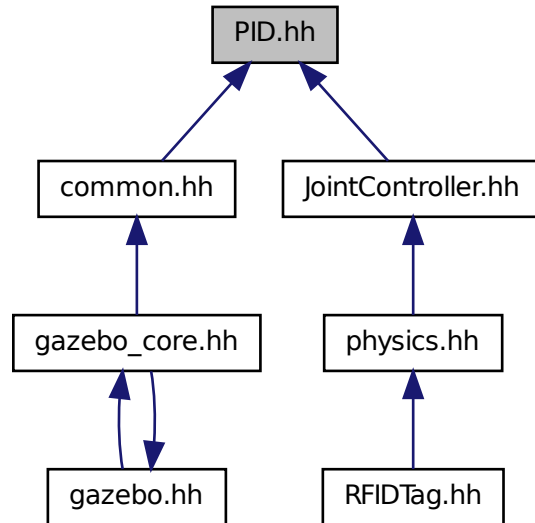
11.100 PID.hh File Reference

```
#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. 543) controller class.*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

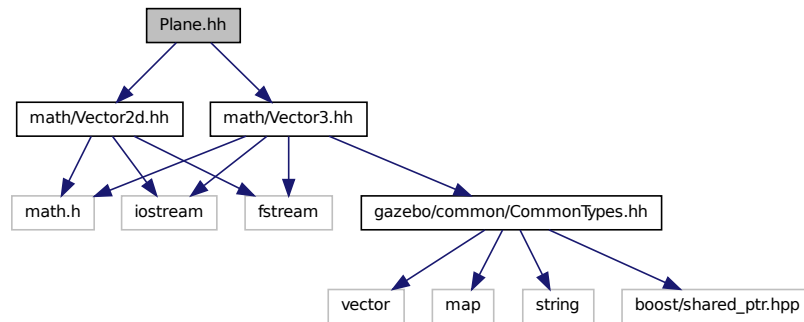
- namespace **gazebo::common**

Common namespace.

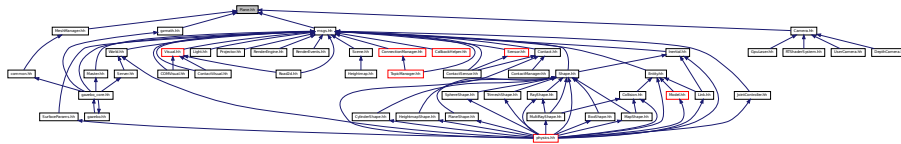
11.101 Plane.hh File Reference

```
#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 plane and related functions.

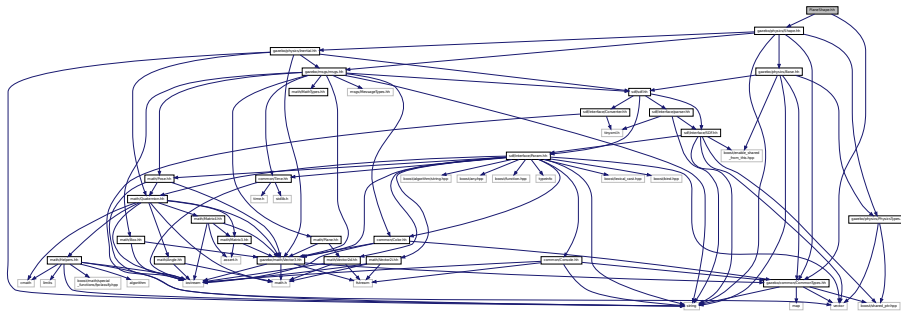
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

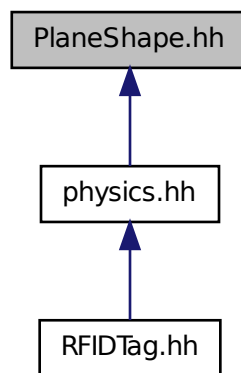
11.102 PlaneShape.hh File Reference

```
#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. 180) for an infinite plane.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

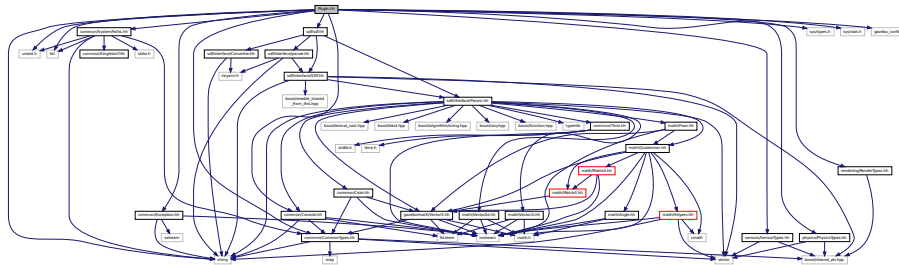
11.103 Plugin.hh File Reference

```

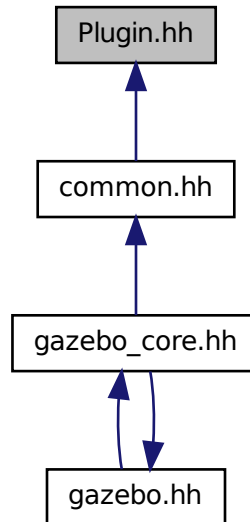
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <gazebo_config.h>
#include <list>
#include <string>
#include "common/CommonTypes.hh"
#include "common/SystemPaths.hh"
#include "common/Console.hh"
#include "common/Exception.hh"
#include "physics/PhysicsTypes.hh"
#include "sensors/SensorTypes.hh"
#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 plugin with access to **physics::Model** (p. 460).*
- class **gazebo::PluginT**< T >
A class which all plugins must inherit from.
- class **gazebo::SensorPlugin**
*A plugin with access to **physics::Sensor**.*
- class **gazebo::SystemPlugin**
A plugin loaded within the gzserver on startup.
- class **gazebo::VisualPlugin**
A plugin loaded within the gzserver on startup.
- class **gazebo::WorldPlugin**
*A plugin with access to **physics::World** (p. 853).*

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.

11.103.1 Macro Definition Documentation

11.103.1.1 #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

11.103.1.2 #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

11.103.1.3 #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

11.103.1.4 #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

11.103.1.5 #define GZ_REGISTER_WORLD_PLUGIN(*classname*)**Value:**

```
extern "C" gazebo::WorldPlugin *RegisterPlugin();    gazebo::WorldPlugin *RegisterPlugin() \
{\
    return new classname();\
}
```

Plugin registration function for world plugin.

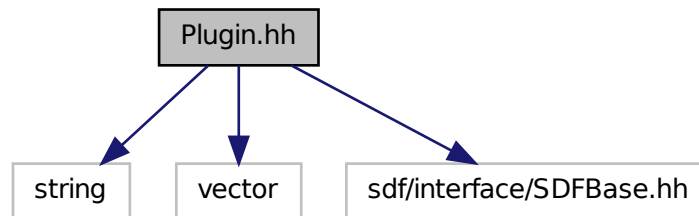
Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.104 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

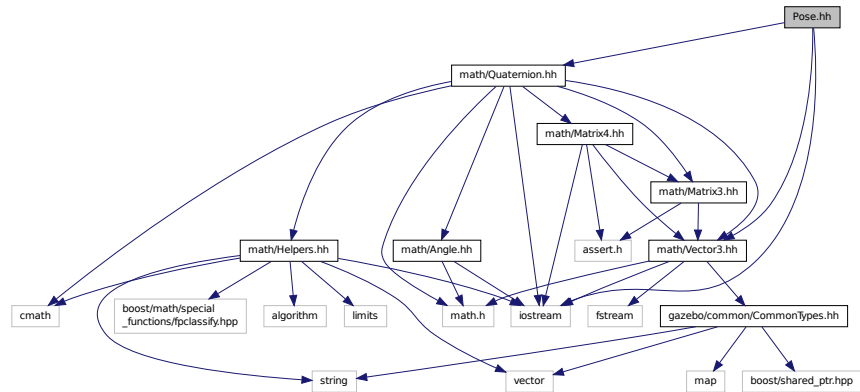
- namespace `sdf`

namespace for Simulation Description Format parser

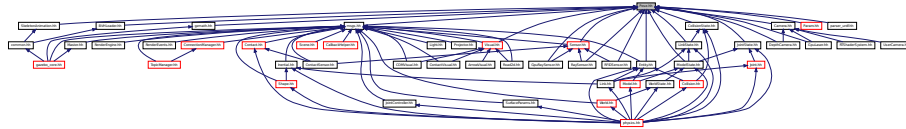
11.105 Pose.hh File Reference

```
#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.

11.106 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.

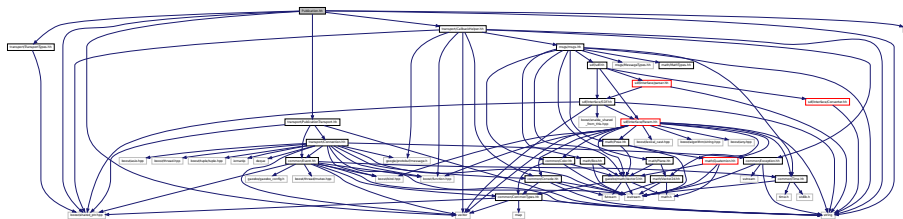
- namespace **gazebo::rendering**

Rendering namespace.

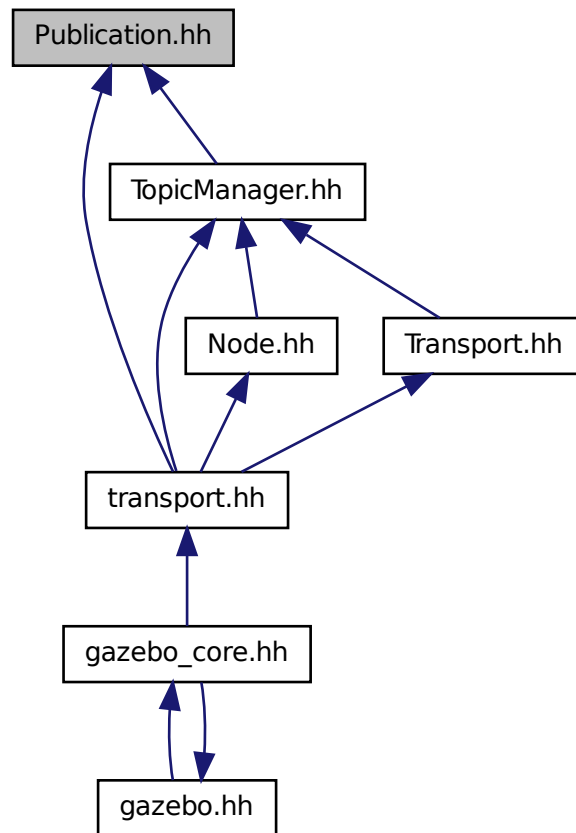
11.107 Publication.hh File Reference

```
#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 publication for a topic.

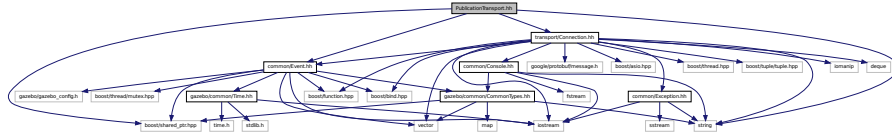
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

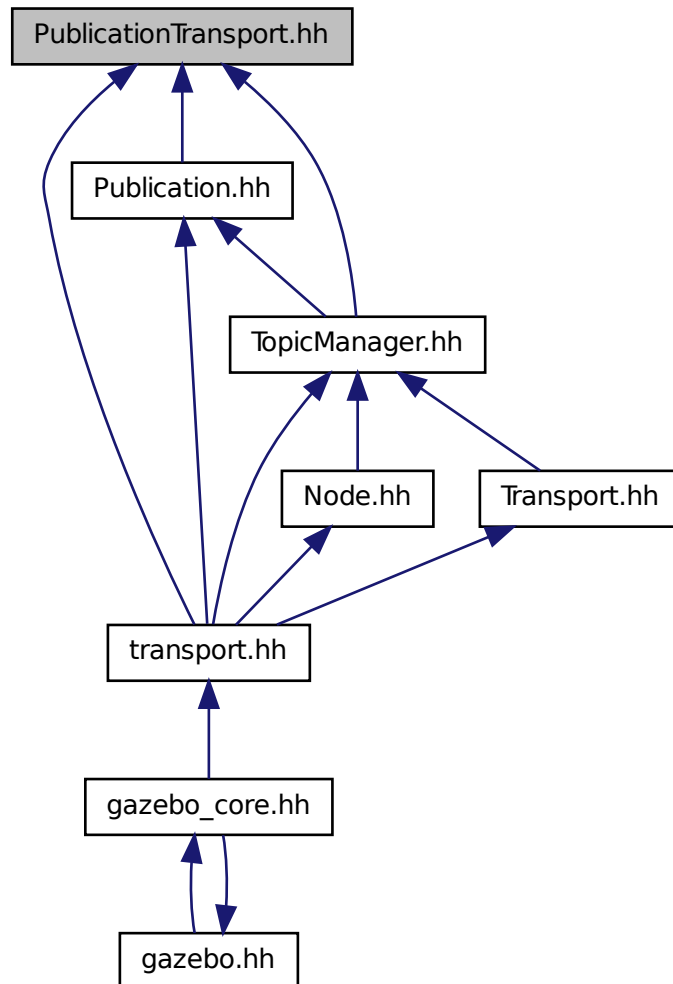
11.108 PublicationTransport.hh File Reference

```
#include <boost/shared_ptr.hpp>
```

```
#include <string>
#include "transport/Connection.hh"
#include "common/Event.hh"
Include dependency graph for PublicationTransport.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::PublicationTransport**

transport/transport.hh

Namespaces

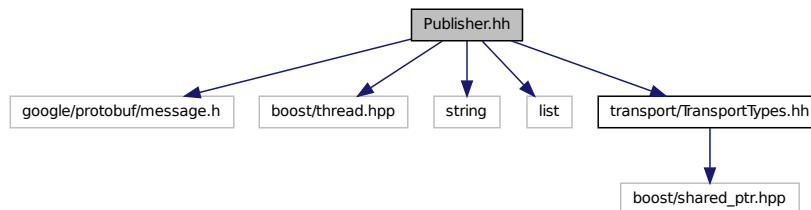
- namespace **gazebo**

Forward declarations for the common classes.

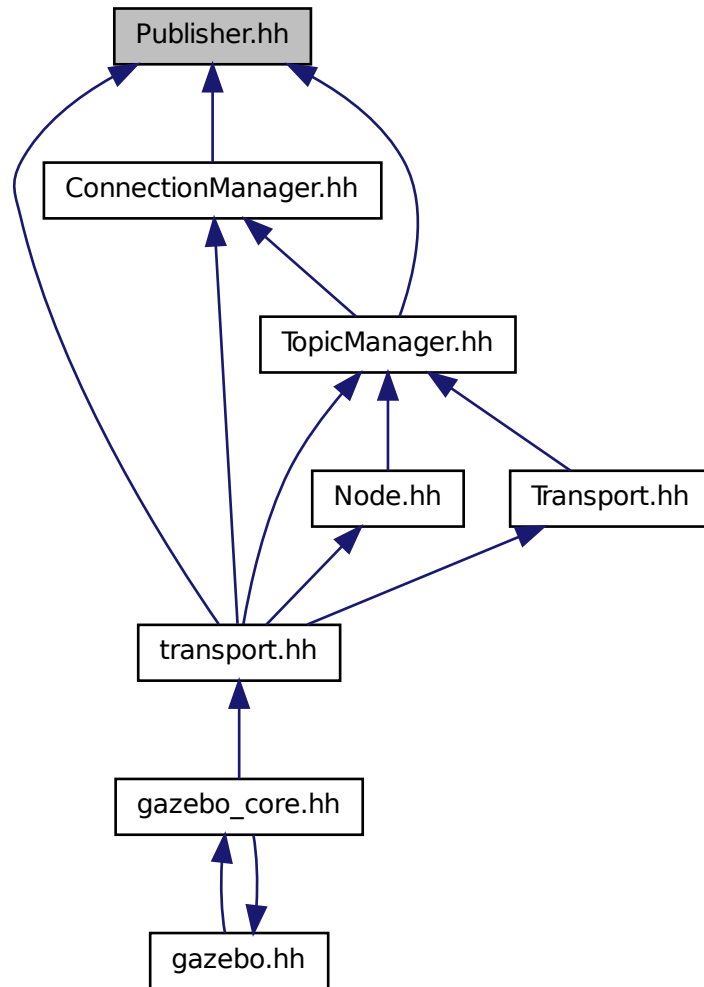
- namespace **gazebo::transport**

11.109 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 publisher of messages on a topic.

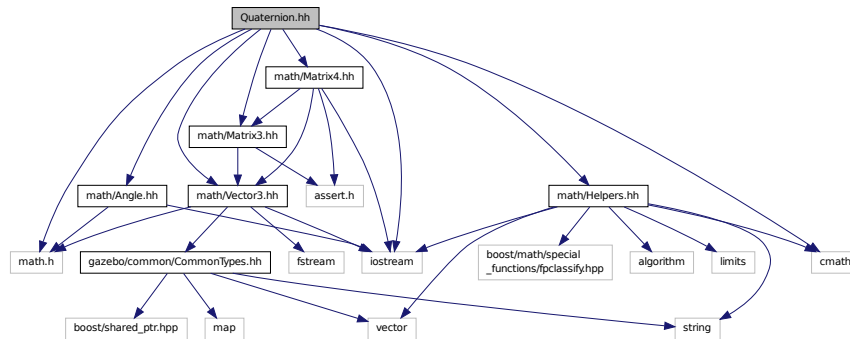
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

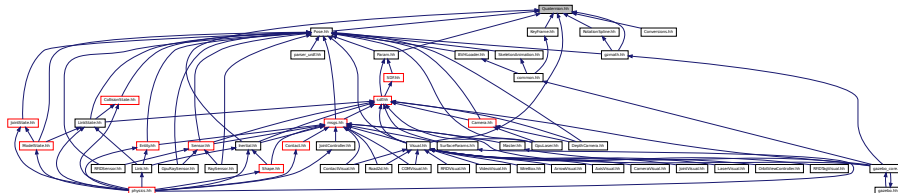
11.110 Quaternion.hh File Reference

```
#include <math.h>
#include <iostream>
#include <cmath>
#include "math/Helpers.hh"
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "math/Matrix3.hh"
#include "math/Matrix4.hh"
```

Include dependency graph for Quaternion.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Quaternion**

A quaternion class.

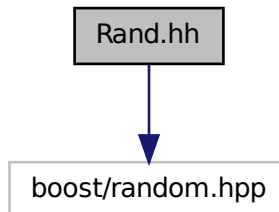
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

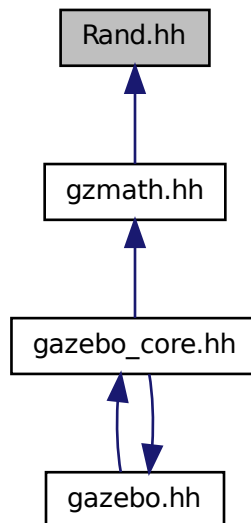
11.111 Rand.hh File Reference

```
#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.

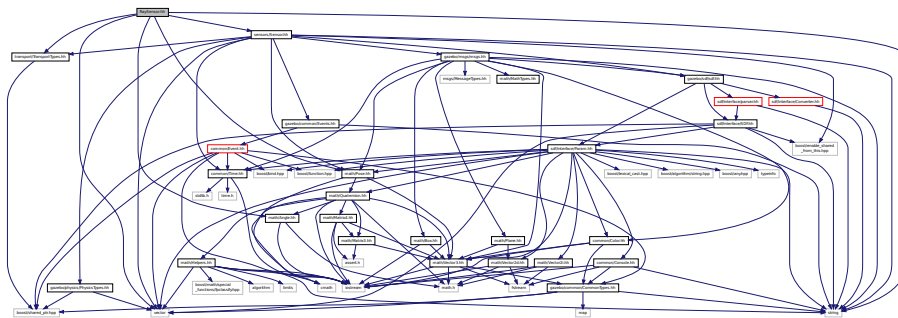
Typedefs

- 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**

11.112 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. 652) with one or more rays.

Namespaces

- namespace **gazebo**

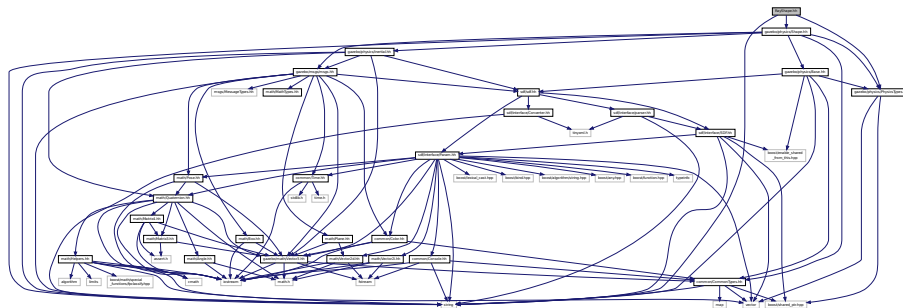
Forward declarations for the common classes.

- namespace **gazebo::sensors**

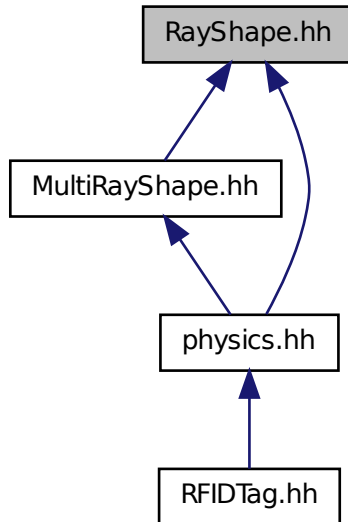
Sensors namespace.

11.113 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. 125) class for Ray collision geometry.

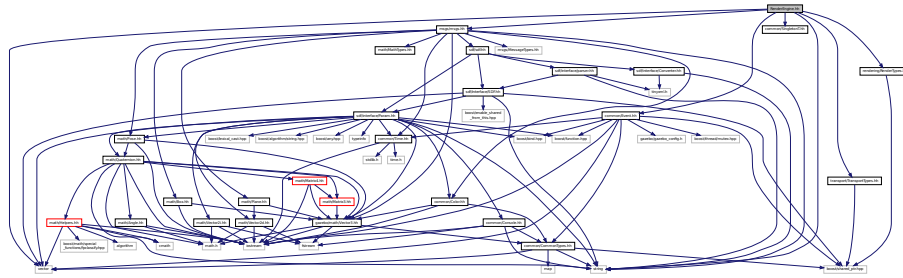
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.114 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.

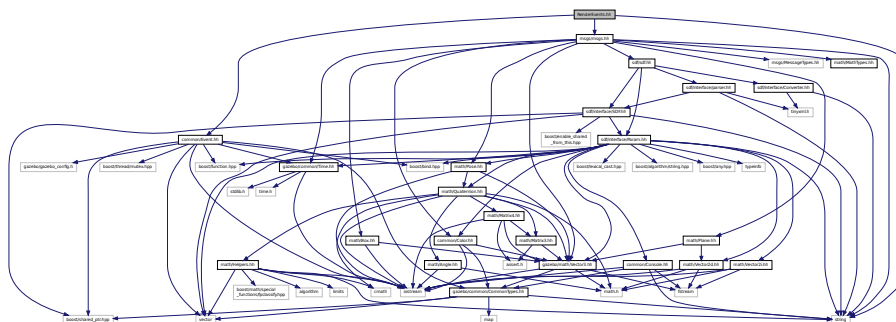
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.115 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**

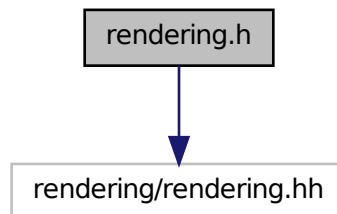
Forward declarations for the common classes.

- namespace **gazebo::rendering**

Rendering namespace.

11.116 rendering.h File Reference

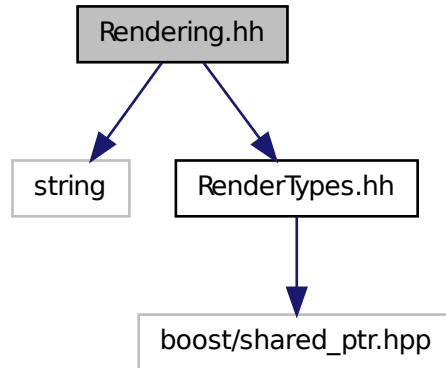
```
#include "rendering/rendering.hh"  
Include dependency graph for rendering.h:
```



11.117 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

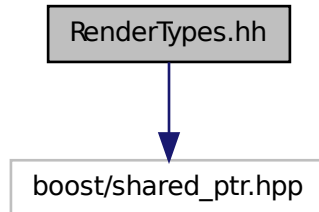
- rendering::ScenePtr **gazebo::rendering::create_scene** (const std::string &_name, bool _enableVisualizations)

*create **rendering::Scene** (p. 632) 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. 632) 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. 632) by name*

11.118 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.

11.118.1 Macro Definition Documentation

11.118.1.1 #define GZ_VISIBILITY_ALL 0xFFFFFFFF

Render everything visibility mask.

11.118.1.2 #define GZ_VISIBILITY_GUI 0x00000001

Render GUI visuals mask.

11.118.1.3 #define GZ_VISIBILITY_NOT_SELECTABLE 0x00000002

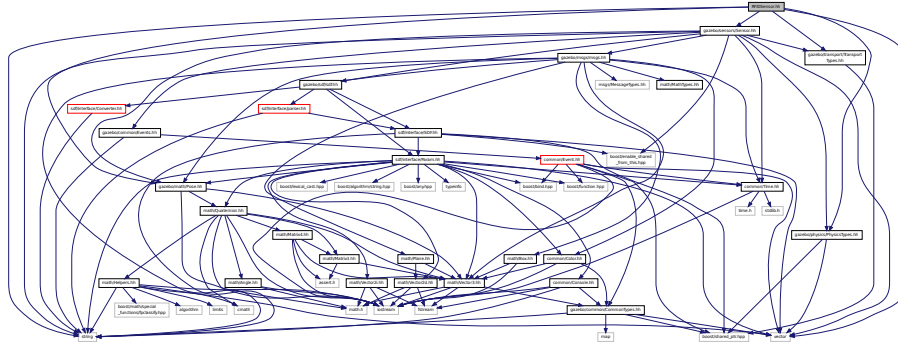
Render visuals that are not selectable mask.

11.118.1.4 #define GZ_VISIBILITY_SELECTION 0x10000000

Renders only objects that can be selected.

11.119 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. 652) class for RFID type of sensor.

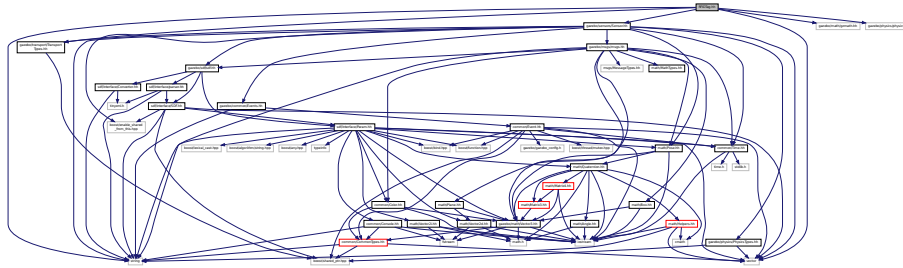
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.120 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. 616) to interact with *RFIDTagSensors*.

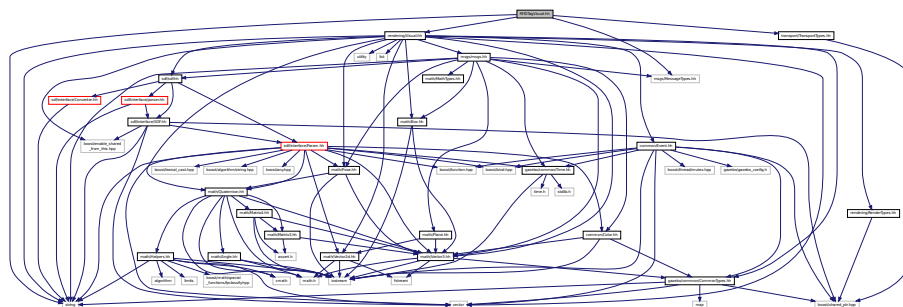
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.121 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

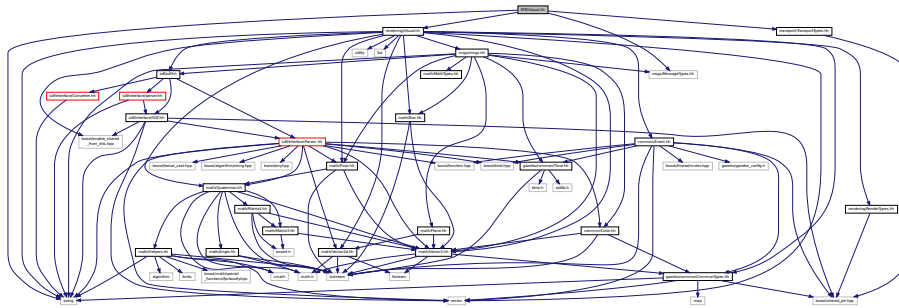
- class **gazebo::rendering::RFIDTagVisual**
Visualization for RFID tags sensor.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.122 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.

11.123 Road.hh File Reference

```
#include <vector>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/Base.hh"
```


11.124 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`

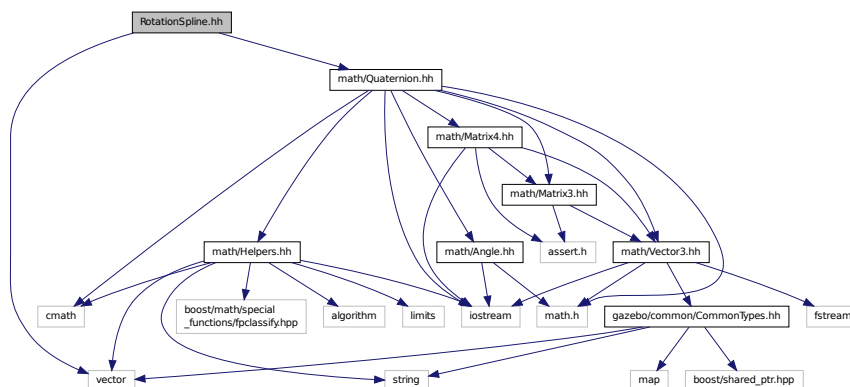
Namespaces

- namespace `gazebo`
Forward declarations for the common classes.
- namespace `gazebo::rendering`
Rendering namespace.

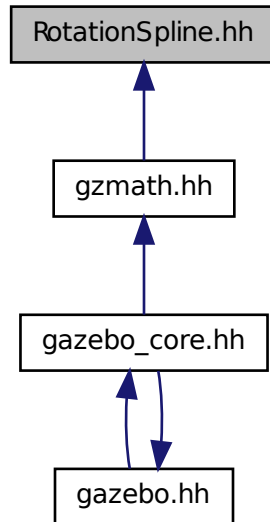
11.125 RotationSpline.hh File Reference

```
#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. 697) for rotations.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

11.126 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. 98)'s Run-Time Shader system.*

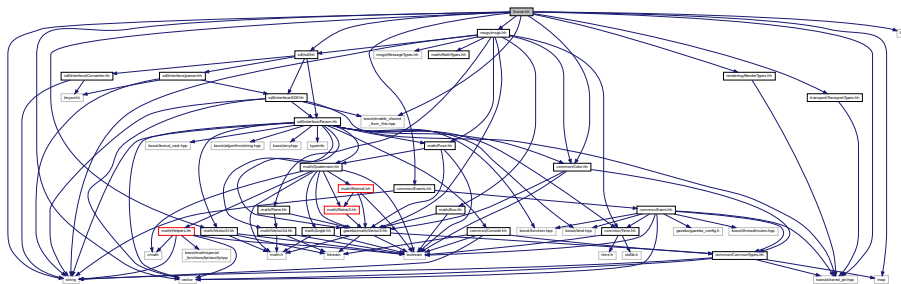
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

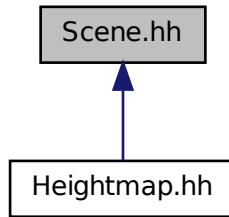
11.127 Scene.hh File Reference

```
#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

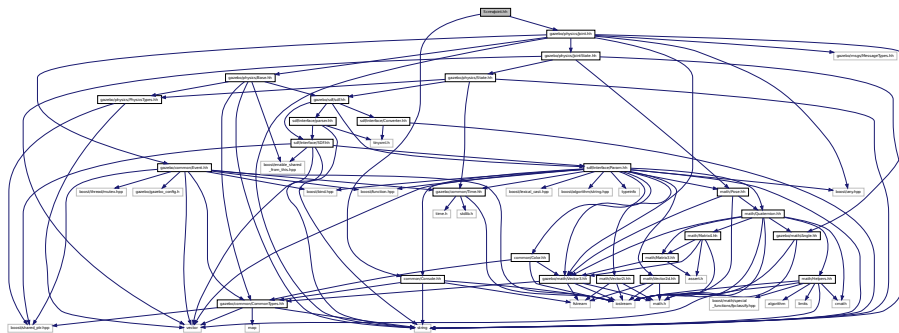
- class **gazebo::rendering::Scene**
Representation of an entire scene graph.

Namespaces

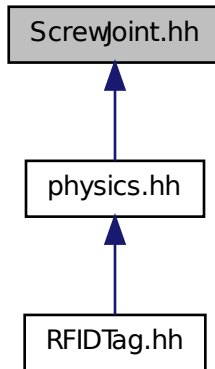
- namespace **boost**
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**
- namespace **SkyX**

11.128 ScrewJoint.hh File Reference

```
#include "gazebo/physics/Joint.hh"
#include "gazebo/common/Console.hh"
Include dependency graph for ScrewJoint.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ScrewJoint**< T >

A screw joint, which has both prismatic and rotational DOFs.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

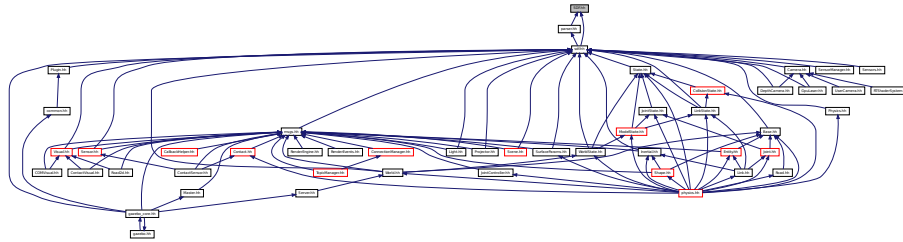
- namespace **gazebo::physics**

namespace for physics

11.129 sdf.hh File Reference

```
#include "sdf/interface/SDF.hh"  
#include "sdf/interface/Param.hh"  
#include "sdf/interface/parser.hh"  
#include "sdf/interface/Converter.hh"
```


This graph shows which files directly or indirectly include this file:



Classes

- class **sdf::Element**
SDF (p. 649) *Element* (p. 258) class.
- class **sdf::SDF**
Base SDF (p. 649) 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`

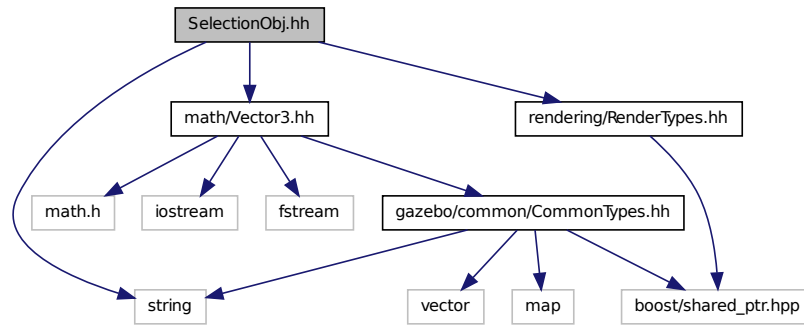
11.130.1 Macro Definition Documentation

11.130.1.1 `#define SDF_VERSION "1.3"`

11.131 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 graphical selection object.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

Rendering namespace.

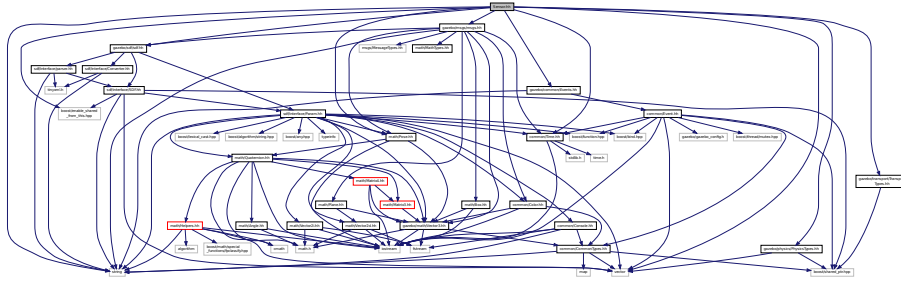
11.132 Sensor.hh File Reference

```

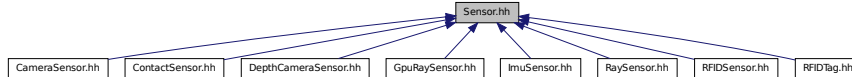
#include <boost/enable_shared_from_this.hpp>
#include <vector>
#include <string>
#include "gazebo/sdf/sdf.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/transport/TransportTypes.hh"

```


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.

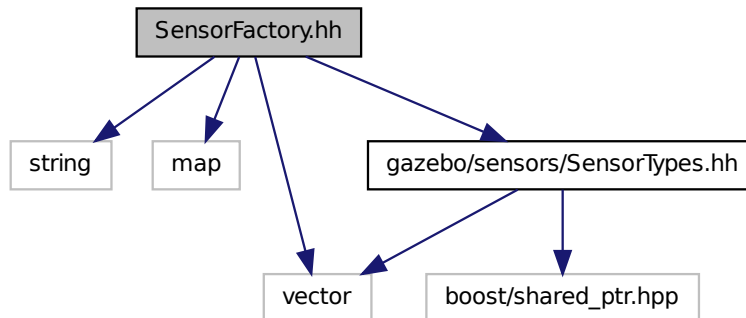
- namespace **gazebo::sensors**

Sensors namespace.

11.133 SensorFactory.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "gazebo/sensors/SensorTypes.hh"
```

Include dependency graph for SensorFactory.hh:



Classes

- class **gazebo::sensors::SensorFactory**

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

Macros

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

Typedefs

- typedef Sensor *(* **gazebo::sensors::SensorFactoryFn**)()

11.134 SensorManager.hh File Reference

```

#include <boost/thread.hpp>
#include <string>
#include <vector>
#include "common/SingletonT.hh"
#include "sensors/SensorTypes.hh"
#include "sdf/sdf.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=true)

Run the sensor generation one step.

- void **gazebo::sensors::stop** ()

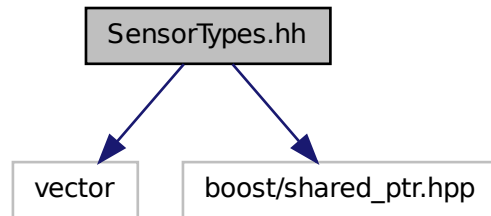
Stop the sensor generation loop.

11.136 SensorTypes.hh File Reference

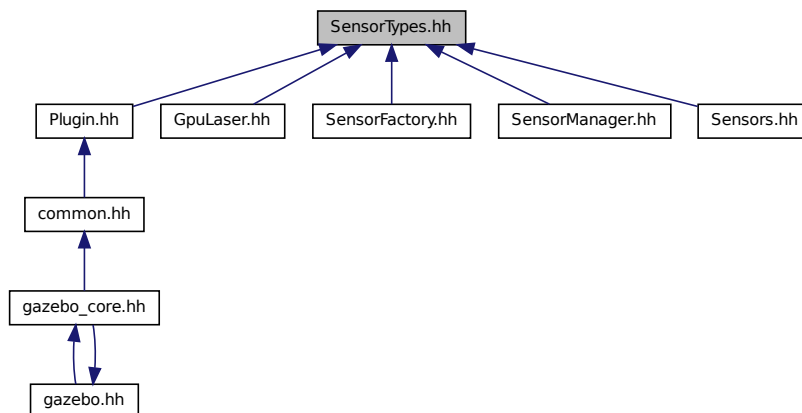
Forward declarations and typedefs for sensors.

```
#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**

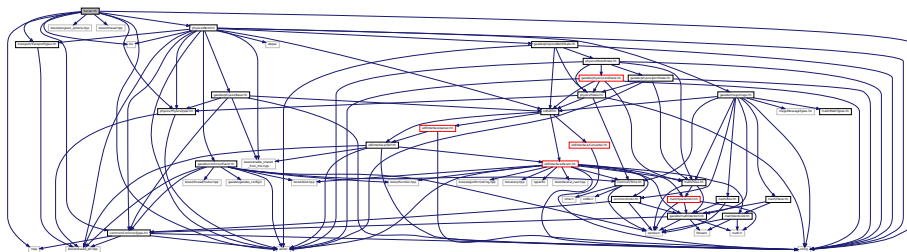
11.136.1 Detailed Description

Forward declarations and typedefs for sensors.

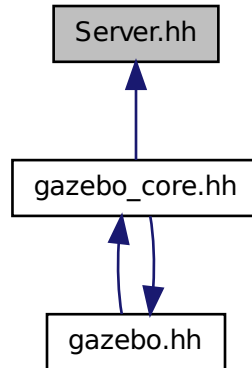
11.137 Server.hh File Reference

```
#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

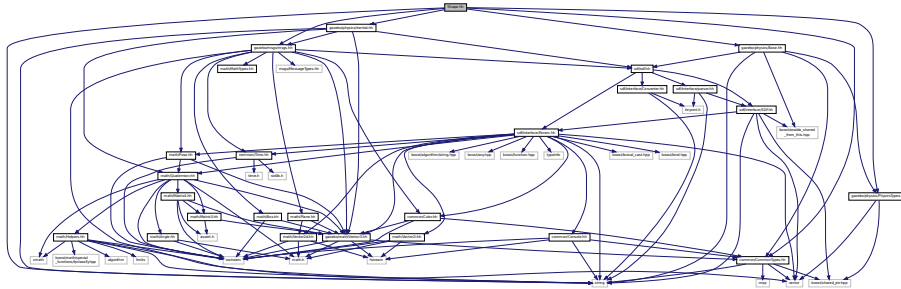
- namespace **boost**
- namespace **gazebo**

Forward declarations for the common classes.

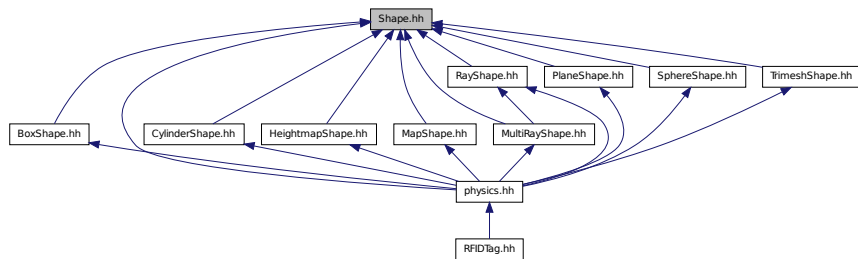
11.138 Shape.hh File Reference

```
#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. 125) class for all shapes.

Namespaces

- namespace **gazebo**

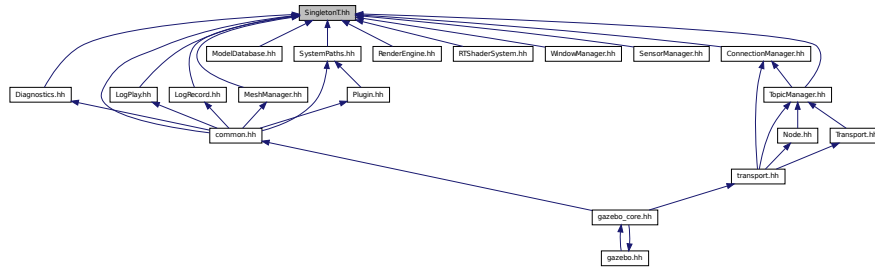
Forward declarations for the common classes.

- namespace **gazebo::physics**

namespace for physics

11.139 SingletonT.hh File Reference

This graph shows which files directly or indirectly include this file:

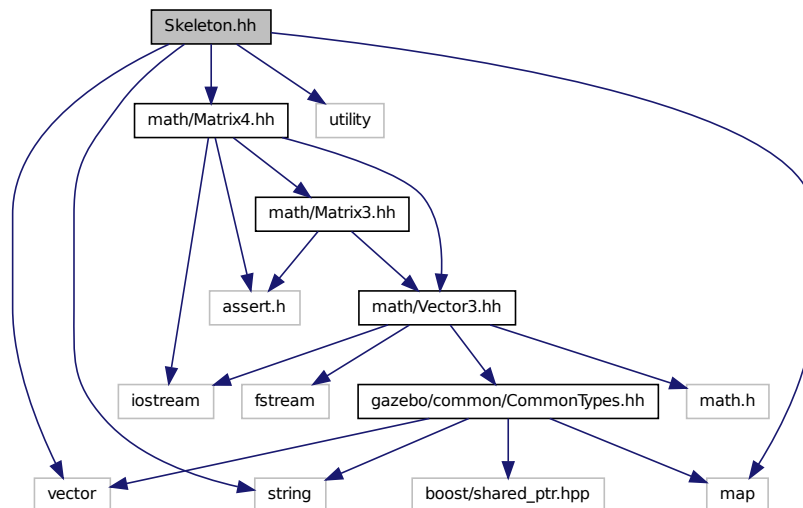


Classes

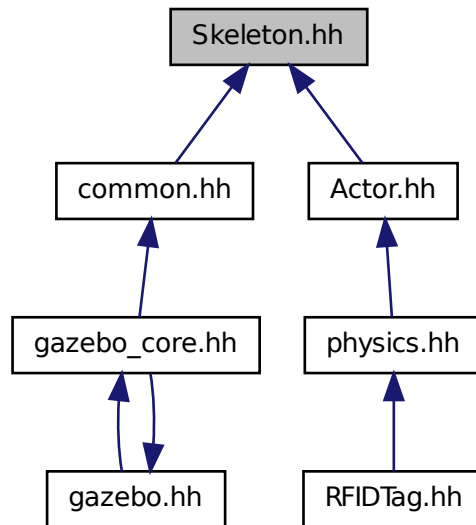
- class **SingletonT** < T >
Singleton template class.

11.140 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. 509) *Skeleton.hh* (p. 1029) *common/common.hh*
- class **gazebo::common::Skeleton**
A skeleton.
- class **gazebo::common::SkeletonNode**
A skeleton node.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

Typedefs

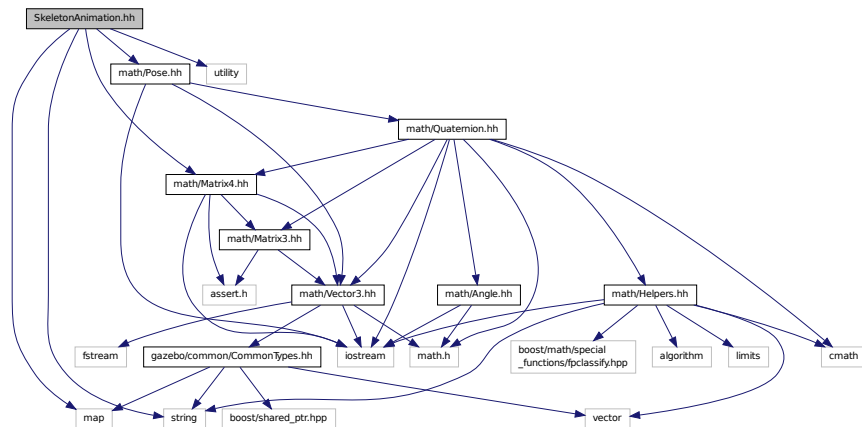
- typedef `std::map< unsigned int, SkeletonNode * >` **gazebo::common::NodeMap**
- typedef `std::map< unsigned int, SkeletonNode * >::iterator` **gazebo::common::NodeMapIter**

- typedef std::map< double,
std::vector< NodeTransform > > **gazebo::common::RawNodeAnim**
- typedef std::vector
< std::vector< std::pair
< std::string, double > > > **gazebo::common::RawNodeWeights**
- typedef std::map< std::string,
RawNodeAnim > **gazebo::common::RawSkeletonAnim**

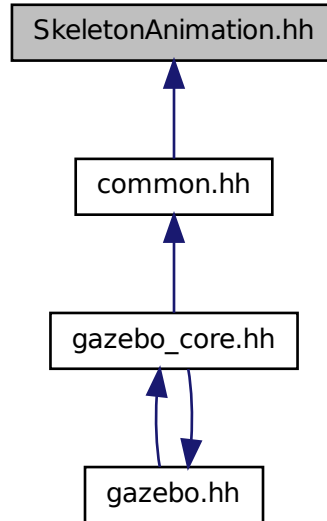
11.141 SkeletonAnimation.hh File Reference

```
#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. 672) animation.*

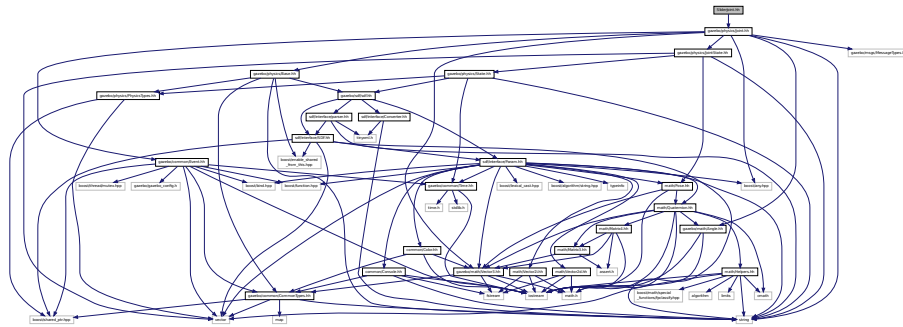
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

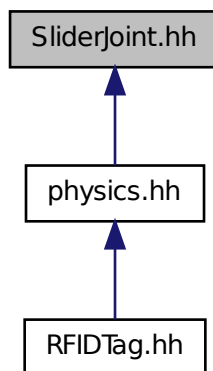
11.142 SliderJoint.hh File Reference

```
#include "gazebo/physics/Joint.hh"
```

Include dependency graph for SliderJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::SliderJoint**< T >
A slider joint.

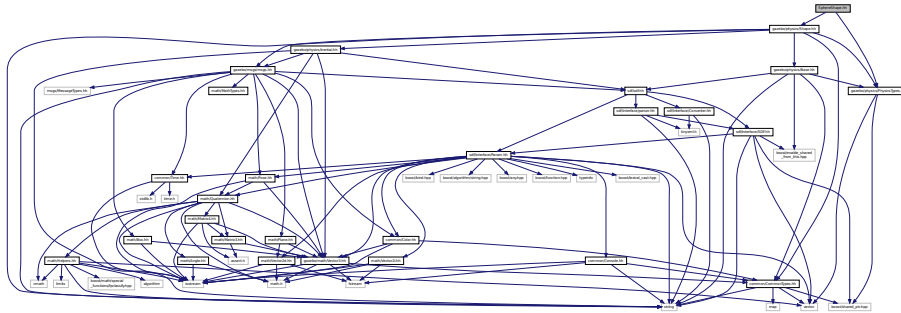
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

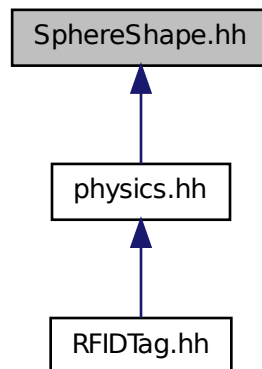
11.143 SphereShape.hh File Reference

```
#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

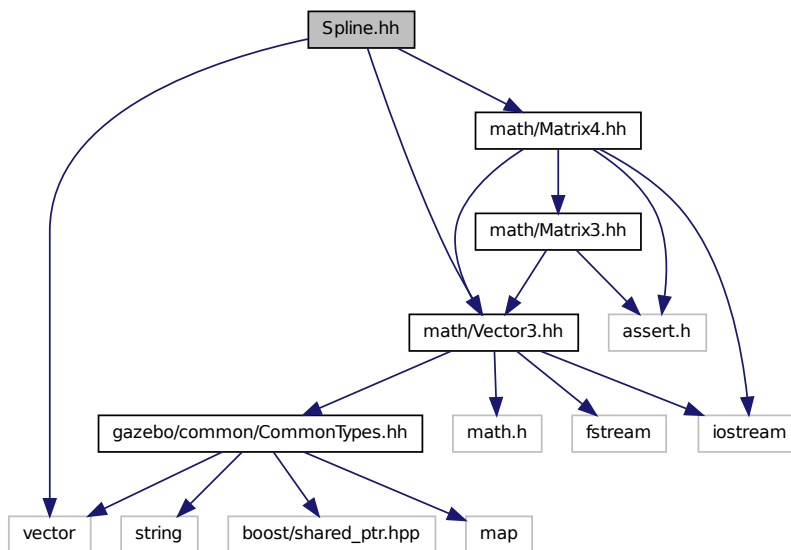
- class **gazebo::physics::SphereShape**
Sphere collision shape.

Namespaces

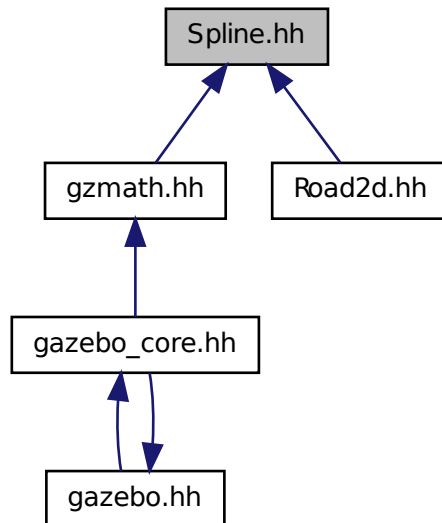
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.144 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.

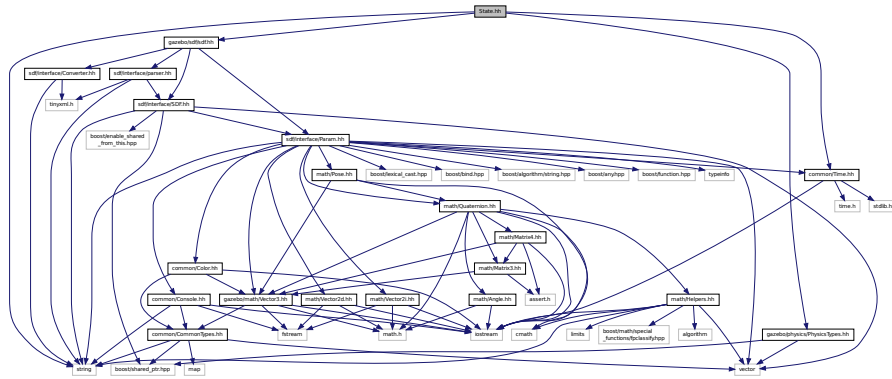
- namespace **gazebo::math**

Math namespace.

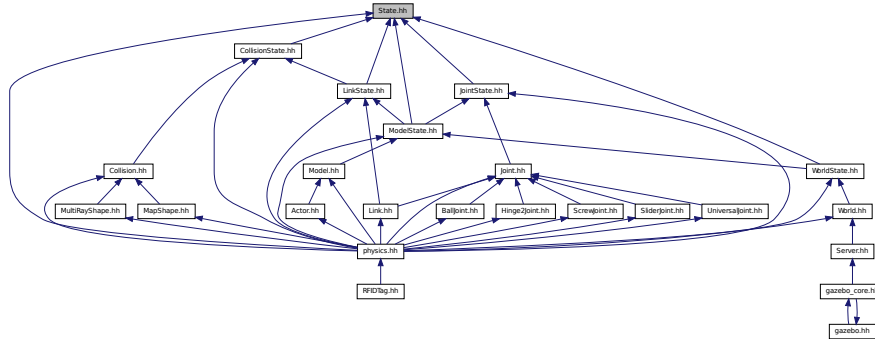
11.145 State.hh File Reference

```
#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. 702) of an entity.

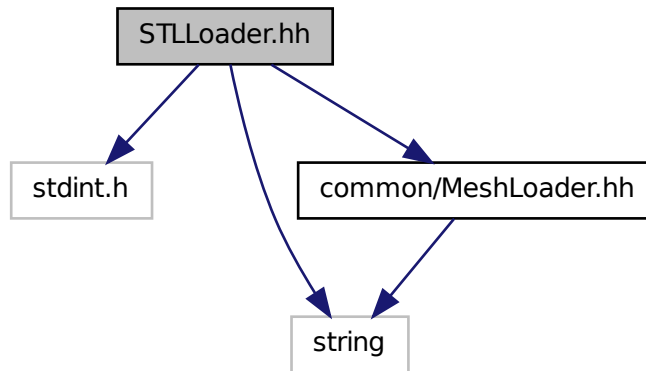
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

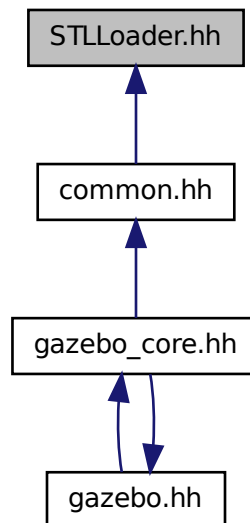
11.146 STLLoader.hh File Reference

```
#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

11.146.1 Macro Definition Documentation

11.146.1.1 #define COR3_MAX 200000

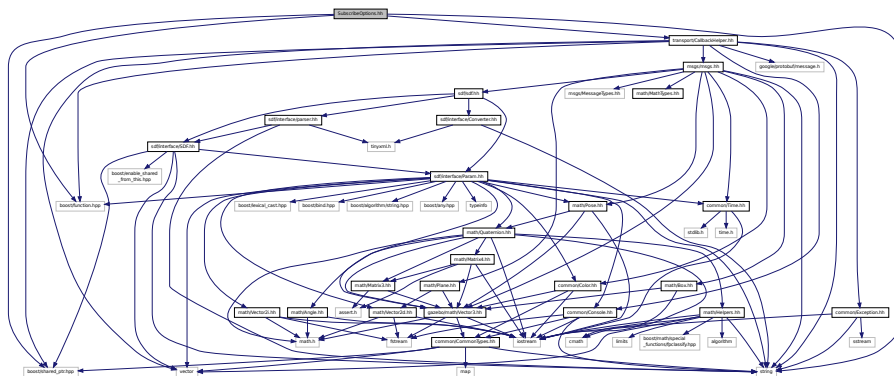
11.146.1.2 #define FACE_MAX 200000

11.146.1.3 #define LINE_MAX_LEN 256

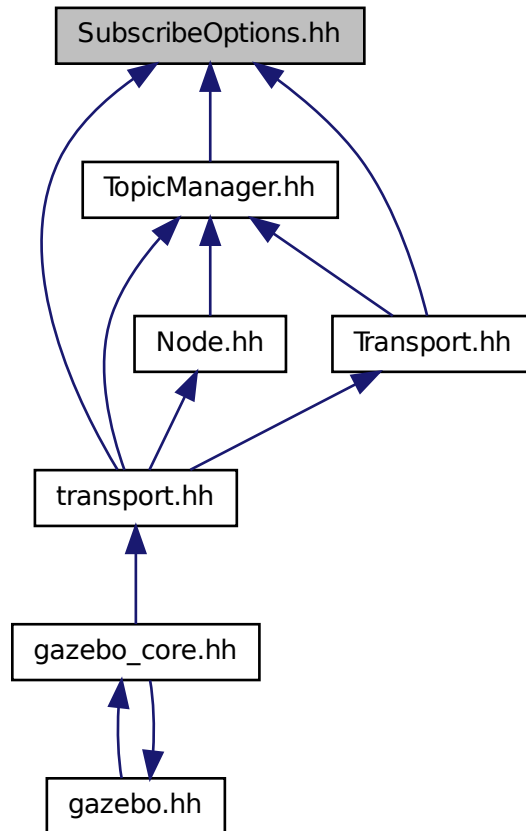
11.146.1.4 #define ORDER_MAX 10

11.147 SubscribeOptions.hh File Reference

```
#include <boost/function.hpp>
#include <boost/shared_ptr.hpp>
#include <string>
#include "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.

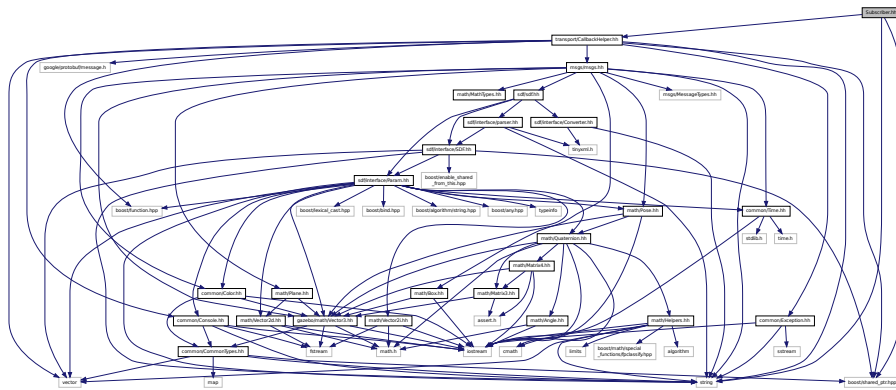
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

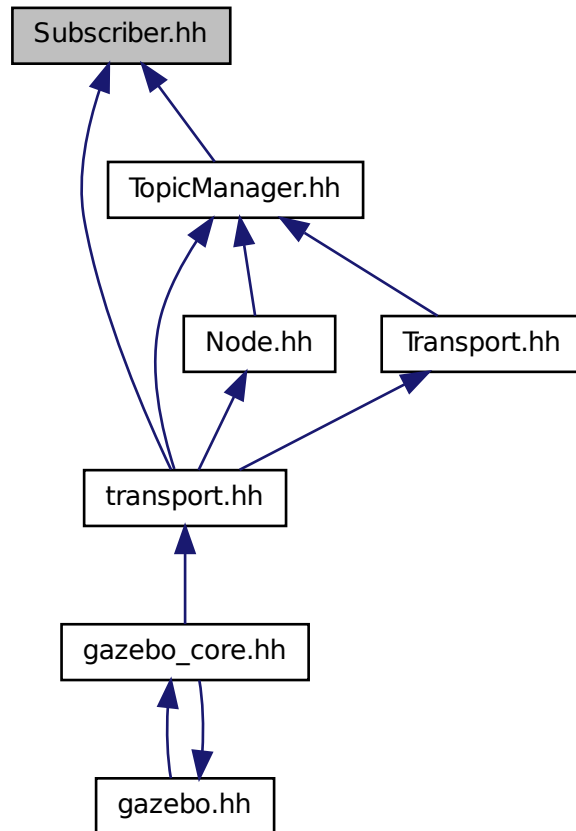
11.148 Subscriber.hh File Reference

```
#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 subscriber to a topic.

Namespaces

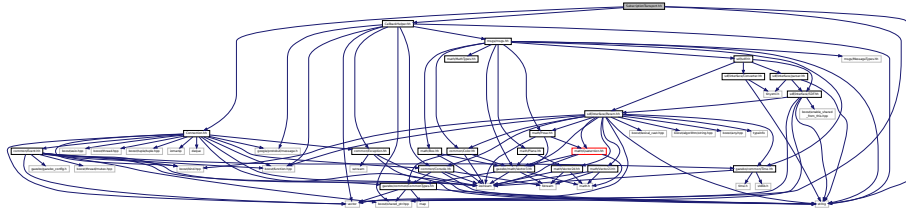
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

11.149 SubscriptionTransport.hh File Reference

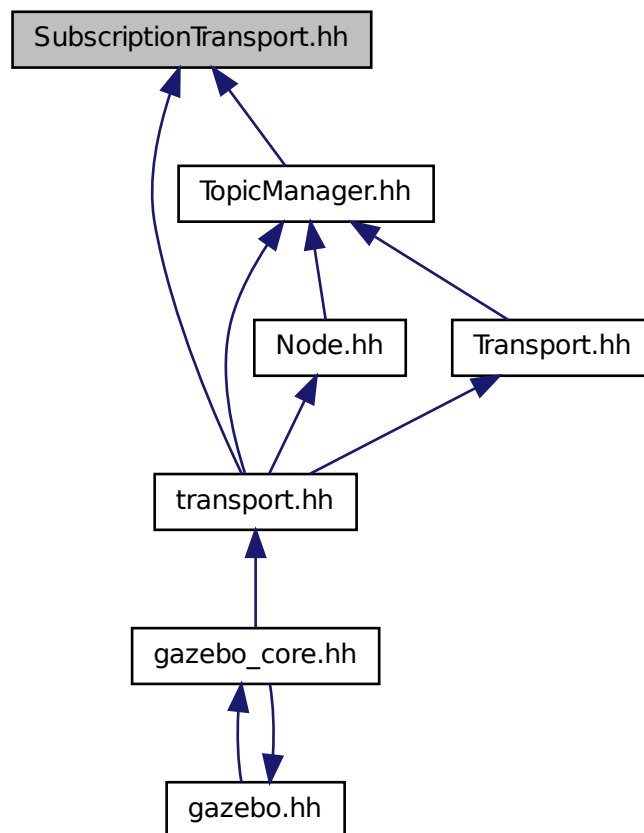
```
#include <boost/shared_ptr.hpp>
```

```
#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.

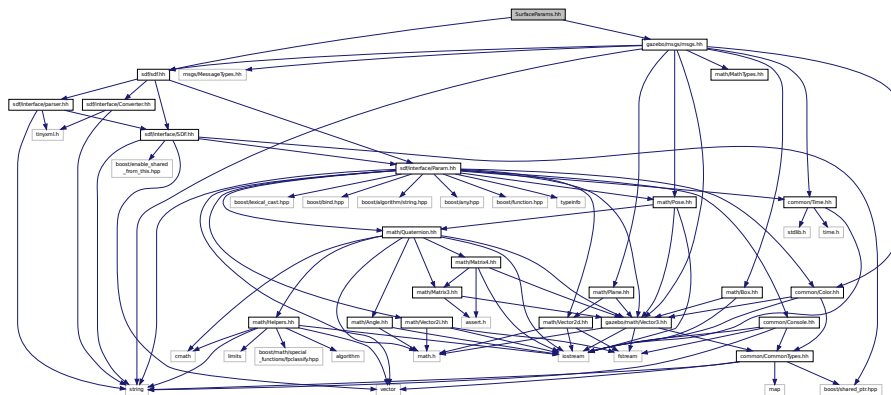
- namespace **gazebo::transport**

11.150 SurfaceParams.hh File Reference

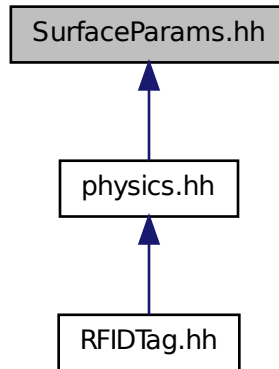
```
#include "gazebo/msgs/msgs.hh"
```

```
#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. 721) defines various Surface contact parameters.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

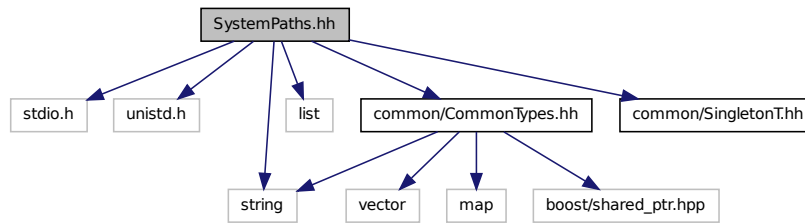
- namespace **gazebo::physics**

namespace for physics

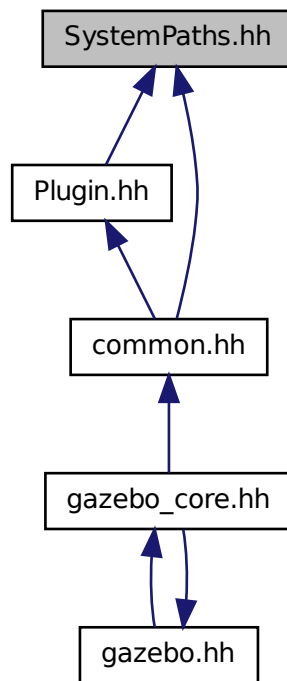
11.151 SystemPaths.hh File Reference

```
#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**

11.151.1 Macro Definition Documentation

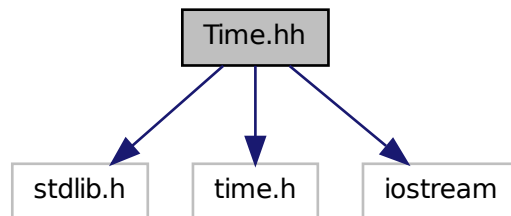
11.151.1.1 #define GetCurrentDir getcwd

11.151.1.2 #define LINUX

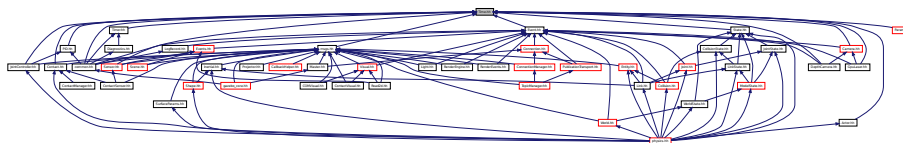
11.152 Time.hh File Reference

```
#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 **Time** (p. 732) class, can be used to hold wall- or sim-time.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

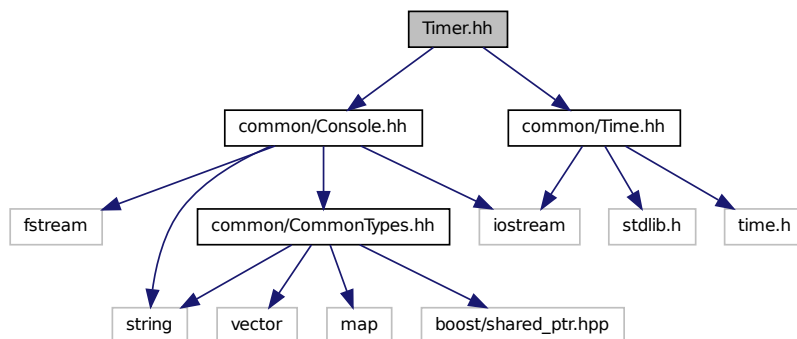
Common namespace.

11.153 Timer.hh File Reference

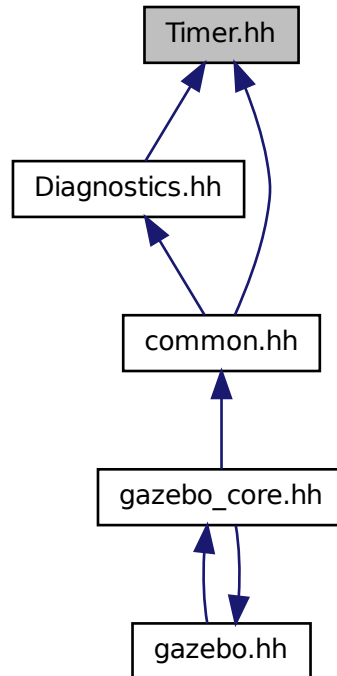
```
#include "common/Console.hh"
```

```
#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 timer class, used to time things in real world walltime.

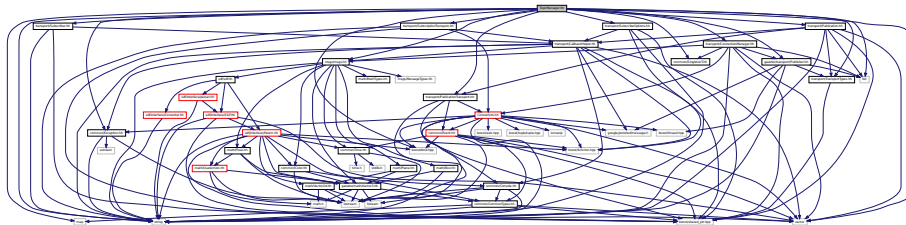
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

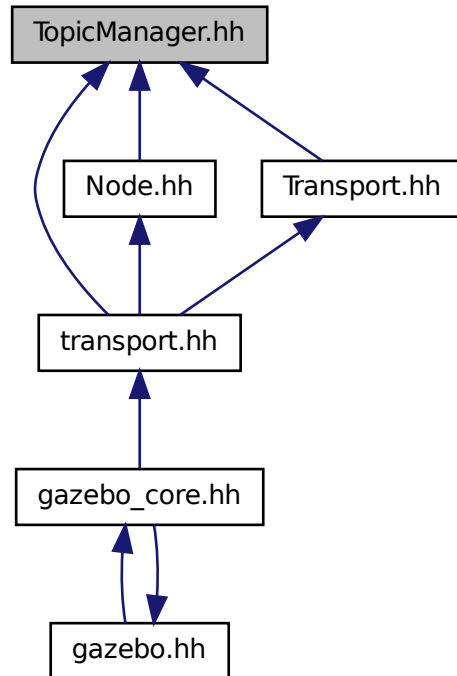
11.154 TopicManager.hh File Reference

```
#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

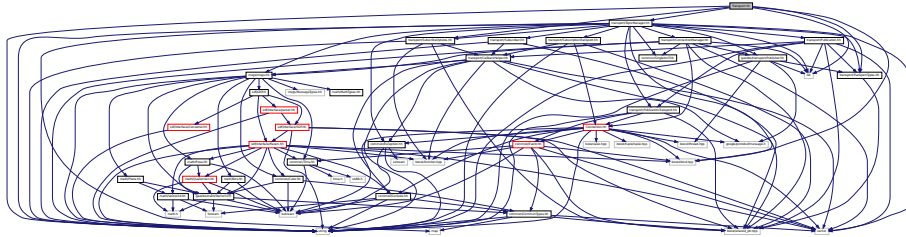
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**

11.155 Transport.hh File Reference

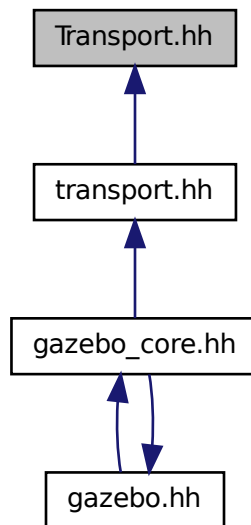
```

#include <boost/bind.hpp>
#include <string>
#include <list>
#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.

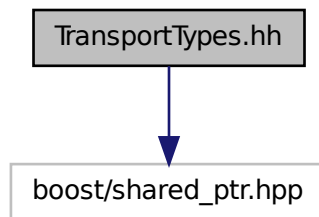
- bool **gazebo::transport::get_master_uri** (std::string &_master_host, unsigned int &_master_port)
Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.
- void **gazebo::transport::get_topic_namespaces** (std::list< std::string > &_namespaces)
Return all the namespace (world names) on the master.
- 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 unpause incoming messages.
- msgs::Response **gazebo::transport::request** (const std::string &_worldName, const msgs::Request &_request)
Send a request and receive a response.
- void **gazebo::transport::run** ()
Run the transport component.
- void **gazebo::transport::stop** ()
Stop the transport component from running.

11.156 TransportTypes.hh File Reference

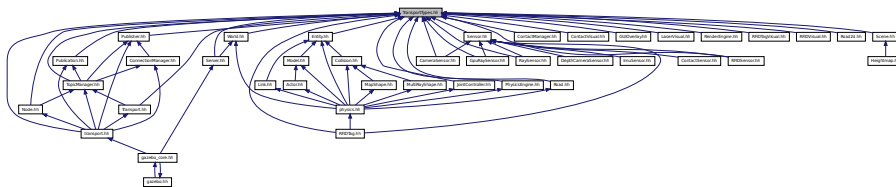
Forward declarations for transport.

```
#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**

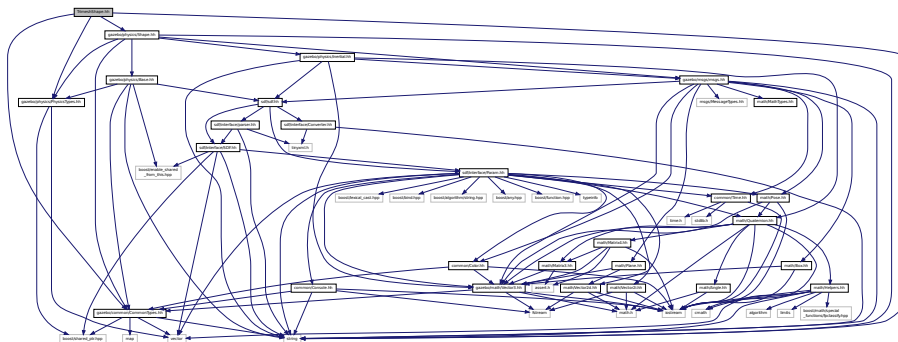
11.156.1 Detailed Description

Forward declarations for transport.

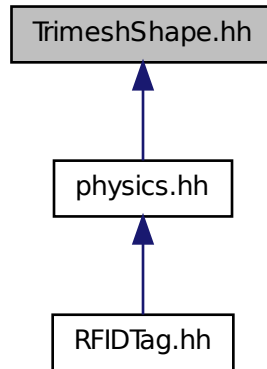
11.157 TrimeshShape.hh File Reference

```
#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.

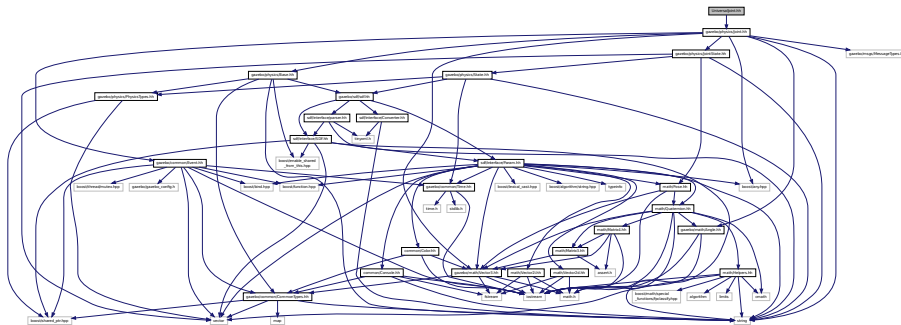
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

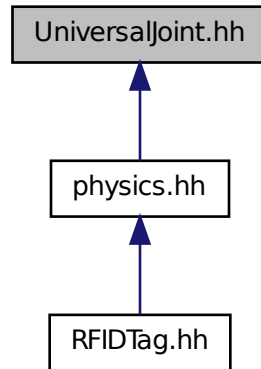
11.158 UniversalJoint.hh File Reference

```
#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 universal joint.

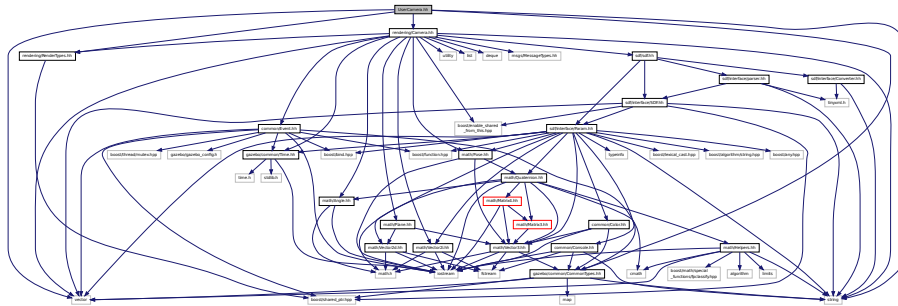
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.159 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 camera used for user visualization of a scene.

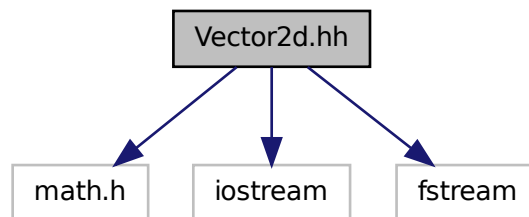
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

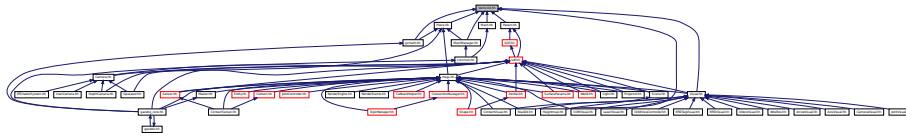
11.160 Vector2d.hh File Reference

```
#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.

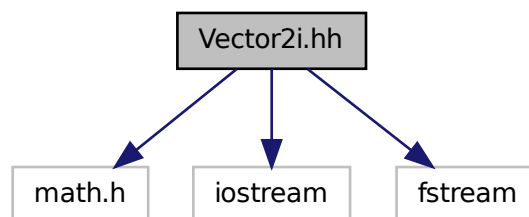
- namespace **gazebo::math**

Math namespace.

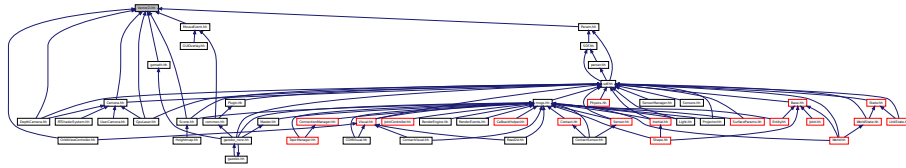
11.161 Vector2i.hh File Reference

```
#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.

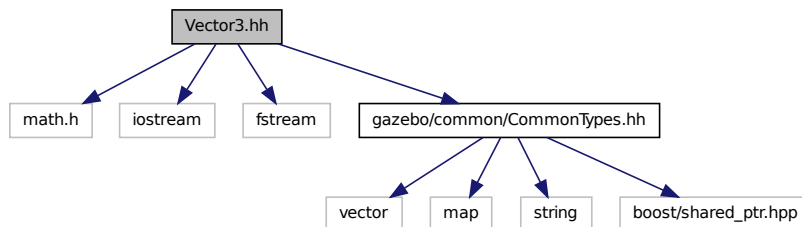
- namespace **gazebo::math**

Math namespace.

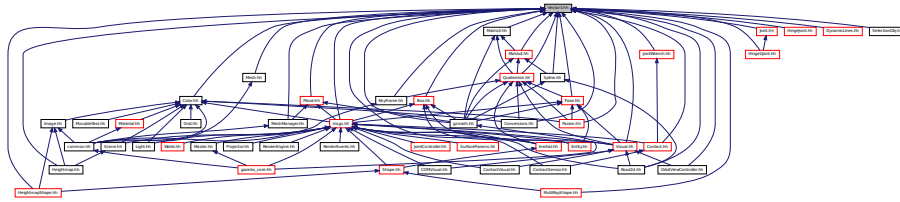
11.162 Vector3.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
#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. 799) class represents the generic vector containing 3 elements.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

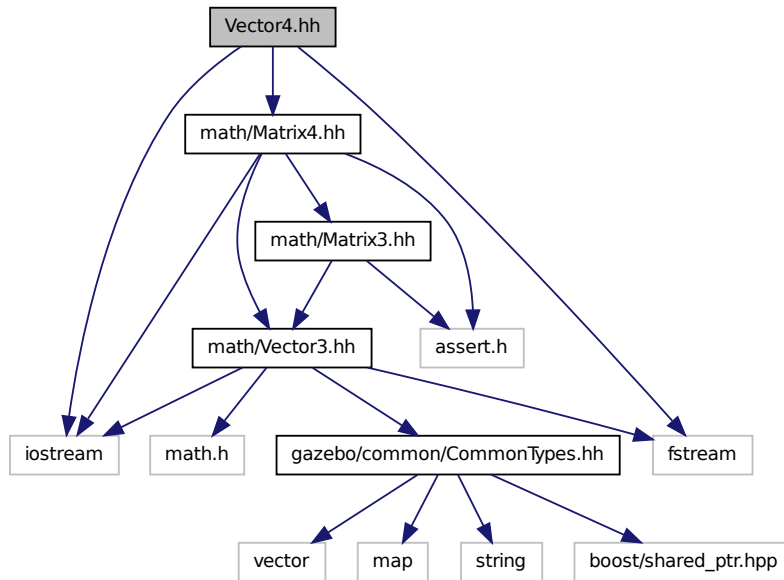
- namespace **gazebo::math**

Math namespace.

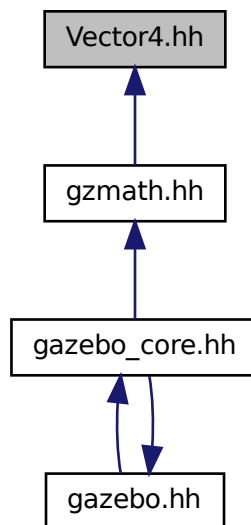
11.163 Vector4.hh File Reference

```
#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.

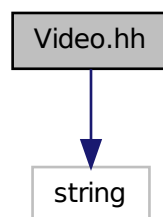
- namespace **gazebo::math**

Math namespace.

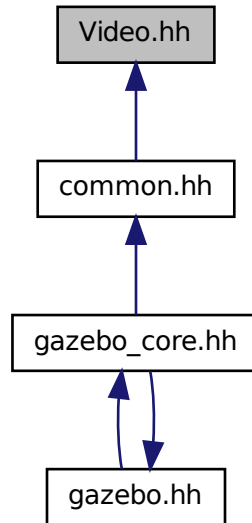
11.164 Video.hh File Reference

```
#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.

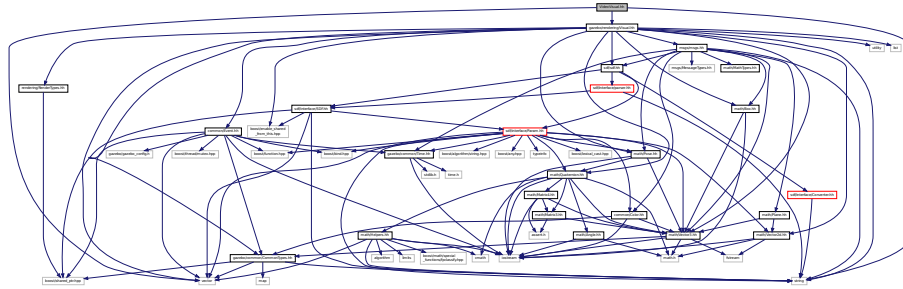
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.165 VideoVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for VideoVisual.hh:



Classes

- class **gazebo::rendering::VideoVisual**
A visual element that displays a video as a texture.

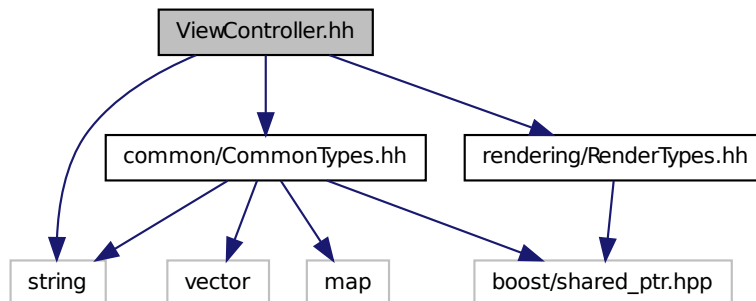
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::rendering**
Rendering namespace.

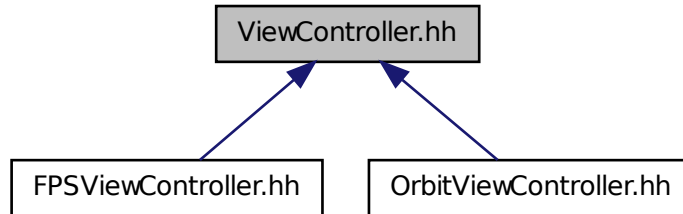
11.166 ViewController.hh File Reference

```
#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.

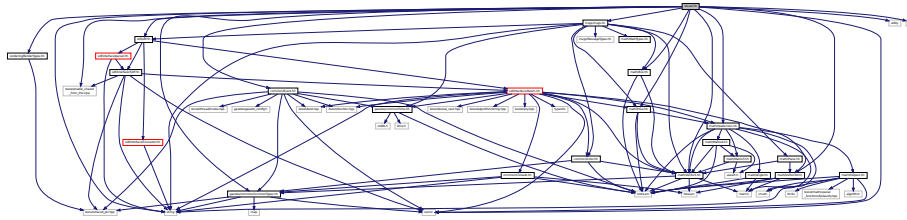
- namespace **gazebo::rendering**

Rendering namespace.

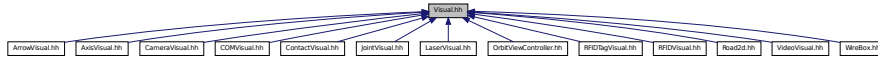
11.167 Visual.hh File Reference

```
#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 renderable object.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

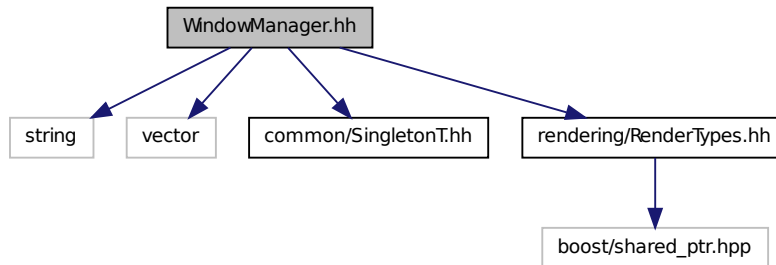
Rendering namespace.

- namespace **Ogre**

11.168 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**

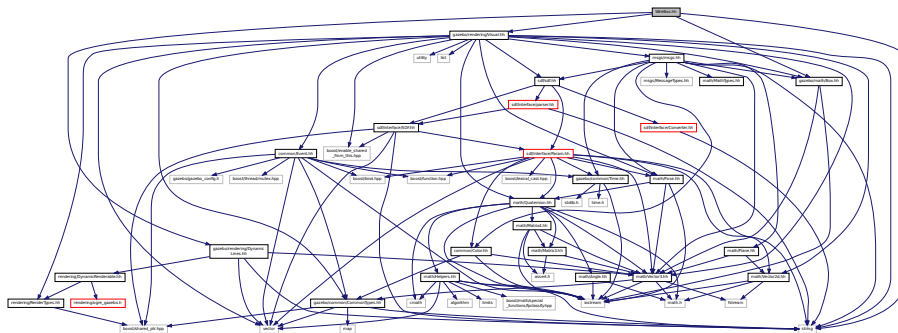
11.169 WireBox.hh File Reference

```

#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.

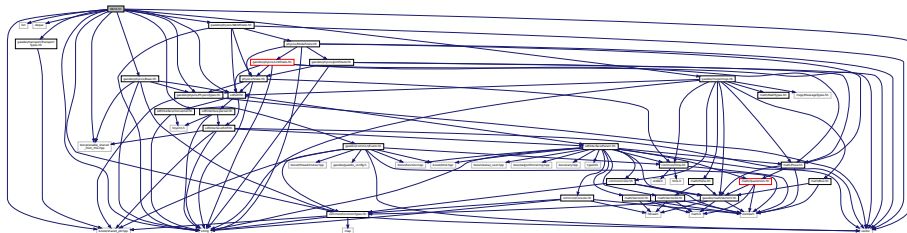
- namespace **gazebo::rendering**

Rendering namespace.

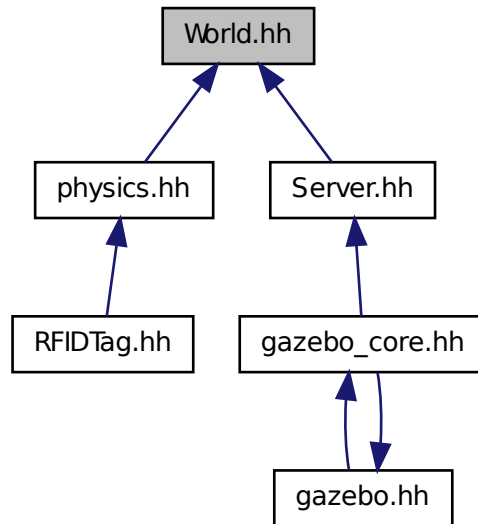
11.170 World.hh File Reference

```
#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.

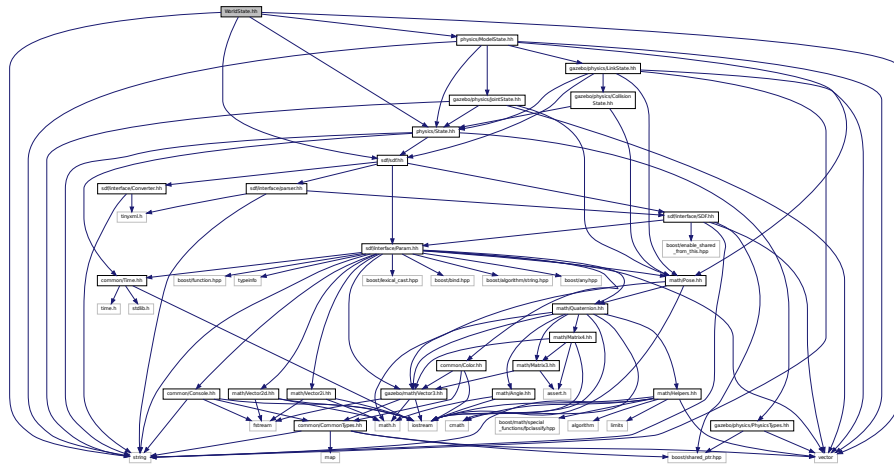
- namespace **gazebo::physics**

namespace for physics

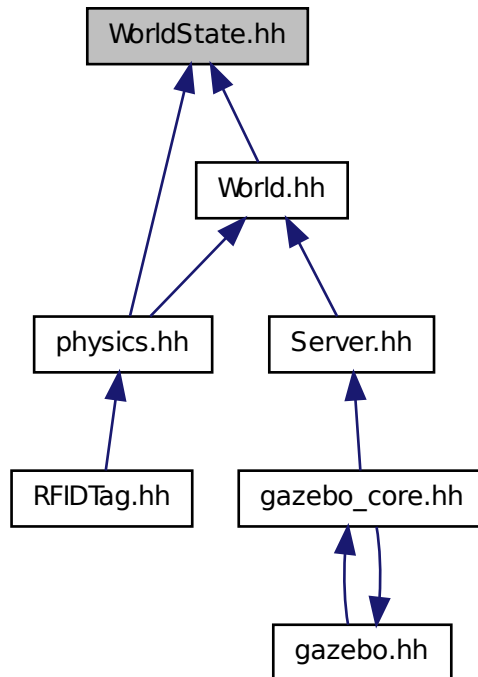
11.171 WorldState.hh File Reference

```
#include <string>
#include <vector>
#include "sdf/sdf.hh"
#include "physics/State.hh"
#include "physics/ModelState.hh"
```

Include dependency graph for WorldState.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::WorldState**
*Store state information of a **physics::World** (p. 853) object.*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Index

- ~Actor
 - gazebo::physics::Actor, 104
- ~Angle
 - gazebo::math::Angle, 109
- ~Animation
 - gazebo::common::Animation, 116
- ~ArrowVisual
 - gazebo::rendering::ArrowVisual, 120
- ~AxisVisual
 - gazebo::rendering::AxisVisual, 122
- ~BVHLoader
 - gazebo::common::BVHLoader, 144
- ~BallJoint
 - gazebo::physics::BallJoint, 124
- ~Base
 - gazebo::physics::Base, 129
- ~Box
 - gazebo::math::Box, 137
- ~BoxShape
 - gazebo::physics::BoxShape, 142
- ~COMVisual
 - gazebo::rendering::COMVisual, 206
- ~CallbackHelper
 - gazebo::transport::CallbackHelper, 145
- ~Camera
 - gazebo::rendering::Camera, 155
- ~CameraSensor
 - gazebo::sensors::CameraSensor, 175
- ~CameraVisual
 - gazebo::rendering::CameraVisual, 179
- ~ColladaLoader
 - gazebo::common::ColladaLoader, 180
- ~Collision
 - gazebo::physics::Collision, 183
- ~CollisionState
 - gazebo::physics::CollisionState, 191
- ~Color
 - gazebo::common::Color, 196
- ~Connection
 - gazebo::event::Connection, 207
 - gazebo::transport::Connection, 210
- ~Contact
 - gazebo::physics::Contact, 220
- ~ContactManager
 - gazebo::physics::ContactManager, 223
- ~ContactSensor
 - gazebo::sensors::ContactSensor, 226
- ~ContactVisual
 - gazebo::rendering::ContactVisual, 230
- ~CylinderShape
 - gazebo::physics::CylinderShape, 234
- ~DepthCamera
 - gazebo::rendering::DepthCamera, 240
- ~DepthCameraSensor
 - gazebo::sensors::DepthCameraSensor, 244
- ~DiagnosticTimer
 - gazebo::common::DiagnosticTimer, 250
- ~DynamicLines
 - gazebo::rendering::DynamicLines, 252
- ~DynamicRenderable
 - gazebo::rendering::DynamicRenderable, 256
- ~Element
 - sdf::Element, 261
- ~Entity
 - gazebo::physics::Entity, 269
- ~Event
 - gazebo::event::Event, 279
- ~EventT
 - Events, 36
- ~Exception
 - gazebo::common::Exception, 304
- ~FPSViewController
 - gazebo::rendering::FPSViewController, 306
- ~GUIOverlay
 - gazebo::rendering::GUIOverlay, 336
- ~GazeboGenerator
 - google::protobuf::compiler::cpp::GazeboGenerator, 312
- ~GpuLaser
 - gazebo::rendering::GpuLaser, 314
- ~GpuRaySensor
 - gazebo::sensors::GpuRaySensor, 320
- ~Grid
 - gazebo::rendering::Grid, 332
- ~Gripper
 - gazebo::physics::Gripper, 335
- ~Heightmap
 - gazebo::rendering::Heightmap, 340
- ~HeightmapShape
 - gazebo::physics::HeightmapShape, 343

- ~Hinge2Joint
 - gazebo::physics::Hinge2Joint, 346
- ~HingeJoint
 - gazebo::physics::HingeJoint, 348
- ~IOManager
 - gazebo::transport::IOManager, 365
- ~Image
 - gazebo::common::Image, 351
- ~ImuSensor
 - gazebo::sensors::ImuSensor, 355
- ~Inertial
 - gazebo::physics::Inertial, 359
- ~Joint
 - gazebo::physics::Joint, 370
- ~JointState
 - gazebo::physics::JointState, 383
- ~JointVisual
 - gazebo::rendering::JointVisual, 387
- ~KeyFrame
 - gazebo::common::KeyFrame, 389
- ~LaserVisual
 - gazebo::rendering::LaserVisual, 391
- ~Light
 - gazebo::rendering::Light, 393
- ~Link
 - gazebo::physics::Link, 402
- ~LinkState
 - gazebo::physics::LinkState, 418
- ~Master
 - gazebo::Master, 427
- ~Material
 - gazebo::common::Material, 431
- ~Matrix3
 - gazebo::math::Matrix3, 439
- ~Matrix4
 - gazebo::math::Matrix4, 443
- ~Mesh
 - gazebo::common::Mesh, 450
- ~MeshLoader
 - gazebo::common::MeshLoader, 455
- ~Model
 - gazebo::physics::Model, 464
- ~ModelPlugin
 - gazebo::ModelPlugin, 476
- ~ModelState
 - gazebo::physics::ModelState, 479
- ~MovableText
 - gazebo::rendering::MovableText, 488
- ~MultiRayShape
 - gazebo::physics::MultiRayShape, 494
- ~Node
 - gazebo::transport::Node, 500
- ~NodeAnimation
 - gazebo::common::NodeAnimation, 505
- ~NodeTransform
 - gazebo::common::NodeTransform, 511
- ~NumericAnimation
 - gazebo::common::NumericAnimation, 515
- ~NumericKeyFrame
 - gazebo::common::NumericKeyFrame, 516
- ~OrbitViewController
 - gazebo::rendering::OrbitViewController, 519
- ~PID
 - gazebo::common::PID, 544
- ~Param
 - sdf::Param, 524
- ~ParamT
 - sdf::ParamT, 529
- ~PhysicsEngine
 - gazebo::physics::PhysicsEngine, 533
- ~Plane
 - gazebo::math::Plane, 548
- ~PlaneShape
 - gazebo::physics::PlaneShape, 551
- ~Pose
 - gazebo::math::Pose, 559
- ~PoseAnimation
 - gazebo::common::PoseAnimation, 566
- ~PoseKeyFrame
 - gazebo::common::PoseKeyFrame, 568
- ~Projector
 - gazebo::rendering::Projector, 570
- ~Publication
 - gazebo::transport::Publication, 572
- ~PublicationTransport
 - gazebo::transport::PublicationTransport, 577
- ~Publisher
 - gazebo::transport::Publisher, 579
- ~Quaternion
 - gazebo::math::Quaternion, 585
- ~RFIDSensor
 - gazebo::sensors::RFIDSensor, 615
- ~RFIDTag
 - gazebo::sensors::RFIDTag, 617
- ~RFIDTagVisual
 - gazebo::rendering::RFIDTagVisual, 620
- ~RFIDVisual
 - gazebo::rendering::RFIDVisual, 621
- ~RaySensor
 - gazebo::sensors::RaySensor, 598
- ~RayShape
 - gazebo::physics::RayShape, 606
- ~Road
 - gazebo::physics::Road, 622
- ~Road2d
 - gazebo::rendering::Road2d, 624
- ~RotationSpline
 - gazebo::math::RotationSpline, 625

- ~STLLoader
 - gazebo::common::STLLoader, 706
- ~Scene
 - gazebo::rendering::Scene, 635
- ~ScrewJoint
 - gazebo::physics::ScrewJoint, 647
- ~SelectionObj
 - gazebo::rendering::SelectionObj, 651
- ~Sensor
 - gazebo::sensors::Sensor, 655
- ~SensorPlugin
 - gazebo::SensorPlugin, 666
- ~Server
 - gazebo::Server, 667
- ~Shape
 - gazebo::physics::Shape, 669
- ~SingletonT
 - SingletonT, 672
- ~Skeleton
 - gazebo::common::Skeleton, 674
- ~SkeletonAnimation
 - gazebo::common::SkeletonAnimation, 680
- ~SkeletonNode
 - gazebo::common::SkeletonNode, 686
- ~SliderJoint
 - gazebo::physics::SliderJoint, 693
- ~SphereShape
 - gazebo::physics::SphereShape, 696
- ~Spline
 - gazebo::math::Spline, 698
- ~State
 - gazebo::physics::State, 703
- ~SubMesh
 - gazebo::common::SubMesh, 709
- ~Subscriber
 - gazebo::transport::Subscriber, 718
- ~SubscriptionTransport
 - gazebo::transport::SubscriptionTransport, 720
- ~SurfaceParams
 - gazebo::physics::SurfaceParams, 722
- ~SystemPlugin
 - gazebo::SystemPlugin, 731
- ~Time
 - gazebo::common::Time, 736
- ~Timer
 - gazebo::common::Timer, 753
- ~TrimeshShape
 - gazebo::physics::TrimeshShape, 763
- ~URDF2Gazebo
 - urdf2gazebo::URDF2Gazebo, 769
- ~UniversalJoint
 - gazebo::physics::UniversalJoint, 766
- ~UserCamera
 - gazebo::rendering::UserCamera, 777
- ~Vector2d
 - gazebo::math::Vector2d, 784
- ~Vector2i
 - gazebo::math::Vector2i, 793
- ~Vector3
 - gazebo::math::Vector3, 802
- ~Vector4
 - gazebo::math::Vector4, 814
- ~Video
 - gazebo::common::Video, 822
- ~VideoVisual
 - gazebo::rendering::VideoVisual, 824
- ~ViewController
 - gazebo::rendering::ViewController, 826
- ~Visual
 - gazebo::rendering::Visual, 834
- ~WireBox
 - gazebo::rendering::WireBox, 852
- ~World
 - gazebo::physics::World, 856
- ~WorldPlugin
 - gazebo::WorldPlugin, 865
- ~WorldState
 - gazebo::physics::WorldState, 867
- _setupGeometry
 - gazebo::rendering::MovableText, 488
- _updateColors
 - gazebo::rendering::MovableText, 488
- a
 - gazebo::common::Color, 203
- ABGR
 - gazebo::common::Color, 196
- ACTOR
 - gazebo::physics::Base, 128
- ADD
 - gazebo::common::Material, 430
- ARGB
 - gazebo::common::Color, 196
- AcceptCallback
 - gazebo::transport::Connection, 209
- active
 - gazebo::physics::Actor, 105
 - gazebo::sensors::Sensor, 659
- Actor
 - gazebo::physics::Actor, 103
- Actor.hh, 871
- Actor_V
 - gazebo::physics, 88
- ActorPtr
 - gazebo::physics, 88
- Add
 - gazebo::common::LogRecord, 425
- add_plugin

- gazebo, 77
- add_search_path_suffix
 - Common, 32
- AddAnimation
 - gazebo::common::Skeleton, 674
- AddAttribute
 - sdf::Element, 261
- AddCallback
 - gazebo::transport::PublicationTransport, 577
- AddChild
 - gazebo::common::SkeletonNode, 686
 - gazebo::physics::Base, 129
- AddChildJoint
 - gazebo::physics::Link, 402
- AddContact
 - gazebo::physics::Collision, 183
- AddElement
 - sdf::Element, 261
- AddElementDescription
 - sdf::Element, 261
- addEntity
 - gazebo::event::Events, 292
- AddForce
 - gazebo::physics::Link, 403
- AddForceAtRelativePosition
 - gazebo::physics::Link, 403
- AddForceAtWorldPosition
 - gazebo::physics::Link, 403
- AddGazeboPaths
 - gazebo::common::SystemPaths, 727
- AddIndex
 - gazebo::common::SubMesh, 709
- AddJoint
 - gazebo::physics::JointController, 380
- AddKeyFrame
 - gazebo::common::NodeAnimation, 505
 - gazebo::common::SkeletonAnimation, 680
- addKeyValue
 - urdf2gazebo::URDF2Gazebo, 769
- AddMaterial
 - gazebo::common::Mesh, 450
- AddMesh
 - gazebo::common::MeshManager, 457
- addNestedModel
 - sdf, 99
- AddNode
 - gazebo::transport::TopicManager, 756
- AddNodeAssignment
 - gazebo::common::SubMesh, 710
- AddNormal
 - gazebo::common::SubMesh, 710
- AddOgrePaths
 - gazebo::common::SystemPaths, 727
- AddParentJoint
 - gazebo::physics::Link, 403
- AddPluginPaths
 - gazebo::common::SystemPaths, 727
- AddPoint
 - gazebo::math::RotationSpline, 625
 - gazebo::math::Spline, 699
 - gazebo::rendering::DynamicLines, 252
- AddPublisher
 - gazebo::transport::Publication, 573
- AddRawTransform
 - gazebo::common::SkeletonNode, 686
- AddRay
 - gazebo::physics::MultiRayShape, 494
- AddRelativeForce
 - gazebo::physics::Link, 403
- AddRelativeTorque
 - gazebo::physics::Link, 403
- AddResourcePath
 - gazebo::rendering::RenderEngine, 611
- AddScene
 - gazebo::rendering::RTShaderSystem, 630
- AddSearchPathSuffix
 - gazebo::common::SystemPaths, 728
- AddSubMesh
 - gazebo::common::Mesh, 450
- AddSubscription
 - gazebo::transport::Publication, 573
- AddTag
 - gazebo::sensors::RFIDSensor, 615
- AddTexCoord
 - gazebo::common::SubMesh, 710
- AddTime
 - gazebo::common::Animation, 117
- AddTorque
 - gazebo::physics::Link, 404
- addTransform
 - urdf2gazebo::URDF2Gazebo, 769
- AddTransport
 - gazebo::transport::Publication, 573
- AddType
 - gazebo::physics::Base, 129
- AddValue
 - sdf::Element, 261
- AddVertNodeWeight
 - gazebo::common::Skeleton, 675
- AddVertex
 - gazebo::common::SubMesh, 710
- AddVisual
 - gazebo::rendering::Scene, 635
- Advertise
 - gazebo::transport::ConnectionManager, 215
 - gazebo::transport::Node, 501
 - gazebo::transport::TopicManager, 756
- alt

- gazebo::common::MouseEvent, 484
- ambient
 - gazebo::common::Material, 436
- anchorLink
 - gazebo::physics::Joint, 379
- anchorPos
 - gazebo::physics::Joint, 379
- Angle
 - gazebo::math::Angle, 109
- Angle.hh, 872
 - GZ_DTOR, 874
 - GZ_NORMALIZE, 874
 - GZ_RTOD, 874
- angularAccel
 - gazebo::physics::Link, 415
- animState
 - gazebo::rendering::Camera, 171
- Animation
 - gazebo::common::Animation, 116
- animation
 - gazebo::physics::Entity, 276
- Animation.hh, 875
- AnimationComplete
 - gazebo::rendering::Camera, 155
 - gazebo::rendering::UserCamera, 777
- animationConnection
 - gazebo::physics::Entity, 276
- AnimationPtr
 - gazebo::common, 80
- animationStartPose
 - gazebo::physics::Entity, 276
- animations
 - gazebo::common::SkeletonAnimation, 682
- anims
 - gazebo::common::Skeleton, 678
- ApplyShadows
 - gazebo::rendering::RTShaderSystem, 630
- AreConnected
 - gazebo::physics::Joint, 370
- ArrowVisual
 - gazebo::rendering::ArrowVisual, 120
- ArrowVisual.hh, 876
- ArrowVisualPtr
 - gazebo::rendering, 92
- AsyncRead
 - gazebo::transport::Connection, 210
- Attach
 - gazebo::physics::Joint, 370
 - gazebo::rendering::SelectionObj, 651
- AttachAxes
 - gazebo::rendering::Visual, 834
- AttachCameraToImage
 - gazebo::rendering::GUIOverlay, 336, 337
- AttachEntity
 - gazebo::rendering::RTShaderSystem, 630
- AttachLineVertex
 - gazebo::rendering::Visual, 834
- AttachMesh
 - gazebo::rendering::Visual, 834
- AttachObject
 - gazebo::rendering::Visual, 834
- AttachStaticModel
 - gazebo::physics::Link, 404
 - gazebo::physics::Model, 464
- AttachToVisual
 - gazebo::rendering::Camera, 155
- AttachToVisualImpl
 - gazebo::rendering::Camera, 155, 156
 - gazebo::rendering::UserCamera, 777
- AttachViewport
 - gazebo::rendering::RTShaderSystem, 630
- AttachVisual
 - gazebo::rendering::Visual, 834
- attachedModels
 - gazebo::physics::Model, 474
- attachedModelsOffset
 - gazebo::physics::Link, 415
 - gazebo::physics::Model, 474
- Attribute
 - gazebo::physics::Joint, 369
- autoCalc
 - gazebo::math::RotationSpline, 628
 - gazebo::math::Spline, 701
- autoStart
 - gazebo::physics::Actor, 105
- AxisVisual
 - gazebo::rendering::AxisVisual, 122
- AxisVisual.hh, 877
- AxisVisualPtr
 - gazebo::rendering, 92
- b
 - gazebo::common::Color, 203
- BALL_JOINT
 - gazebo::physics::Base, 128
- BASE
 - gazebo::physics::Base, 128
- BAYER_GBRG8
 - gazebo::common::Image, 351
- BAYER_GRBG8
 - gazebo::common::Image, 351
- BAYER_RGGB8
 - gazebo::common::Image, 350
- BAYER_RGGR8
 - gazebo::common::Image, 351
- BGR_INT16
 - gazebo::common::Image, 350
- BGR_INT32

- gazebo::common::Image, 350
- BGR_INT8
 - gazebo::common::Image, 350
- BGRA
 - gazebo::common::Color, 196
- BGRA_INT8
 - gazebo::common::Image, 350
- BLEND_COUNT
 - gazebo::common::Material, 430
- BLINN
 - gazebo::common::Material, 431
- BOX_SHAPE
 - gazebo::physics::Base, 128
- BVHLoader
 - gazebo::common::BVHLoader, 144
- BVHLoader.hh, 882
 - X_POSITION, 883
 - X_ROTATION, 883
 - Y_POSITION, 883
 - Y_ROTATION, 883
 - Z_POSITION, 883
 - Z_ROTATION, 883
- BallJoint
 - gazebo::physics::BallJoint, 124
- BallJoint.hh, 877
- Base
 - gazebo::physics::Base, 128
- Base.hh, 878
- Base_V
 - gazebo::physics, 88
- BasePtr
 - gazebo::physics, 88
- bayerFrameBuffer
 - gazebo::rendering::Camera, 171
- bindShapeTransform
 - gazebo::common::Skeleton, 678
- Black
 - gazebo::common::Color, 203
- BlendMode
 - gazebo::common::Material, 430
- blendMode
 - gazebo::common::Material, 436
- BlendModeStr
 - gazebo::common::Material, 436
- blobs
 - urdf2gazebo::GazeboExtension, 309
- Blue
 - gazebo::common::Color, 203
- body1Force
 - gazebo::physics::JointWrench, 388
- body1Torque
 - gazebo::physics::JointWrench, 388
- body2Force
 - gazebo::physics::JointWrench, 388
- body2Torque
 - gazebo::physics::JointWrench, 388
- bonePosePub
 - gazebo::physics::Actor, 105
- boost, 75
- bounce
 - gazebo::physics::SurfaceParams, 723
- bounceThreshold
 - gazebo::physics::SurfaceParams, 723
- Box
 - gazebo::math::Box, 137
- Box.hh, 879
- BoxShape
 - gazebo::physics::BoxShape, 142
- BoxShape.hh, 880
- BoxShapePtr
 - gazebo::physics, 88
- build
 - gazebo::common::Animation, 118
- BuildInterpolationSplines
 - gazebo::common::PoseAnimation, 566
- BuildNodeMap
 - gazebo::common::Skeleton, 675
- button
 - gazebo::common::MouseEvent, 484
- ButtonCallback
 - gazebo::rendering::GUIOverlay, 337
- Buttons
 - gazebo::common::MouseEvent, 484
- buttons
 - gazebo::common::MouseEvent, 485
- CFM
 - gazebo::physics::Joint, 369
- COLLISION
 - gazebo::physics::Base, 128
- COMVisual
 - gazebo::rendering::COMVisual, 205
- COMVisual.hh, 896
- COMVisualPtr
 - gazebo::rendering, 92
- COR3_MAX
 - STLLoader.hh, 1039
- CYLINDER_SHAPE
 - gazebo::physics::Base, 128
- CallbackHelper
 - gazebo::transport::CallbackHelper, 145
- CallbackHelper.hh, 883
- CallbackHelperPtr
 - Transport, 71
- CallbackHelperT
 - gazebo::transport::CallbackHelperT, 148
- Camera
 - gazebo::rendering::Camera, 155

- camera
 - gazebo::rendering::Camera, 171
 - gazebo::rendering::ViewController, 828
- Camera.hh, 885
- cameraCount
 - gazebo::sensors::GpuRaySensor, 328
- cameraElem
 - gazebo::sensors::GpuRaySensor, 328
- CameraPtr
 - gazebo::rendering, 92
- CameraSensor
 - gazebo::sensors::CameraSensor, 175
- CameraSensor.hh, 886
- CameraSensor_V
 - gazebo::sensors, 95
- CameraSensorPtr
 - gazebo::sensors, 95
- CameraVisual
 - gazebo::rendering::CameraVisual, 178
- CameraVisual.hh, 887
- CameraVisualPtr
 - gazebo::rendering, 92
- Cancel
 - gazebo::transport::Connection, 210
- captureData
 - gazebo::rendering::Camera, 171
- cegui.h, 887
- cfm
 - gazebo::physics::SurfaceParams, 723
- cgVisuals
 - gazebo::physics::Link, 415
- chfov
 - gazebo::sensors::GpuRaySensor, 329
- childLink
 - gazebo::physics::Joint, 379
- children
 - gazebo::common::SkeletonNode, 691
 - gazebo::physics::Base, 135
- childrenEnd
 - gazebo::physics::Base, 135
- clamp
 - Math, 48
- Classes for physics and dynamics, 39
 - create_world, 42
 - EntityTypename, 44
 - fini, 42
 - GZ_REGISTER_PHYSICS_ENGINE, 42
 - get_world, 42
 - init_world, 43
 - init_worlds, 43
 - load, 43
 - load_world, 43
 - load_worlds, 43
 - pause_world, 43
 - pause_worlds, 44
 - PhysicsFactoryFn, 42
 - remove_worlds, 44
 - run_world, 44
 - run_worlds, 44
 - stop_world, 44
 - stop_worlds, 44
- Clear
 - gazebo::math::RotationSpline, 625
 - gazebo::math::Spline, 699
 - gazebo::physics::ContactManager, 223
 - gazebo::physics::World, 856
 - gazebo::rendering::DynamicLines, 253
 - gazebo::rendering::RTShaderSystem, 631
 - gazebo::rendering::Scene, 636
 - gazebo::rendering::SelectionObj, 651
 - sdf::Plugin, 553
- clear_buffers
 - Transport, 71
- ClearBuffers
 - gazebo::transport::TopicManager, 756
- ClearElements
 - sdf::Element, 261
- ClearGazeboPaths
 - gazebo::common::SystemPaths, 728
- ClearOgrePaths
 - gazebo::common::SystemPaths, 728
- ClearParent
 - gazebo::rendering::Visual, 835
- ClearPluginPaths
 - gazebo::common::SystemPaths, 728
- Clone
 - gazebo::physics::Contact, 220
 - gazebo::rendering::Visual, 835
 - sdf::Element, 261
 - sdf::Param, 524
 - sdf::ParamT, 529
- CloneVisual
 - gazebo::rendering::Scene, 636
- coeffs
 - gazebo::math::Spline, 701
- ColladaLoader
 - gazebo::common::ColladaLoader, 180
- ColladaLoader.hh, 888
- Collision
 - gazebo::physics::Collision, 183
- Collision.hh, 890
- collision1
 - gazebo::physics::Contact, 221
- collision2
 - gazebo::physics::Contact, 221
- Collision_V
 - gazebo::physics, 88
- collisionParent

- gazebo::physics::Shape, 670
- CollisionPtr
 - gazebo::physics, 88
 - Gazebo_parser, 65
- CollisionState
 - gazebo::physics::CollisionState, 191
- CollisionState.hh, 891
- Color
 - gazebo::common::Color, 196
- Color.hh, 892
- ColorErr
 - Common, 32
- ColorMsg
 - Common, 32
- Common, 27
 - add_search_path_suffix, 32
 - ColorErr, 32
 - ColorMsg, 32
 - DIAG_TIMER, 30
 - DiagnosticTimerPtr, 32
 - DownloadDependencies, 33
 - find_file, 33
 - find_file_path, 33
 - GetManifest, 33
 - GetModelFile, 33
 - GetModelName, 34
 - GetModelPath, 34
 - GetModels, 34
 - GetURI, 35
 - gzclr_end, 30
 - gzclr_start, 31
 - gzdbg, 31
 - gzerr, 31
 - gzmsg, 31
 - gzthrow, 31
 - gzwarn, 31
 - HasModel, 35
 - Instance, 35
 - Load, 35
 - MODEL_PLUGIN, 32
 - NullStream, 32
 - PluginType, 32
 - SENSOR_PLUGIN, 32
 - SYSTEM_PLUGIN, 32
 - SetQuiet, 35
 - VISUAL_PLUGIN, 32
 - WORLD_PLUGIN, 32
- Common.hh, 893
- common/Plugin.hh
 - GZ_REGISTER_MODEL_PLUGIN, 990
 - GZ_REGISTER_SENSOR_PLUGIN, 990
 - GZ_REGISTER_SYSTEM_PLUGIN, 991
 - GZ_REGISTER_VISUAL_PLUGIN, 991
 - GZ_REGISTER_WORLD_PLUGIN, 991
- CommonTypes.hh, 895
 - GAZEBO_DEPRECATED, 896
 - GAZEBO_FORCEINLINE, 896
 - NULL, 896
- Connect
 - Events, 36
 - gazebo::transport::Connection, 210
- ConnectAddEntity
 - gazebo::event::Events, 285
- ConnectContact
 - gazebo::physics::Collision, 183
- ConnectCreateEntity
 - gazebo::event::Events, 285
- ConnectCreateScene
 - gazebo::rendering::Events, 280
- ConnectDeleteEntity
 - gazebo::event::Events, 285
- ConnectDiagTimerStart
 - gazebo::event::Events, 286
- ConnectDiagTimerStop
 - gazebo::event::Events, 286
- ConnectEnabled
 - gazebo::physics::Link, 404
- ConnectJointUpdate
 - gazebo::physics::Joint, 370
- ConnectNewDepthFrame
 - gazebo::rendering::DepthCamera, 240
- ConnectNewImageFrame
 - gazebo::rendering::Camera, 156
- ConnectNewLaserFrame
 - gazebo::rendering::GpuLaser, 314
 - gazebo::sensors::GpuRaySensor, 320
- ConnectNewLaserScans
 - gazebo::physics::MultiRayShape, 494
- ConnectNewRGBPointCloud
 - gazebo::rendering::DepthCamera, 240
- ConnectPause
 - gazebo::event::Events, 286
- ConnectPostRender
 - gazebo::event::Events, 287
- ConnectPreRender
 - gazebo::event::Events, 287
- ConnectPubToSub
 - gazebo::transport::TopicManager, 757
- ConnectRemoveScene
 - gazebo::rendering::Events, 281
- ConnectRender
 - gazebo::event::Events, 287
- ConnectSetSelectedEntity
 - gazebo::event::Events, 288
- ConnectStep
 - gazebo::event::Events, 288
- ConnectStop
 - gazebo::event::Events, 288

- ConnectSubToPub
 - gazebo::transport::TopicManager, 757
- ConnectSubscribers
 - gazebo::transport::TopicManager, 757
- ConnectToRemoteHost
 - gazebo::transport::ConnectionManager, 215
- ConnectToShutdown
 - gazebo::transport::Connection, 210
- ConnectViewContacts
 - gazebo::rendering::Events, 281
- ConnectWorldCreated
 - gazebo::event::Events, 288
- ConnectWorldUpdateEnd
 - gazebo::event::Events, 289
- ConnectWorldUpdateStart
 - gazebo::event::Events, 289
- Connection
 - gazebo::event::Connection, 207
 - gazebo::transport::Connection, 210
- Connection.hh, 897
 - HEADER_LENGTH, 899
- Connection_V
 - gazebo::event, 81
- ConnectionCount
 - Events, 37
- ConnectionManager.hh, 899
- ConnectionPtr
 - gazebo::event, 81
 - gazebo::transport, 97
- connections
 - gazebo::physics::Entity, 276
 - gazebo::rendering::Camera, 171
 - gazebo::sensors::Sensor, 659
- Console.hh, 900
- Contact
 - gazebo::physics::Contact, 220
- Contact.hh, 902
 - MAX_COLLIDE_RETURNS, 903
 - MAX_CONTACT_JOINTS, 903
- contactFiducial
 - gazebo::physics::RayShape, 608
- contactLen
 - gazebo::physics::RayShape, 608
- ContactManager
 - gazebo::physics::ContactManager, 223
- contactManager
 - gazebo::physics::PhysicsEngine, 541
- ContactManager.hh, 903
- ContactPtr
 - gazebo::physics, 88
- contactRetro
 - gazebo::physics::RayShape, 609
- ContactSensor
 - gazebo::sensors::ContactSensor, 226
- ContactSensor.hh, 904
- ContactSensor_V
 - gazebo::sensors, 95
- ContactSensorPtr
 - gazebo::sensors, 95
- ContactVisual
 - gazebo::rendering::ContactVisual, 229
- ContactVisual.hh, 905
- ContactVisualPtr
 - gazebo::rendering, 92
- control
 - gazebo::common::MouseEvent, 485
- Conversions.hh, 906
- Convert
 - gazebo::rendering::Conversions, 231, 232
 - Messages, 53–56
 - sdf::Converter, 232
- Converter.hh, 906
- CoordPoseSolve
 - gazebo::math::Pose, 559
- CoordPositionAdd
 - gazebo::math::Pose, 559
- CoordPositionSub
 - gazebo::math::Pose, 559
- CoordRotationAdd
 - gazebo::math::Pose, 560
- CoordRotationSub
 - gazebo::math::Pose, 560
- Copy
 - sdf::Element, 261
- copyChildren
 - sdf, 99
- CopyNormals
 - gazebo::common::SubMesh, 711
- copyPose
 - urdf2gazebo::URDF2Gazebo, 769
- CopyVertices
 - gazebo::common::SubMesh, 711
- Correct
 - gazebo::math::Pose, 560
 - gazebo::math::Quaternion, 585
 - gazebo::math::Vector3, 803
- count
 - gazebo::physics::Contact, 221
- Create
 - gazebo::PluginT, 555
- create_scene
 - Rendering, 63
- create_sensor
 - Sensors, 68
- create_world
 - Classes for physics and dynamics, 42
- CreateBox
 - gazebo::common::MeshManager, 457

- CreateCamera
 - gazebo::common::MeshManager, 457
 - gazebo::rendering::Scene, 636
- CreateCollision
 - gazebo::physics::PhysicsEngine, 534
- createCollision
 - urdf2gazebo::URDF2Gazebo, 770
- createCollisions
 - urdf2gazebo::URDF2Gazebo, 770
- CreateCone
 - gazebo::common::MeshManager, 457
- CreateCylinder
 - gazebo::common::MeshManager, 458
- CreateDepthCamera
 - gazebo::rendering::Scene, 636
- CreateDepthTexture
 - gazebo::rendering::DepthCamera, 240
- CreateDynamicLine
 - gazebo::rendering::Visual, 835
- createGeometry
 - urdf2gazebo::URDF2Gazebo, 770
- CreateGrid
 - gazebo::rendering::Scene, 637
- createInertial
 - urdf2gazebo::URDF2Gazebo, 770
- CreateJoint
 - gazebo::physics::PhysicsEngine, 534
- createJoint
 - urdf2gazebo::URDF2Gazebo, 770
- CreateKeyFrame
 - gazebo::common::NumericAnimation, 515
 - gazebo::common::PoseAnimation, 566
- CreateLaserTexture
 - gazebo::rendering::GpuLaser, 314
- CreateLink
 - gazebo::physics::PhysicsEngine, 534
- createLink
 - urdf2gazebo::URDF2Gazebo, 770
- CreatePlane
 - gazebo::common::MeshManager, 458
 - gazebo::physics::PlaneShape, 551
- CreateRenderTexture
 - gazebo::rendering::Camera, 156
- CreateRequest
 - Messages, 56
- createSDF
 - urdf2gazebo::URDF2Gazebo, 770
- CreateScene
 - gazebo::rendering::RenderEngine, 611
- createScene
 - gazebo::rendering::Events, 282
- CreateSensor
 - gazebo::sensors::SensorManager, 663
- CreateShape
 - gazebo::physics::PhysicsEngine, 534
- CreateSphere
 - gazebo::common::MeshManager, 458
- CreateTimer
 - gazebo::common::DiagnosticManager, 247
- CreateTube
 - gazebo::common::MeshManager, 459
- CreateUserCamera
 - gazebo::rendering::Scene, 637
- CreateVertexDeclaration
 - gazebo::rendering::DynamicLines, 253
 - gazebo::rendering::DynamicRenderable, 256
- createVisual
 - urdf2gazebo::URDF2Gazebo, 770
- createVisuals
 - urdf2gazebo::URDF2Gazebo, 770
- CreateWindow
 - gazebo::rendering::GUIOverlay, 337
 - gazebo::rendering::WindowManager, 850
- Cross
 - gazebo::math::Vector2d, 784
 - gazebo::math::Vector2i, 793
 - gazebo::math::Vector3, 803
- cvfov
 - gazebo::sensors::GpuRaySensor, 329
- CylinderShape
 - gazebo::physics::CylinderShape, 234
- CylinderShape.hh, 907
- CylinderShapePtr
 - gazebo::physics, 88
- d
 - gazebo::math::Plane, 549
- DEFERRED
 - gazebo::rendering::RenderEngine, 611
- DIAG_TIMER
 - Common, 30
- damping_coefficient
 - gazebo::physics::Joint, 379
- damping_factor
 - urdf2gazebo::GazeboExtension, 309
- data
 - sdf::Plugin, 554
- DebugCallbackHelper
 - gazebo::transport::DebugCallbackHelper, 237
- DebugPrint
 - gazebo::physics::PhysicsEngine, 535
- DebugString
 - gazebo::physics::Contact, 220
- DecCount
 - gazebo::transport::IOManager, 365
- DecodeTopicName
 - gazebo::transport::Node, 501
- defaultValue

- sdf::ParamT, 530
- Degree
 - gazebo::math::Angle, 109
- DeleteDynamicLine
 - gazebo::rendering::Visual, 835
- deleteEntity
 - gazebo::event::Events, 292
- DepthCamera
 - gazebo::rendering::DepthCamera, 239
- DepthCamera.hh, 908
- DepthCameraPtr
 - gazebo::rendering, 92
- DepthCameraSensor
 - gazebo::sensors::DepthCameraSensor, 244
- DepthCameraSensor.hh, 909
- DepthCameraSensor_V
 - gazebo::sensors, 95
- DepthCameraSensorPtr
 - gazebo::sensors, 95
- depthTarget
 - gazebo::rendering::DepthCamera, 242
- depthTexture
 - gazebo::rendering::DepthCamera, 242
- depthViewport
 - gazebo::rendering::DepthCamera, 242
- depths
 - gazebo::physics::Contact, 221
- description
 - sdf::Param, 527
- Detach
 - gazebo::physics::Joint, 371
- DetachAllStaticModels
 - gazebo::physics::Link, 404
- DetachEntity
 - gazebo::rendering::RTShaderSystem, 631
- DetachObjects
 - gazebo::rendering::Visual, 835
- DetachStaticModel
 - gazebo::physics::Link, 404
 - gazebo::physics::Model, 464
- DetachViewport
 - gazebo::rendering::RTShaderSystem, 631
- DetachVisual
 - gazebo::rendering::Visual, 835, 836
- diagTimerStart
 - gazebo::event::Events, 293
- diagTimerStop
 - gazebo::event::Events, 293
- DiagnosticTimer
 - gazebo::common::DiagnosticTimer, 249
- DiagnosticTimerPtr
 - Common, 32
- Diagnostics.hh, 910
- diffuse
 - gazebo::common::Material, 436
- dirtyPose
 - gazebo::physics::Entity, 276
- dirtyPoses
 - gazebo::physics::World, 864
- DisableAllModels
 - gazebo::physics::World, 856
- DisableTrackVisual
 - gazebo::rendering::Visual, 836
- Disconnect
 - Events, 37, 38
 - gazebo::event::Event, 279
- DisconnectAddEntity
 - gazebo::event::Events, 289
- DisconnectContact
 - gazebo::physics::Collision, 184
- DisconnectCreateEntity
 - gazebo::event::Events, 289
- DisconnectCreateScene
 - gazebo::rendering::Events, 281
- DisconnectDeleteEntity
 - gazebo::event::Events, 290
- DisconnectDiagTimerStart
 - gazebo::event::Events, 290
- DisconnectDiagTimerStop
 - gazebo::event::Events, 290
- DisconnectEnabled
 - gazebo::physics::Link, 405
- DisconnectJointUpdate
 - gazebo::physics::Joint, 371
- DisconnectNewDepthFrame
 - gazebo::rendering::DepthCamera, 240
- DisconnectNewImageFrame
 - gazebo::rendering::Camera, 156
- DisconnectNewLaserFrame
 - gazebo::rendering::GpuLaser, 315
 - gazebo::sensors::GpuRaySensor, 321
- DisconnectNewLaserScans
 - gazebo::physics::MultiRayShape, 495
- DisconnectNewRGBPointCloud
 - gazebo::rendering::DepthCamera, 241
- DisconnectPause
 - gazebo::event::Events, 290
- DisconnectPostRender
 - gazebo::event::Events, 291
- DisconnectPreRender
 - gazebo::event::Events, 291
- DisconnectPubFromSub
 - gazebo::transport::TopicManager, 757
- DisconnectRemoveScene
 - gazebo::rendering::Events, 281
- DisconnectRender
 - gazebo::event::Events, 291
- DisconnectSetSelectedEntity

- gazebo::event::Events, 291
- DisconnectShutdown
 - gazebo::transport::Connection, 210
- DisconnectStep
 - gazebo::event::Events, 291
- DisconnectStop
 - gazebo::event::Events, 292
- DisconnectSubFromPub
 - gazebo::transport::TopicManager, 757
- DisconnectViewContacts
 - gazebo::rendering::Events, 282
- DisconnectWorldCreated
 - gazebo::event::Events, 292
- DisconnectWorldUpdateEnd
 - gazebo::event::Events, 292
- DisconnectWorldUpdateStart
 - gazebo::event::Events, 292
- Distance
 - gazebo::math::Plane, 548
 - gazebo::math::Vector2d, 785
 - gazebo::math::Vector2i, 793
 - gazebo::math::Vector3, 803
 - gazebo::math::Vector4, 815
- Dot
 - gazebo::math::Quaternion, 585
 - gazebo::math::Vector3, 803
- Double
 - gazebo::common::Time, 736
- DownloadDependencies
 - Common, 33
- dragging
 - gazebo::common::MouseEvent, 485
- DrawLine
 - gazebo::rendering::Scene, 637
- dummyContext
 - gazebo::rendering::RenderEngine, 613
- dummyDisplay
 - gazebo::rendering::RenderEngine, 613
- dummyWindowId
 - gazebo::rendering::RenderEngine, 613
- duration
 - gazebo::physics::TrajectoryInfo, 761
- DynamicLines
 - gazebo::rendering::DynamicLines, 252
- DynamicLines.hh, 912
- DynamicLinesPtr
 - gazebo::rendering, 92
- DynamicRenderable
 - gazebo::rendering::DynamicRenderable, 256
- DynamicRenderable.hh, 912
- ENTITY
 - gazebo::physics::Base, 128
- ERP
 - gazebo::physics::Joint, 369
- Element
 - sdf::Element, 261
- ElementPtr
 - sdf, 99
- ElementPtr_V
 - sdf, 99
- emissive
 - gazebo::common::Material, 436
- Enable
 - gazebo::rendering::Grid, 332
- EnableAllModels
 - gazebo::physics::World, 856
- EnablePhysicsEngine
 - gazebo::physics::World, 856
- EnableSaveFrame
 - gazebo::rendering::Camera, 157
- EnableTrackVisual
 - gazebo::rendering::Visual, 836
- EnableViewController
 - gazebo::rendering::UserCamera, 777
- enabled
 - gazebo::rendering::ViewController, 828
- EncodeTopicName
 - gazebo::transport::Node, 501
- endTime
 - gazebo::physics::TrajectoryInfo, 761
- EnqueueMsg
 - gazebo::transport::Connection, 211
- Entity
 - gazebo::physics::Entity, 269
- Entity.hh, 913
- entityCreated
 - gazebo::event::Events, 293
- EntityPtr
 - gazebo::physics, 88
- EntityType
 - gazebo::physics::Base, 128
- EntityTypeName
 - Classes for physics and dynamics, 44
- Equal
 - gazebo::math::Vector3, 803
- equal
 - Math, 48
- erp
 - gazebo::physics::SurfaceParams, 723
- EulerToQuaternion
 - gazebo::math::Quaternion, 585, 586
- Event.hh, 915
- eventConnections
 - gazebo::transport::ConnectionManager, 218
- EventType
 - gazebo::common::MouseEvent, 484
- Events, 36

- ~EventT, 36
- Connect, 36
- ConnectionCount, 37
- Disconnect, 37, 38
- Events.hh, 915
- Exception
 - gazebo::common::Exception, 304
- Exception.hh, 916
- FACE_MAX
 - STLLoader.hh, 1039
- FLAT
 - gazebo::common::Material, 431
- FMAX
 - gazebo::physics::Joint, 369
- FORWARD
 - gazebo::rendering::RenderEngine, 611
- FPSViewController
 - gazebo::rendering::FPSViewController, 306
- FPSViewController.hh, 918
- FUDGE_FACTOR
 - gazebo::physics::Joint, 369
- fakeAnchor
 - gazebo::physics::ScrewJoint, 649
 - gazebo::physics::SliderJoint, 694
- far
 - gazebo::sensors::GpuRaySensor, 329
- fdir1
 - gazebo::physics::SurfaceParams, 724
 - urdf2gazebo::GazeboExtension, 309
- filename
 - gazebo::PluginT, 556
 - sdf::Plugin, 554
- FillArrays
 - gazebo::common::Mesh, 450
 - gazebo::common::SubMesh, 711
- FillCollisionMsg
 - gazebo::physics::Collision, 184
- FillHardwareBuffers
 - gazebo::rendering::DynamicLines, 253
 - gazebo::rendering::DynamicRenderable, 256
- FillJointMsg
 - gazebo::physics::Joint, 371
- FillLinkMsg
 - gazebo::physics::Link, 405
- FillModelMsg
 - gazebo::physics::Model, 465
- FillMsg
 - gazebo::physics::BoxShape, 142
 - gazebo::physics::Collision, 184
 - gazebo::physics::Contact, 220
 - gazebo::physics::CylinderShape, 234
 - gazebo::physics::HeightmapShape, 343
 - gazebo::physics::Joint, 371
 - gazebo::physics::Link, 405
 - gazebo::physics::Model, 465
 - gazebo::physics::MultiRayShape, 495
 - gazebo::physics::PlaneShape, 551
 - gazebo::physics::RayShape, 606
 - gazebo::physics::Shape, 669
 - gazebo::physics::SphereShape, 696
 - gazebo::physics::SurfaceParams, 723
 - gazebo::physics::TrimeshShape, 763
 - gazebo::rendering::Light, 393
 - gazebo::sensors::Sensor, 655
- FillShapeMsg
 - gazebo::physics::BoxShape, 142
 - gazebo::physics::Shape, 669
- FillSurfaceMsg
 - gazebo::physics::SurfaceParams, 723
- find_file
 - Common, 33
 - gazebo, 77
- find_file_path
 - Common, 33
- FindFile
 - gazebo::common::SystemPaths, 728
- FindFileURI
 - gazebo::common::SystemPaths, 728
- FindPublication
 - gazebo::transport::TopicManager, 758
- Fini
 - gazebo::Master, 427
 - gazebo::physics::Actor, 104
 - gazebo::physics::Base, 129
 - gazebo::physics::Collision, 184
 - gazebo::physics::Entity, 269
 - gazebo::physics::Link, 405
 - gazebo::physics::Model, 465
 - gazebo::physics::PhysicsEngine, 535
 - gazebo::physics::World, 857
 - gazebo::rendering::Camera, 157
 - gazebo::rendering::DepthCamera, 241
 - gazebo::rendering::GpuLaser, 315
 - gazebo::rendering::RenderEngine, 612
 - gazebo::rendering::RTShaderSystem, 631
 - gazebo::rendering::UserCamera, 778
 - gazebo::rendering::Visual, 836
 - gazebo::rendering::WindowManager, 850
 - gazebo::sensors::CameraSensor, 175
 - gazebo::sensors::ContactSensor, 226
 - gazebo::sensors::DepthCameraSensor, 244
 - gazebo::sensors::GpuRaySensor, 321
 - gazebo::sensors::ImuSensor, 355
 - gazebo::sensors::RaySensor, 599
 - gazebo::sensors::RFIDSensor, 615
 - gazebo::sensors::RFIDTag, 617
 - gazebo::sensors::Sensor, 655

- gazebo::sensors::SensorManager, 663
- gazebo::Server, 667
- gazebo::transport::ConnectionManager, 215
- gazebo::transport::Node, 501
- gazebo::transport::PublicationTransport, 577
- gazebo::transport::TopicManager, 758
- fini
 - Classes for physics and dynamics, 42
 - gazebo, 77
 - Rendering, 63
 - Sensors, 68
 - Transport, 71
- Float
 - gazebo::common::Time, 737
- FogFromSDF
 - Messages, 57
- fudge_factor
 - urdf2gazebo::GazeboExtension, 309
- g
 - gazebo::common::Color, 204
- GAZEBO_DEPRECATED
 - CommonTypes.hh, 896
- GAZEBO_FORCEINLINE
 - CommonTypes.hh, 896
- GOURAUD
 - gazebo::common::Material, 431
- GUIFromSDF
 - Messages, 57
- GUIOverlay
 - gazebo::rendering::GUIOverlay, 336
- GUIOverlay.hh, 925
- GUIPluginPtr
 - gazebo, 77
- GZ_ALL_COLLIDE
 - PhysicsTypes.hh, 984
- GZ_DBL_MAX
 - Helpers.hh, 929
- GZ_DBL_MIN
 - Helpers.hh, 929
- GZ_DTOR
 - Angle.hh, 874
- GZ_FIXED_COLLIDE
 - PhysicsTypes.hh, 984
- GZ_FLT_MAX
 - Helpers.hh, 929
- GZ_FLT_MIN
 - Helpers.hh, 929
- GZ_GHOST_COLLIDE
 - PhysicsTypes.hh, 984
- GZ_LOG_VERSION
 - LogRecord.hh, 950
- GZ_NONE_COLLIDE
 - PhysicsTypes.hh, 984
- GZ_NORMALIZE
 - Angle.hh, 874
- GZ_REGISTER_MODEL_PLUGIN
 - common/Plugin.hh, 990
- GZ_REGISTER_PHYSICS_ENGINE
 - Classes for physics and dynamics, 42
- GZ_REGISTER_SENSOR_PLUGIN
 - common/Plugin.hh, 990
- GZ_REGISTER_STATIC_SENSOR
 - Sensors, 67
- GZ_REGISTER_SYSTEM_PLUGIN
 - common/Plugin.hh, 991
- GZ_REGISTER_VISUAL_PLUGIN
 - common/Plugin.hh, 991
- GZ_REGISTER_WORLD_PLUGIN
 - common/Plugin.hh, 991
- GZ_RTOD
 - Angle.hh, 874
- GZ_SENSOR_COLLIDE
 - PhysicsTypes.hh, 984
- GZ_VISIBILITY_ALL
 - RenderTypes.hh, 1008
- GZ_VISIBILITY_GUI
 - RenderTypes.hh, 1008
- GZ_VISIBILITY_NOT_SELECTABLE
 - RenderTypes.hh, 1008
- GZ_VISIBILITY_SELECTION
 - RenderTypes.hh, 1008
- gazebo, 75
 - add_plugin, 77
 - find_file, 77
 - fini, 77
 - GUIPluginPtr, 77
 - init, 77
 - load, 77
 - ModelPluginPtr, 77
 - print_version, 77
 - run, 77
 - SensorPluginPtr, 77
 - stop, 77
 - SystemPluginPtr, 77
 - VisualPluginPtr, 77
 - WorldPluginPtr, 77
- gazebo.hh, 918
- gazebo::Master, 426
 - ~Master, 427
 - Fini, 427
 - Init, 427
 - Master, 427
 - Run, 427
 - RunOnce, 427
 - RunThread, 427
 - Stop, 428
- gazebo::ModelPlugin, 475

- ~ModelPlugin, 476
- Init, 476
- Load, 476
- ModelPlugin, 476
- Reset, 477
- gazebo::PluginT
 - Create, 555
 - filename, 556
 - GetFilename, 555
 - GetHandle, 555
 - GetType, 555
 - handle, 556
 - TPtr, 555
 - type, 556
- gazebo::PluginT< T >, 554
- gazebo::SensorPlugin, 665
 - ~SensorPlugin, 666
 - Init, 666
 - Load, 666
 - Reset, 666
 - SensorPlugin, 666
- gazebo::Server, 667
 - ~Server, 667
 - Fini, 667
 - GetInitialized, 667
 - Init, 667
 - LoadFile, 667
 - LoadString, 667
 - ParseArgs, 667
 - PrintUsage, 667
 - Run, 667
 - Server, 667
 - SetParams, 667
 - Stop, 667
 - systemPluginsArgc, 668
 - systemPluginsArgv, 668
- gazebo::SystemPlugin, 730
 - ~SystemPlugin, 731
 - Init, 731
 - Load, 731
 - Reset, 732
 - SystemPlugin, 731
- gazebo::VisualPlugin, 848
 - Init, 848
 - Load, 849
 - Reset, 849
 - VisualPlugin, 848
- gazebo::WorldPlugin, 864
 - ~WorldPlugin, 865
 - Init, 865
 - Load, 865
 - Reset, 865
 - WorldPlugin, 865
- gazebo::common, 77
 - AnimationPtr, 80
 - NodeMap, 80
 - NodeMapIter, 80
 - NumericAnimationPtr, 80
 - Param_V, 80
 - PoseAnimationPtr, 80
 - RawNodeAnim, 80
 - RawNodeWeights, 80
 - RawSkeletonAnim, 80
 - StrStr_M, 80
 - gazebo::common::Animation, 115
 - ~Animation, 116
 - AddTime, 117
 - Animation, 116
 - build, 118
 - GetKeyFrame, 117
 - GetKeyFrameCount, 117
 - GetKeyFramesAtTime, 117
 - GetLength, 117
 - GetTime, 118
 - KeyFrame_V, 116
 - keyFrames, 118
 - length, 118
 - loop, 118
 - name, 119
 - SetLength, 118
 - SetTime, 118
 - timePos, 119
 - gazebo::common::BVHLoader, 143
 - ~BVHLoader, 144
 - BVHLoader, 144
 - Load, 144
 - gazebo::common::ColladaLoader, 179
 - ~ColladaLoader, 180
 - ColladaLoader, 180
 - Load, 180
 - gazebo::common::Color, 193
 - ~Color, 196
 - a, 203
 - ABGR, 196
 - ARGB, 196
 - b, 203
 - BGRA, 196
 - Black, 203
 - Blue, 203
 - Color, 196
 - g, 204
 - GetAsABGR, 196
 - GetAsARGB, 197
 - GetAsBGRA, 197
 - GetAsHSV, 197
 - GetAsRGBA, 197
 - GetAsYUV, 197
 - Green, 204

- operator<<, 203
- operator>>, 203
- operator*, 198
- operator*=: 198
- operator+, 198, 199
- operator+=, 199
- operator-, 199
- operator-=, 200
- operator/, 200
- operator/=, 200
- operator=, 201
- operator==, 201
- operator[], 201
- Purple, 204
- r, 204
- RGBA, 196
- Red, 204
- Reset, 201
- Set, 201
- SetFromABGR, 202
- SetFromARGB, 202
- SetFromBGRA, 202
- SetFromHSV, 202
- SetFromRGBA, 202
- SetFromYUV, 203
- White, 204
- Yellow, 204
- gazebo::common::Console, 218
- gazebo::common::DiagnosticManager, 246
 - CreateTimer, 247
 - GetEnabled, 247
 - GetLabel, 247
 - GetTime, 247, 248
 - GetTimerCount, 248
 - SetEnabled, 248
 - TimerStart, 248
 - TimerStop, 248
- gazebo::common::DiagnosticTimer, 249
 - ~DiagnosticTimer, 250
 - DiagnosticTimer, 249
 - GetName, 250
- gazebo::common::Exception, 303
 - ~Exception, 304
 - Exception, 304
 - GetErrorFile, 304
 - GetErrorStr, 304
 - operator<<, 305
 - Print, 304
- gazebo::common::Image, 349
 - ~Image, 351
 - BAYER_GBRG8, 351
 - BAYER_GRBG8, 351
 - BAYER_RGGB8, 350
 - BAYER_RGGR8, 351
 - BGR_INT16, 350
 - BGR_INT32, 350
 - BGR_INT8, 350
 - BGRA_INT8, 350
 - GetAvgColor, 351
 - GetBPP, 351
 - GetData, 351
 - GetFilename, 351
 - GetHeight, 352
 - GetMaxColor, 352
 - GetPitch, 352
 - GetPixel, 352
 - GetPixelFormat, 352
 - GetRGBData, 352
 - GetWidth, 353
 - Image, 351
 - L_INT16, 350
 - L_INT8, 350
 - Load, 353
 - PixelFormat, 350
 - R_FLOAT16, 350
 - R_FLOAT32, 350
 - RGB_FLOAT16, 350
 - RGB_FLOAT32, 350
 - RGB_INT16, 350
 - RGB_INT32, 350
 - RGB_INT8, 350
 - RGBA_INT8, 350
 - Rescale, 353
 - SavePNG, 353
 - SetFromData, 353
 - UNKNOWN, 350
 - Valid, 354
- gazebo::common::KeyFrame, 388
 - ~KeyFrame, 389
 - GetTime, 390
 - KeyFrame, 389
 - time, 390
- gazebo::common::LogPlay, 421
 - IsOpen, 422
 - Open, 422
 - Step, 423
- gazebo::common::LogRecord, 423
 - Add, 425
 - GetEncoding, 425
 - Init, 425
 - Remove, 425
 - Start, 426
 - Stop, 426
- gazebo::common::Material, 428
 - ~Material, 431
 - ADD, 430
 - ambient, 436
 - BLEND_COUNT, 430

- BLINN, 431
- BlendMode, 430
- blendMode, 436
- BlendModeStr, 436
- diffuse, 436
- emissive, 436
- FLAT, 431
- GOURAUD, 431
- GetAmbient, 431
- GetBlendFactors, 431
- GetBlendMode, 431
- GetDepthWrite, 432
- GetDiffuse, 432
- GetEmissive, 432
- GetLighting, 432
- GetName, 432
- GetPointSize, 432
- GetShadeMode, 433
- GetShininess, 433
- GetSpecular, 433
- GetTextureImage, 433
- GetTransparency, 433
- MODULATE, 430
- Material, 431
- name, 436
- operator<<, 436
- PHONG, 431
- pointSize, 436
- REPLACE, 430
- SHADE_COUNT, 431
- SetAmbient, 433
- SetBlendFactors, 434
- SetBlendMode, 434
- SetDepthWrite, 434
- SetDiffuse, 434
- SetEmissive, 434
- SetLighting, 435
- SetPointSize, 435
- SetShadeMode, 435
- SetShininess, 435
- SetSpecular, 435
- SetTextureImage, 435
- SetTransparency, 436
- ShadeMode, 430
- shadeMode, 437
- ShadeModeStr, 437
- shininess, 437
- specular, 437
- texImage, 437
- transparency, 437
- gazebo::common::Mesh, 448
 - ~Mesh, 450
 - AddMaterial, 450
 - AddSubMesh, 450
 - FillArrays, 450
 - GenSphericalTexCoord, 450
 - GetAABB, 451
 - GetIndexCount, 451
 - GetMaterial, 451
 - GetMaterialCount, 451
 - GetMax, 451
 - GetMin, 451
 - GetName, 452
 - GetNormalCount, 452
 - GetPath, 452
 - GetSkeleton, 452
 - GetSubMesh, 452
 - GetSubMeshCount, 453
 - GetTexCoordCount, 453
 - GetVertexCount, 453
 - HasSkeleton, 453
 - Mesh, 450
 - RecalculateNormals, 453
 - Scale, 453
 - SetName, 453
 - SetPath, 454
 - SetSkeleton, 454
- gazebo::common::MeshLoader, 454
 - ~MeshLoader, 455
 - Load, 455
 - MeshLoader, 455
- gazebo::common::MeshManager, 455
 - AddMesh, 457
 - CreateBox, 457
 - CreateCamera, 457
 - CreateCone, 457
 - CreateCylinder, 458
 - CreatePlane, 458
 - CreateSphere, 458
 - CreateTube, 459
 - GenSphericalTexCoord, 459
 - GetMesh, 459
 - GetMeshAABB, 459
 - HasMesh, 459
 - IsValidFilename, 460
 - Load, 460
- gazebo::common::ModelDatabase, 474
- gazebo::common::MouseEvent, 483
 - alt, 484
 - button, 484
 - Buttons, 484
 - buttons, 485
 - control, 485
 - dragging, 485
 - EventType, 484
 - LEFT, 484
 - MIDDLE, 484
 - MOVE, 484

- MouseEvent, 484
- moveScale, 485
- NO_BUTTON, 484
- NO_EVENT, 484
- PRESS, 484
- pos, 485
- pressPos, 485
- prevPos, 485
- RELEASE, 484
- RIGHT, 484
- SCROLL, 484
- scroll, 485
- shift, 485
- type, 485
- gazebo::common::NodeAnimation, 504
 - ~NodeAnimation, 505
 - AddKeyFrame, 505
 - GetFrameAt, 505
 - GetFrameCount, 506
 - GetKeyFrame, 506
 - GetLength, 506
 - GetName, 507
 - GetTimeAtX, 507
 - keyFrames, 507
 - length, 507
 - name, 507
 - NodeAnimation, 505
 - Scale, 507
 - SetName, 507
- gazebo::common::NodeAssignment, 508
 - nodeIndex, 508
 - vertexIndex, 508
 - weight, 508
- gazebo::common::NodeTransform, 509
 - ~NodeTransform, 511
 - Get, 511
 - GetSID, 511
 - GetType, 511
 - MATRIX, 510
 - NodeTransform, 510
 - operator*, 511, 512
 - operator(), 511
 - PrintSource, 512
 - ROTATE, 510
 - RecalculateMatrix, 512
 - SCALE, 510
 - Set, 512
 - SetComponent, 512
 - SetSID, 512
 - SetSourceValues, 512, 513
 - SetType, 513
 - sid, 513
 - source, 513
 - TRANSLATE, 510
 - transform, 513
 - TransformType, 510
 - type, 513
- gazebo::common::NumericAnimation, 514
 - ~NumericAnimation, 515
 - CreateKeyFrame, 515
 - GetInterpolatedKeyFrame, 515
 - NumericAnimation, 514
- gazebo::common::NumericKeyFrame, 515
 - ~NumericKeyFrame, 516
 - GetValue, 517
 - NumericKeyFrame, 516
 - SetValue, 517
 - value, 517
- gazebo::common::PID, 543
 - ~PID, 544
 - GetCmd, 544
 - GetErrors, 544
 - Init, 544
 - operator=, 545
 - PID, 544
 - Reset, 545
 - SetCmd, 545
 - SetCmdMax, 545
 - SetCmdMin, 545
 - SetDGain, 546
 - SetIGain, 546
 - SetIMax, 546
 - SetIMin, 546
 - SetPGain, 546
 - Update, 546
- gazebo::common::PoseAnimation, 564
 - ~PoseAnimation, 566
 - BuildInterpolationSplines, 566
 - CreateKeyFrame, 566
 - GetInterpolatedKeyFrame, 566
 - PoseAnimation, 565
- gazebo::common::PoseKeyFrame, 567
 - ~PoseKeyFrame, 568
 - GetRotation, 568
 - GetTranslation, 568
 - PoseKeyFrame, 568
 - rotate, 569
 - SetRotation, 568
 - SetTranslation, 568
 - translate, 569
- gazebo::common::STLLoader, 705
 - ~STLLoader, 706
 - Load, 706
 - STLLoader, 706
- gazebo::common::Skeleton, 672
 - ~Skeleton, 674
 - AddAnimation, 674
 - AddVertNodeWeight, 675

- anims, 678
- bindShapeTransform, 678
- BuildNodeMap, 675
- GetAnimation, 675
- GetBindShapeTransform, 675
- GetNodeByHandle, 675
- GetNodeById, 675
- GetNodeByName, 676
- GetNodes, 676
- GetNumAnimations, 676
- GetNumJoints, 676
- GetNumNodes, 676
- GetNumVertNodeWeights, 676
- GetRootNode, 677
- GetVertNodeWeight, 677
- nodes, 678
- PrintTransforms, 677
- rawNW, 678
- root, 678
- Scale, 677
- SetBindShapeTransform, 677
- SetNumVertAttached, 678
- SetRootNode, 678
- Skeleton, 674
- gazebo::common::SkeletonAnimation, 679
 - ~SkeletonAnimation, 680
 - AddKeyFrame, 680
 - animations, 682
 - GetLength, 680
 - GetName, 681
 - GetNodeCount, 681
 - GetNodePoseAt, 681
 - GetPoseAt, 681
 - GetPoseAtX, 682
 - HasNode, 682
 - length, 682
 - name, 683
 - Scale, 682
 - SetName, 682
 - SkeletonAnimation, 680
- gazebo::common::SkeletonNode, 683
 - ~SkeletonNode, 686
 - AddChild, 686
 - AddRawTransform, 686
 - children, 691
 - GetChild, 686
 - GetChildById, 686
 - GetChildByName, 687
 - GetChildCount, 687
 - GetHandle, 687
 - GetId, 687
 - GetInverseBindTransform, 687
 - GetModelTransform, 687
 - GetName, 688
 - GetNumRawTrans, 688
 - GetParent, 688
 - GetRawTransform, 688
 - GetRawTransforms, 688
 - GetTransform, 689
 - GetTransforms, 689
 - handle, 691
 - id, 691
 - initialTransform, 691
 - invBindTransform, 691
 - IsJoint, 689
 - IsRootNode, 689
 - JOINT, 685
 - modelTransform, 691
 - NODE, 685
 - name, 691
 - parent, 692
 - rawTransforms, 692
 - Reset, 689
 - SetHandle, 689
 - SetId, 689
 - SetInitialTransform, 690
 - SetInverseBindTransform, 690
 - SetModelTransform, 690
 - SetName, 690
 - SetParent, 690
 - SetTransform, 690
 - SetType, 691
 - SkeletonNode, 685, 686
 - SkeletonNodeType, 685
 - transform, 692
 - type, 692
 - UpdateChildrenTransforms, 691
- gazebo::common::SubMesh, 707
 - ~SubMesh, 709
 - AddIndex, 709
 - AddNodeAssignment, 710
 - AddNormal, 710
 - AddTexCoord, 710
 - AddVertex, 710
 - CopyNormals, 711
 - CopyVertices, 711
 - FillArrays, 711
 - GenSphericalTexCoord, 711
 - GetIndex, 711
 - GetIndexCount, 712
 - GetMaterialIndex, 712
 - GetMax, 712
 - GetMaxIndex, 712
 - GetMin, 712
 - GetNodeAssignment, 712
 - GetNodeAssignmentsCount, 712
 - GetNormal, 712
 - GetNormalCount, 713

- GetPrimitiveType, 713
- GetTexCoord, 713
- GetTexCoordCount, 713
- GetVertex, 713
- GetVertexCount, 713
- GetVertexIndex, 714
- HasVertex, 714
- LINES, 709
- LINESTRIPS, 709
- POINTS, 709
- PrimitiveType, 709
- RecalculateNormals, 714
- Scale, 714
- SetIndexCount, 714
- SetMaterialIndex, 714
- SetNormal, 715
- SetNormalCount, 715
- SetPrimitiveType, 715
- SetSubMeshCenter, 715
- SetTexCoord, 715
- SetTexCoordCount, 715
- SetVertex, 716
- SetVertexCount, 716
- SubMesh, 709
- TRIANGLES, 709
- TRIFANS, 709
- TRISTRIPS, 709
- gazebo::common::SystemPaths, 726
 - AddGazeboPaths, 727
 - AddOgrePaths, 727
 - AddPluginPaths, 727
 - AddSearchPathSuffix, 728
 - ClearGazeboPaths, 728
 - ClearOgrePaths, 728
 - ClearPluginPaths, 728
 - FindFile, 728
 - FindFileURI, 728
 - gazeboPathsFromEnv, 730
 - GetGazeboPaths, 729
 - GetLogPath, 729
 - GetModelPaths, 729
 - GetOgrePaths, 729
 - GetPluginPaths, 729
 - GetWorldPathExtension, 729
 - modelPathsFromEnv, 730
 - ogrePathsFromEnv, 730
 - pluginPathsFromEnv, 730
- gazebo::common::Time, 732
 - ~Time, 736
 - Double, 736
 - Float, 737
 - GetWallTime, 737
 - MSleep, 737
 - MicToNano, 737
 - MilToNano, 737
 - NSleep, 738
 - nsec, 752
 - operator<, 745, 746
 - operator<<, 751
 - operator<=, 746, 747
 - operator>, 749
 - operator>>, 752
 - operator>=, 750
 - operator*, 739
 - operator*=: 740
 - operator+, 740, 741
 - operator+=, 741, 742
 - operator-, 742
 - operator-=, 743
 - operator/, 743, 744
 - operator/=, 744, 745
 - operator=, 747
 - operator==, 748
 - sec, 752
 - SecToNano, 751
 - Set, 751
 - SetToWallTime, 751
 - Time, 735, 736
- gazebo::common::Timer, 752
 - ~Timer, 753
 - GetElapsed, 754
 - operator<<, 754
 - Start, 754
 - Timer, 753
- gazebo::common::Video, 821
 - ~Video, 822
 - GetHeight, 822
 - GetNextFrame, 822
 - GetWidth, 822
 - Load, 823
 - Video, 822
- gazebo::event, 80
 - Connection_V, 81
 - ConnectionPtr, 81
- gazebo::event::Connection, 206
 - ~Connection, 207
 - Connection, 207
 - GetId, 207
- gazebo::event::Event, 277
 - ~Event, 279
 - Disconnect, 279
- gazebo::event::EventT
 - operator(), 297–299
 - Signal, 300–302
- gazebo::event::EventT < T >, 294
- gazebo::event::Events, 282
 - addEntity, 292
 - ConnectAddEntity, 285

- ConnectCreateEntity, 285
- ConnectDeleteEntity, 285
- ConnectDiagTimerStart, 286
- ConnectDiagTimerStop, 286
- ConnectPause, 286
- ConnectPostRender, 287
- ConnectPreRender, 287
- ConnectRender, 287
- ConnectSetSelectedEntity, 288
- ConnectStep, 288
- ConnectStop, 288
- ConnectWorldCreated, 288
- ConnectWorldUpdateEnd, 289
- ConnectWorldUpdateStart, 289
- deleteEntity, 292
- diagTimerStart, 293
- diagTimerStop, 293
- DisconnectAddEntity, 289
- DisconnectCreateEntity, 289
- DisconnectDeleteEntity, 290
- DisconnectDiagTimerStart, 290
- DisconnectDiagTimerStop, 290
- DisconnectPause, 290
- DisconnectPostRender, 291
- DisconnectPreRender, 291
- DisconnectRender, 291
- DisconnectSetSelectedEntity, 291
- DisconnectStep, 291
- DisconnectStop, 292
- DisconnectWorldCreated, 292
- DisconnectWorldUpdateEnd, 292
- DisconnectWorldUpdateStart, 292
- entityCreated, 293
- pause, 293
- postRender, 293
- preRender, 293
- render, 293
- setSelectedEntity, 293
- step, 293
- stop, 294
- worldCreated, 294
- worldUpdateEnd, 294
- worldUpdateStart, 294
- gazebo::math, 81
 - GeneratorType, 83
 - NRealGen, 83
 - NormalRealDist, 83
 - UIntGen, 83
 - URealGen, 83
 - UniformIntDist, 83
 - UniformRealDist, 83
- gazebo::math::Angle, 107
 - ~Angle, 109
 - Angle, 109
 - Degree, 109
 - GetAsDegree, 109
 - GetAsRadian, 109
 - Normalize, 110
 - operator<, 112
 - operator<<, 114
 - operator<=, 112
 - operator>, 113
 - operator>>, 114
 - operator>=, 113
 - operator*, 110
 - operator*=:, 110
 - operator+, 111
 - operator+=, 111
 - operator-, 111
 - operator-=, 111
 - operator/, 112
 - operator/=, 112
 - operator==, 113
 - Radian, 113
 - SetFromDegree, 114
 - SetFromRadian, 114
- gazebo::math::Box, 136
 - ~Box, 137
 - Box, 137
 - GetCenter, 137
 - GetSize, 137
 - GetXLength, 138
 - GetYLength, 138
 - GetZLength, 138
 - max, 140
 - Merge, 138
 - min, 140
 - operator<<, 140
 - operator+, 138
 - operator+=, 139
 - operator-, 139
 - operator=, 139
 - operator==, 139
- gazebo::math::Matrix3, 437
 - ~Matrix3, 439
 - m, 441
 - Matrix3, 438
 - operator<<, 440
 - operator==, 439
 - operator[], 439
 - SetCol, 440
 - SetFromAxes, 440
 - SetFromAxis, 440
- gazebo::math::Matrix4, 441
 - ~Matrix4, 443
 - GetAsPose, 443
 - GetEulerRotation, 444
 - GetRotation, 444

- GetTranslation, 444
- IDENTITY, 448
- Inverse, 444
- IsAffine, 444
- m, 448
- Matrix4, 443
- operator<<, 447
- operator*, 444, 445
- operator=, 445
- operator==, 446
- operator[], 446
- Set, 446
- SetScale, 447
- SetTranslate, 447
- TransformAffine, 447
- ZERO, 448
- gazebo::math::Plane, 547
 - ~Plane, 548
 - d, 549
 - Distance, 548
 - normal, 549
 - operator=, 548
 - Plane, 548
 - Set, 549
 - size, 549
- gazebo::math::Pose, 556
 - ~Pose, 559
 - CoordPoseSolve, 559
 - CoordPositionAdd, 559
 - CoordPositionSub, 559
 - CoordRotationAdd, 560
 - CoordRotationSub, 560
 - Correct, 560
 - GetInverse, 560
 - IsFinite, 561
 - operator<<, 563
 - operator>>, 564
 - operator*, 561
 - operator+, 561
 - operator+=, 561
 - operator-, 562
 - operator-=, 562
 - operator==, 562
 - pos, 564
 - Pose, 558
 - Reset, 562
 - rot, 564
 - RotatePositionAboutOrigin, 562
 - Round, 563
 - Set, 563
 - Zero, 564
- gazebo::math::Quaternion, 581
 - ~Quaternion, 585
 - Correct, 585
 - Dot, 585
 - EulerToQuaternion, 585, 586
 - GetAsAxis, 586
 - GetAsEuler, 586
 - GetAsMatrix3, 586
 - GetAsMatrix4, 586
 - GetExp, 586
 - GetInverse, 587
 - GetLog, 587
 - GetPitch, 587
 - GetRoll, 587
 - GetXAxis, 587
 - GetYAxis, 588
 - GetYaw, 587
 - GetZAxis, 588
 - Invert, 588
 - IsFinite, 588
 - Normalize, 588
 - operator<<, 593
 - operator>>, 594
 - operator*, 589
 - operator*=, 589
 - operator+, 589
 - operator+=, 590
 - operator-, 590
 - operator-=, 590
 - operator=, 591
 - operator==, 591
 - Quaternion, 584, 585
 - RotateVector, 591
 - RotateVectorReverse, 591
 - Round, 592
 - Scale, 592
 - Set, 592
 - SetFromAxis, 592
 - SetFromEuler, 593
 - SetToIdentity, 593
 - Slerp, 593
 - Squad, 593
 - w, 594
 - x, 594
 - y, 594
 - z, 594
- gazebo::math::Rand, 595
 - GetDbfNormal, 595
 - GetDbfUniform, 595
 - GetIntNormal, 595
 - GetIntUniform, 596
 - GetSeed, 596
 - SetSeed, 596
- gazebo::math::RotationSpline, 624
 - ~RotationSpline, 625
 - AddPoint, 625
 - autoCalc, 628

- Clear, 625
- GetNumPoints, 626
- GetPoint, 626
- Interpolate, 626, 627
- points, 628
- RecalcTangents, 627
- RotationSpline, 625
- SetAutoCalculate, 627
- tangents, 628
- UpdatePoint, 627
- gazebo::math::Spline, 697
 - ~Spline, 698
 - AddPoint, 699
 - autoCalc, 701
 - Clear, 699
 - coeffs, 701
 - GetPoint, 699
 - GetPointCount, 699
 - GetTangent, 699
 - GetTension, 699
 - Interpolate, 700
 - points, 701
 - RecalcTangents, 700
 - SetAutoCalculate, 700
 - SetTension, 700
 - Spline, 698
 - tangents, 701
 - tension, 701
 - UpdatePoint, 701
- gazebo::math::Vector2d, 782
 - ~Vector2d, 784
 - Cross, 784
 - Distance, 785
 - IsFinite, 785
 - Normalize, 785
 - operator<<, 790
 - operator>>, 790
 - operator*, 785
 - operator*=:, 786
 - operator+, 786
 - operator+=, 787
 - operator-, 787
 - operator-=, 787
 - operator/, 787, 788
 - operator/=, 788
 - operator=, 788, 789
 - operator==, 789
 - operator[], 789
 - Set, 789
 - Vector2d, 784
 - x, 790
 - y, 790
- gazebo::math::Vector2i, 790
 - ~Vector2i, 793
- Cross, 793
- Distance, 793
- IsFinite, 793
- Normalize, 793
- operator<<, 798
- operator>>, 799
- operator*, 794
- operator*=:, 794, 795
- operator+, 795
- operator+=, 795
- operator-, 795
- operator-=, 796
- operator/, 796
- operator/=, 796, 797
- operator=, 797
- operator==, 798
- operator[], 798
- Set, 798
- Vector2i, 792
- x, 799
- y, 799
- gazebo::math::Vector3, 799
 - ~Vector3, 802
 - Correct, 803
 - Cross, 803
 - Distance, 803
 - Dot, 803
 - Equal, 803
 - GetAbs, 804
 - GetDistToLine, 804
 - GetLength, 804
 - GetMax, 804
 - GetMin, 804
 - GetNormal, 805
 - GetPerpendicular, 805
 - GetRounded, 805
 - GetSquaredLength, 805
 - GetSum, 805
 - IsFinite, 806
 - Normalize, 806
 - operator<<, 811
 - operator>>, 811
 - operator*, 806
 - operator*=:, 807
 - operator+, 807
 - operator+=, 807
 - operator-, 808
 - operator-=, 808
 - operator/, 808
 - operator/=, 809
 - operator=, 809
 - operator==, 810
 - operator[], 810
 - Round, 810

- Set, 810
- SetToMax, 810
- SetToMin, 811
- Vector3, 802
- x, 811
- y, 812
- z, 812
- Zero, 812
- gazebo::math::Vector4, 812
 - ~Vector4, 814
 - Distance, 815
 - GetLength, 815
 - GetSquaredLength, 815
 - IsFinite, 815
 - Normalize, 815
 - operator<<, 820
 - operator>>, 821
 - operator*, 816
 - operator*=:, 816, 817
 - operator+, 817
 - operator+=, 817
 - operator-, 817
 - operator-=, 818
 - operator/, 818
 - operator/=, 819
 - operator=, 819
 - operator==, 820
 - operator[], 820
 - Set, 820
 - Vector4, 814
 - w, 821
 - x, 821
 - y, 821
 - z, 821
- gazebo::msgs, 83
- gazebo::physics, 85
 - Actor_V, 88
 - ActorPtr, 88
 - Base_V, 88
 - BasePtr, 88
 - BoxShapePtr, 88
 - Collision_V, 88
 - CollisionPtr, 88
 - ContactPtr, 88
 - CylinderShapePtr, 88
 - EntityPtr, 88
 - HeightmapShapePtr, 88
 - InertialPtr, 89
 - Joint_V, 89
 - JointPtr, 89
 - Link_V, 89
 - LinkPtr, 89
 - MeshShapePtr, 89
 - Model_V, 89
 - ModelPtr, 89
 - MultiRayShapePtr, 89
 - PhysicsEnginePtr, 89
 - RayShapePtr, 89
 - RoadPtr, 89
 - ShapePtr, 89
 - SphereShapePtr, 89
 - SurfaceParamsPtr, 89
 - WorldPtr, 89
- gazebo::physics::Actor, 101
 - ~Actor, 104
 - active, 105
 - Actor, 103
 - autoStart, 105
 - bonePosePub, 105
 - Fini, 104
 - GetSDF, 104
 - Init, 104
 - interpolateX, 105
 - IsActive, 104
 - lastPos, 105
 - lastScriptTime, 105
 - lastTraj, 105
 - Load, 104
 - loop, 105
 - mainLink, 106
 - mesh, 106
 - oldAction, 106
 - pathLength, 106
 - Play, 104
 - playStartTime, 106
 - prevFrameTime, 106
 - scriptLength, 106
 - skelAnimation, 106
 - skelNodesMap, 106
 - skeleton, 106
 - skinFile, 106
 - skinScale, 107
 - startDelay, 107
 - Stop, 104
 - trajInfo, 107
 - trajectories, 107
 - Update, 104
 - UpdateParameters, 105
 - visualName, 107
- gazebo::physics::BallJoint
 - ~BallJoint, 124
 - BallJoint, 124
 - GetAngleCount, 124
 - GetHighStop, 124
 - GetLowStop, 124
 - Load, 124
 - SetAxis, 124
 - SetHighStop, 125

SetLowStop, 125
 gazebo::physics::BallJoint< T >, 123
 gazebo::physics::Base, 125
 ~Base, 129
 ACTOR, 128
 AddChild, 129
 AddType, 129
 BALL_JOINT, 128
 BASE, 128
 BOX_SHAPE, 128
 Base, 128
 COLLISION, 128
 CYLINDER_SHAPE, 128
 children, 135
 childrenEnd, 135
 ENTITY, 128
 EntityType, 128
 Fini, 129
 GetById, 129
 GetByName, 129
 GetChild, 130
 GetChildCount, 130
 GetId, 130
 GetName, 130
 GetParent, 131
 GetParentId, 131
 GetSDF, 131
 GetSaveable, 131
 GetScopedName, 131
 GetType, 131
 GetWorld, 132
 HEIGHTMAP_SHAPE, 128
 HINGE2_JOINT, 128
 HINGE_JOINT, 128
 HasType, 132
 Init, 132
 IsSelected, 132
 JOINT, 128
 LIGHT, 128
 LINK, 128
 Load, 132
 MAP_SHAPE, 128
 MODEL, 128
 MULTIRAY_SHAPE, 128
 operator==, 133
 PLANE_SHAPE, 128
 parent, 135
 Print, 133
 RAY_SHAPE, 128
 RemoveChild, 133
 RemoveChildren, 133
 Reset, 133, 134
 SCREW_JOINT, 128
 SHAPE, 128
 SLIDER_JOINT, 128
 SPHERE_SHAPE, 128
 sdf, 135
 SetName, 134
 SetParent, 134
 SetSaveable, 134
 SetSelected, 134
 SetWorld, 134
 TRIMESH_SHAPE, 128
 UNIVERSAL_JOINT, 128
 Update, 135
 UpdateParameters, 135
 VISUAL, 128
 world, 135
 gazebo::physics::BoxShape, 140
 ~BoxShape, 142
 BoxShape, 142
 FillMsg, 142
 FillShapeMsg, 142
 GetInertial, 142
 GetMass, 142
 GetSize, 142
 Init, 143
 ProcessMsg, 143
 SetSize, 143
 gazebo::physics::Collision, 180
 ~Collision, 183
 AddContact, 183
 Collision, 183
 ConnectContact, 183
 DisconnectContact, 184
 FillCollisionMsg, 184
 FillMsg, 184
 Fini, 184
 GetBoundingBox, 184
 GetContactsEnabled, 184
 GetLaserRetro, 184
 GetLink, 185
 GetModel, 185
 GetRelativeAngularAccel, 185
 GetRelativeAngularVel, 185
 GetRelativeLinearAccel, 185
 GetRelativeLinearVel, 185
 GetShape, 186
 GetShapeType, 186
 GetState, 186
 GetSurface, 186
 GetWorldAngularAccel, 186
 GetWorldAngularVel, 187
 GetWorldLinearAccel, 187
 GetWorldLinearVel, 187
 Init, 187
 IsPlaceable, 187
 link, 189

- Load, 187
- placeable, 189
- ProcessMsg, 188
- SetCategoryBits, 188
- SetCollideBits, 188
- SetCollision, 188
- SetContactsEnabled, 188
- SetLaserRetro, 188
- SetShape, 189
- SetState, 189
- shape, 189
- UpdateParameters, 189
- gazebo::physics::CollisionState, 190
 - ~CollisionState, 191
 - CollisionState, 191
 - GetPose, 191
 - IsZero, 192
 - Load, 192
 - operator<<, 193
 - operator+, 192
 - operator-, 192
 - operator=, 192
- gazebo::physics::Contact, 218
 - ~Contact, 220
 - Clone, 220
 - collision1, 221
 - collision2, 221
 - Contact, 220
 - count, 221
 - DebugString, 220
 - depths, 221
 - FillMsg, 220
 - normals, 221
 - operator=, 220, 221
 - positions, 221
 - Reset, 221
 - time, 221
 - wrench, 222
- gazebo::physics::ContactManager, 222
 - ~ContactManager, 223
 - Clear, 223
 - ContactManager, 223
 - GetContact, 223
 - GetContactCount, 223
 - GetContacts, 223
 - Init, 223
 - NewContact, 224
 - PublishContacts, 224
 - ResetCount, 224
- gazebo::physics::CylinderShape, 233
 - ~CylinderShape, 234
 - CylinderShape, 234
 - FillMsg, 234
 - GetInertial, 234
 - GetLength, 234
 - GetMass, 235
 - GetRadius, 235
 - Init, 235
 - ProcessMsg, 235
 - SetLength, 235
 - SetRadius, 235
 - SetSize, 236
- gazebo::physics::Entity, 265
 - ~Entity, 269
 - animation, 276
 - animationConnection, 276
 - animationStartPose, 276
 - connections, 276
 - dirtyPose, 276
 - Entity, 269
 - Fini, 269
 - GetBoundingBox, 269
 - GetChildCollision, 269
 - GetChildLink, 269
 - GetCollisionBoundingBox, 270
 - GetDirtyPose, 270
 - GetNearestEntityBelow, 270
 - GetParentModel, 270
 - GetRelativeAngularAccel, 270
 - GetRelativeAngularVel, 271
 - GetRelativeLinearAccel, 271
 - GetRelativeLinearVel, 271
 - GetRelativePose, 271
 - GetWorldAngularAccel, 271
 - GetWorldAngularVel, 272
 - GetWorldLinearAccel, 272
 - GetWorldLinearVel, 272
 - GetWorldPose, 272
 - IsCanonicalLink, 272
 - IsStatic, 273
 - Load, 273
 - node, 276
 - OnPoseChange, 273
 - parentEntity, 276
 - PlaceOnEntity, 273
 - PlaceOnNearestEntityBelow, 273
 - poseMsg, 276
 - prevAnimationTime, 276
 - requestPub, 277
 - Reset, 273
 - SetAnimation, 274
 - SetCanonicalLink, 274
 - SetInitialRelativePose, 274
 - SetName, 274
 - SetRelativePose, 274
 - SetStatic, 275
 - SetWorldPose, 275
 - SetWorldTwist, 275

- StopAnimation, 275
- UpdateParameters, 275
- visPub, 277
- visualMsg, 277
- gazebo::physics::Gripper, 334
 - ~Gripper, 335
 - Gripper, 335
 - Init, 335
 - Load, 335
- gazebo::physics::HeightmapShape, 341
 - ~HeightmapShape, 343
 - FillMsg, 343
 - GetHeight, 343
 - GetMaxHeight, 343
 - GetMinHeight, 343
 - GetPos, 344
 - GetSize, 344
 - GetSubSampling, 344
 - GetURI, 344
 - GetVertexCount, 344
 - HeightmapShape, 343
 - heights, 345
 - img, 345
 - Init, 344
 - Load, 344
 - ProcessMsg, 345
 - scale, 345
 - subSampling, 345
 - vertSize, 345
- gazebo::physics::Hinge2Joint
 - ~Hinge2Joint, 346
 - GetAngleCount, 347
 - Hinge2Joint, 346
 - Load, 347
- gazebo::physics::Hinge2Joint< T >, 345
- gazebo::physics::HingeJoint
 - ~HingeJoint, 348
 - GetAngleCount, 348
 - HingeJoint, 348
 - Init, 348
 - Load, 348
- gazebo::physics::HingeJoint< T >, 347
- gazebo::physics::Inertial, 357
 - ~Inertial, 359
 - GetCoG, 359
 - GetIX, 359
 - GetIXY, 359
 - GetIXZ, 360
 - GetIYY, 360
 - GetIYZ, 360
 - GetIZZ, 360
 - GetMass, 360
 - GetPose, 360
 - GetPrincipalMoments, 360
 - GetProductsofInertia, 361
 - Inertial, 359
 - Load, 361
 - operator<<, 364
 - operator+, 361
 - operator+=", 361
 - operator=, 361
 - ProcessMsg, 362
 - Reset, 362
 - Rotate, 362
 - SetCoG, 362
 - SetIX, 363
 - SetIXY, 363
 - SetIXZ, 363
 - SetIYY, 363
 - SetIYZ, 363
 - SetIZZ, 364
 - SetInertiaMatrix, 363
 - SetMass, 364
 - UpdateParameters, 364
- gazebo::physics::Joint, 366
 - ~Joint, 370
 - anchorLink, 379
 - anchorPos, 379
 - AreConnected, 370
 - Attach, 370
 - Attribute, 369
 - CFM, 369
 - childLink, 379
 - ConnectJointUpdate, 370
 - damping_coefficient, 379
 - Detach, 371
 - DisconnectJointUpdate, 371
 - ERP, 369
 - FMAX, 369
 - FUDGE_FACTOR, 369
 - FillJointMsg, 371
 - FillMsg, 371
 - GetAnchor, 371
 - GetAngle, 372
 - GetAngleCount, 372
 - GetAngleImpl, 372
 - GetChild, 372
 - GetForce, 372
 - GetGlobalAxis, 373
 - GetHighStop, 373
 - GetJointLink, 373
 - GetLinkForce, 373
 - GetLinkTorque, 374
 - GetLocalAxis, 374
 - GetLowStop, 374
 - GetMaxForce, 375
 - GetParent, 375
 - GetVelocity, 375

- HI_STOP, 369
- Init, 375
- Joint, 370
- LO_STOP, 369
- Load, 375, 376
- model, 379
- parentLink, 379
- Reset, 376
- STOP_CFM, 369
- STOP_ERP, 369
- SUSPENSION_CFM, 369
- SUSPENSION_ERP, 369
- SetAnchor, 376
- SetAngle, 376
- SetAttribute, 376, 377
- SetAxis, 377
- SetDamping, 377
- SetForce, 377
- SetHighStop, 377
- SetLowStop, 378
- SetMaxForce, 378
- SetModel, 378
- SetState, 378
- SetVelocity, 378
- Update, 379
- UpdateParameters, 379
- VEL, 369
- gazebo::physics::JointController, 380
 - AddJoint, 380
 - JointController, 380
 - Reset, 380
 - SetJointPosition, 381
 - SetJointPositions, 381
 - Update, 381
- gazebo::physics::JointState, 381
 - ~JointState, 383
 - GetAngle, 383
 - GetAngleCount, 384
 - GetAngles, 384
 - IsZero, 384
 - JointState, 383
 - Load, 384
 - operator<<, 385
 - operator+, 384
 - operator-, 384
 - operator=, 385
- gazebo::physics::JointWrench, 387
 - body1Force, 388
 - body1Torque, 388
 - body2Force, 388
 - body2Torque, 388
 - operator=, 388
- gazebo::physics::Link, 398
 - ~Link, 402
 - AddChildJoint, 402
 - AddForce, 403
 - AddForceAtRelativePosition, 403
 - AddForceAtWorldPosition, 403
 - AddParentJoint, 403
 - AddRelativeForce, 403
 - AddRelativeTorque, 403
 - AddTorque, 404
 - angularAccel, 415
 - AttachStaticModel, 404
 - attachedModelsOffset, 415
 - cgVisuals, 415
 - ConnectEnabled, 404
 - DetachAllStaticModels, 404
 - DetachStaticModel, 404
 - DisconnectEnabled, 405
 - FillLinkMsg, 405
 - FillMsg, 405
 - Fini, 405
 - GetAngularDamping, 405
 - GetBoundingBox, 405
 - GetChildJointsLinks, 405
 - GetCollision, 406
 - GetCollisionById, 406
 - GetCollisions, 406
 - GetEnabled, 407
 - GetGravityMode, 407
 - GetInertial, 407
 - GetKinematic, 407
 - GetLinearDamping, 407
 - GetModel, 407
 - GetParentJointsLinks, 408
 - GetRelativeAngularAccel, 408
 - GetRelativeAngularVel, 408
 - GetRelativeForce, 408
 - GetRelativeLinearAccel, 408
 - GetRelativeLinearVel, 408
 - GetRelativeTorque, 409
 - GetSelfCollide, 409
 - GetSensorCount, 409
 - GetSensorName, 409
 - GetWorldAngularAccel, 410
 - GetWorldForce, 410
 - GetWorldLinearAccel, 410
 - GetWorldTorque, 410
 - inertial, 415
 - Init, 410
 - linearAccel, 415
 - Link, 402
 - Load, 410
 - OnPoseChange, 411
 - ProcessMsg, 411
 - RemoveChildJoint, 411
 - RemoveParentJoint, 411

- Reset, 411
- SetAngularAccel, 411
- SetAngularDamping, 411
- SetAngularVel, 412
- SetAutoDisable, 412
- SetCollideMode, 412
- SetEnabled, 412
- SetForce, 412
- SetGravityMode, 413
- SetInertial, 413
- SetKinematic, 413
- SetLaserRetro, 413
- SetLinearAccel, 413
- SetLinearDamping, 413
- SetLinearVel, 414
- SetSelected, 414
- SetSelfCollide, 414
- SetState, 414
- SetTorque, 414
- Update, 414
- UpdateMass, 415
- UpdateParameters, 415
- UpdateSurface, 415
- visuals, 415
- gazebo::physics::LinkState, 416
 - ~LinkState, 418
 - GetAcceleration, 418
 - GetCollisionState, 418
 - GetCollisionStateCount, 419
 - GetCollisionStates, 419
 - GetPose, 419
 - GetVelocity, 419
 - GetWrench, 419
 - IsZero, 420
 - LinkState, 417
 - Load, 420
 - operator<<, 421
 - operator+, 420
 - operator-, 420
 - operator=, 421
- gazebo::physics::Model, 460
 - ~Model, 464
 - AttachStaticModel, 464
 - attachedModels, 474
 - attachedModelsOffset, 474
 - DetachStaticModel, 464
 - FillModelMsg, 465
 - FillMsg, 465
 - Fini, 465
 - GetAllLinks, 465
 - GetAutoDisable, 465
 - GetBoundingBox, 465
 - GetJoint, 466
 - GetJointCount, 466
 - GetJoints, 466
 - GetLink, 466, 467
 - GetLinkById, 467
 - GetLinks, 467
 - GetPluginCount, 467
 - GetRelativeAngularAccel, 467
 - GetRelativeAngularVel, 467
 - GetRelativeLinearAccel, 468
 - GetRelativeLinearVel, 468
 - GetSDF, 468
 - GetSensorCount, 468
 - GetWorldAngularAccel, 468
 - GetWorldAngularVel, 469
 - GetWorldLinearAccel, 469
 - GetWorldLinearVel, 469
 - Init, 469
 - Load, 469
 - LoadPlugins, 470
 - Model, 464
 - OnPoseChange, 470
 - ProcessMsg, 470
 - RemoveChild, 470
 - Reset, 470
 - SetAngularAccel, 470
 - SetAngularVel, 470
 - SetAutoDisable, 471
 - SetCollideMode, 471
 - SetEnabled, 471
 - SetGravityMode, 471
 - SetJointAnimation, 471
 - SetJointPosition, 472
 - SetJointPositions, 472
 - SetLaserRetro, 472
 - SetLinearAccel, 472
 - SetLinearVel, 472
 - SetLinkWorldPose, 473
 - SetState, 473
 - StopAnimation, 473
 - Update, 473
 - UpdateParameters, 473
- gazebo::physics::ModelState, 477
 - ~ModelState, 479
 - GetJointState, 479
 - GetJointStateCount, 480
 - GetJointStates, 480
 - GetLinkState, 480
 - GetLinkStateCount, 481
 - GetLinkStates, 481
 - GetPose, 481
 - IsZero, 481
 - Load, 481
 - ModelState, 478, 479
 - operator<<, 482
 - operator+, 481

- operator-, 482
- operator=, 482
- gazebo::physics::MultiRayShape, 492
 - ~MultiRayShape, 494
 - AddRay, 494
 - ConnectNewLaserScans, 494
 - DisconnectNewLaserScans, 495
 - FillMsg, 495
 - GetFiducial, 495
 - GetMaxAngle, 495
 - GetMaxRange, 495
 - GetMinAngle, 496
 - GetMinRange, 496
 - GetRange, 496
 - GetResRange, 496
 - GetRetro, 496
 - GetSampleCount, 497
 - GetScanResolution, 497
 - GetVerticalMaxAngle, 497
 - GetVerticalMinAngle, 497
 - GetVerticalSampleCount, 497
 - GetVerticalScanResolution, 497
 - horzElem, 498
 - Init, 498
 - MultiRayShape, 494
 - newLaserScans, 498
 - offset, 498
 - ProcessMsg, 498
 - rangeElem, 498
 - rayElem, 499
 - rays, 499
 - scanElem, 499
 - Update, 498
 - UpdateRays, 498
 - vertElem, 499
- gazebo::physics::PhysicsEngine, 531
 - ~PhysicsEngine, 533
 - contactManager, 541
 - CreateCollision, 534
 - CreateJoint, 534
 - CreateLink, 534
 - CreateShape, 534
 - DebugPrint, 535
 - Fini, 535
 - GetAutoDisableFlag, 535
 - GetContactManager, 535
 - GetContactMaxCorrectingVel, 535
 - GetContactSurfaceLayer, 535
 - GetGravity, 535
 - GetMaxContacts, 536
 - GetPhysicsUpdateMutex, 536
 - GetSORPGSIlters, 536
 - GetSORPGSPreconIlters, 536
 - GetSORPGSW, 536
 - GetStepTime, 537
 - GetUpdatePeriod, 537
 - GetUpdateRate, 537
 - GetWorldCFM, 537
 - GetWorldERP, 537
 - Init, 537
 - InitForThread, 538
 - Load, 538
 - node, 541
 - OnPhysicsMsg, 538
 - OnRequest, 538
 - PhysicsEngine, 533
 - physicsSub, 541
 - physicsUpdateMutex, 541
 - requestSub, 541
 - Reset, 538
 - responsePub, 541
 - sdf, 541
 - SetAutoDisableFlag, 538
 - SetContactMaxCorrectingVel, 538
 - SetContactSurfaceLayer, 539
 - SetGravity, 539
 - SetMaxContacts, 539
 - SetSORPGSIlters, 539
 - SetSORPGSPreconIlters, 539
 - SetSORPGSW, 540
 - SetStepTime, 540
 - SetUpdateRate, 540
 - SetWorldCFM, 540
 - SetWorldERP, 540
 - UpdateCollision, 541
 - UpdatePhysics, 541
 - world, 541
- gazebo::physics::PhysicsFactory, 542
 - NewPhysicsEngine, 542
 - RegisterAll, 542
 - RegisterPhysicsEngine, 542
- gazebo::physics::PlaneShape, 549
 - ~PlaneShape, 551
 - CreatePlane, 551
 - FillMsg, 551
 - GetNormal, 551
 - GetSize, 551
 - Init, 552
 - PlaneShape, 551
 - ProcessMsg, 552
 - SetAltitude, 552
 - SetNormal, 552
 - SetSize, 552
- gazebo::physics::RayShape, 603
 - ~RayShape, 606
 - contactFiducial, 608
 - contactLen, 608
 - contactRetro, 609

- FillMsg, 606
- GetFiducial, 606
- GetGlobalPoints, 606
- GetIntersection, 606
- GetLength, 607
- GetRelativePoints, 607
- GetRetro, 607
- globalEndPos, 609
- globalStartPos, 609
- Init, 607
- ProcessMsg, 607
- RayShape, 605, 606
- relativeEndPos, 609
- relativeStartPos, 609
- SetFiducial, 607
- SetLength, 608
- SetPoints, 608
- SetRetro, 608
- Update, 608
- gazebo::physics::Road, 621
 - ~Road, 622
 - Init, 623
 - Load, 623
 - Road, 622
- gazebo::physics::ScrewJoint
 - ~ScrewJoint, 647
 - fakeAnchor, 649
 - GetAnchor, 648
 - GetAngleCount, 648
 - Load, 648
 - ScrewJoint, 647
 - SetAnchor, 648
 - SetThreadPitch, 648
 - threadPitch, 649
- gazebo::physics::ScrewJoint< T >, 646
- gazebo::physics::Shape, 668
 - ~Shape, 669
 - collisionParent, 670
 - FillMsg, 669
 - FillShapeMsg, 669
 - GetInertial, 670
 - GetMass, 670
 - Init, 670
 - ProcessMsg, 670
 - Shape, 669
- gazebo::physics::SliderJoint
 - ~SliderJoint, 693
 - fakeAnchor, 694
 - GetAnchor, 693
 - GetAngleCount, 694
 - Load, 694
 - SetAnchor, 694
 - SliderJoint, 693
- gazebo::physics::SliderJoint< T >, 692
- gazebo::physics::SphereShape, 694
 - ~SphereShape, 696
 - FillMsg, 696
 - GetInertial, 696
 - GetMass, 696
 - GetRadius, 696
 - Init, 697
 - ProcessMsg, 697
 - SetRadius, 697
 - SphereShape, 696
- gazebo::physics::State, 702
 - ~State, 703
 - GetName, 703
 - GetRealTime, 704
 - GetSimTime, 704
 - GetWallTime, 704
 - Load, 704
 - name, 705
 - operator-, 704
 - operator=, 705
 - realTime, 705
 - SetName, 705
 - simTime, 705
 - State, 703
 - wallTime, 705
- gazebo::physics::SurfaceParams, 721
 - ~SurfaceParams, 722
 - bounce, 723
 - bounceThreshold, 723
 - cfm, 723
 - erp, 723
 - fdir1, 724
 - FillMsg, 723
 - FillSurfaceMsg, 723
 - kd, 724
 - kp, 724
 - Load, 723
 - maxVel, 724
 - minDepth, 724
 - mu1, 725
 - mu2, 725
 - ProcessMsg, 723
 - slip1, 725
 - slip2, 725
 - SurfaceParams, 722
- gazebo::physics::TrajectoryInfo, 760
 - duration, 761
 - endTime, 761
 - id, 761
 - startTime, 761
 - translated, 761
 - type, 761
- gazebo::physics::TrimeshShape, 761
 - ~TrimeshShape, 763

- FillMsg, 763
- GetFilename, 763
- GetMass, 763
- GetSize, 764
- Init, 764
- mesh, 765
- ProcessMsg, 764
- SetFilename, 764
- SetScale, 764
- TrimeshShape, 763
- Update, 764
- gazebo::physics::UniversalJoint
 - ~UniversalJoint, 766
 - GetAngleCount, 766
 - Load, 766
 - UniversalJoint, 766
- gazebo::physics::UniversalJoint< T >, 765
- gazebo::physics::World, 853
 - ~World, 856
 - Clear, 856
 - dirtyPoses, 864
 - DisableAllModels, 856
 - EnableAllModels, 856
 - EnablePhysicsEngine, 856
 - Fini, 857
 - GetByName, 857
 - GetEnablePhysicsEngine, 857
 - GetEntity, 857
 - GetEntityBelowPoint, 857
 - GetEntityByName, 858
 - GetModel, 858
 - GetModelBelowPoint, 858
 - GetModelByName, 859
 - GetModelCount, 859
 - GetModels, 859
 - GetName, 859
 - GetPauseTime, 859
 - GetPhysicsEngine, 859
 - GetRealTime, 859
 - GetSelectedEntity, 860
 - GetSetWorldPoseMutex, 860
 - GetSimTime, 860
 - GetStartTime, 860
 - Init, 860
 - InsertModelFile, 860
 - InsertModelSDF, 861
 - InsertModelString, 861
 - IsPaused, 861
 - Load, 861
 - LoadPlugin, 861
 - PrintEntityTree, 862
 - RemovePlugin, 862
 - Reset, 862
 - ResetEntities, 862
 - ResetTime, 862
 - Run, 862
 - Save, 862
 - SetPaused, 863
 - SetSimTime, 863
 - SetState, 863
 - StepWorld, 863
 - Stop, 863
 - StripWorldName, 863
 - UpdateStateSDF, 864
 - World, 856
- gazebo::physics::WorldState, 866
 - ~WorldState, 867
 - GetModelState, 868
 - GetModelStateCount, 868
 - GetModelStates, 868
 - HasModelState, 869
 - IsZero, 869
 - Load, 869
 - operator<<, 870
 - operator+, 869
 - operator-, 869
 - operator=, 870
 - WorldState, 867
- gazebo::rendering, 89
 - ArrowVisualPtr, 92
 - AxisVisualPtr, 92
 - COMVisualPtr, 92
 - CameraPtr, 92
 - CameraVisualPtr, 92
 - ContactVisualPtr, 92
 - DepthCameraPtr, 92
 - DynamicLinesPtr, 92
 - GpuLaserPtr, 92
 - JointVisualPtr, 92
 - LaserVisualPtr, 92
 - LightPtr, 92
 - RENDERING_LINE_LIST, 93
 - RENDERING_LINE_STRIP, 93
 - RENDERING_MESH_RESOURCE, 93
 - RENDERING_POINT_LIST, 93
 - RENDERING_TRIANGLE_FAN, 93
 - RENDERING_TRIANGLE_LIST, 93
 - RENDERING_TRIANGLE_STRIP, 93
 - RFIDTagVisualPtr, 92
 - RFIDVisualPtr, 92
 - RenderOpType, 93
 - ScenePtr, 92
 - UserCameraPtr, 92
 - VisualPtr, 92
- gazebo::rendering::ArrowVisual, 119
 - ~ArrowVisual, 120
 - ArrowVisual, 120
 - Load, 120

- ShowRotation, 120
- gazebo::rendering::AxisVisual, 120
 - ~AxisVisual, 122
 - AxisVisual, 122
 - Load, 122
 - ScaleXAxis, 122
 - ScaleYAxis, 122
 - ScaleZAxis, 122
 - SetAxisMaterial, 122
 - ShowRotation, 123
- gazebo::rendering::COMVisual, 204
 - ~COMVisual, 206
 - COMVisual, 205
 - Load, 206
- gazebo::rendering::Camera, 149
 - ~Camera, 155
 - animState, 171
 - AnimationComplete, 155
 - AttachToVisual, 155
 - AttachToVisualImpl, 155, 156
 - bayerFrameBuffer, 171
 - Camera, 155
 - camera, 171
 - captureData, 171
 - ConnectNewImageFrame, 156
 - connections, 171
 - CreateRenderTexture, 156
 - DisconnectNewImageFrame, 156
 - EnableSaveFrame, 157
 - Fini, 157
 - GetAspectRatio, 157
 - GetAvgFPS, 157
 - GetCameraToViewportRay, 157
 - GetDirection, 157
 - GetFarClip, 158
 - GetFrameFilename, 158
 - GetHFOV, 158
 - GetImageByteSize, 158
 - GetImageData, 159
 - GetImageDepth, 159
 - GetImageFormat, 159
 - GetImageHeight, 159
 - GetImageWidth, 159
 - GetInitialized, 159
 - GetLastRenderWallTime, 160
 - GetName, 160
 - GetNearClip, 160
 - GetOgreCamera, 160
 - GetRenderRate, 160
 - GetRenderTexture, 160
 - GetRight, 161
 - GetScene, 161
 - GetSceneNode, 161
 - GetTextureHeight, 161
 - GetTextureWidth, 161
 - GetTriangleCount, 161
 - GetUp, 162
 - GetVFOV, 162
 - GetViewport, 162
 - GetViewportHeight, 162
 - GetViewportWidth, 162
 - GetWindowId, 162
 - GetWorldPointOnPlane, 163
 - GetWorldPose, 163
 - GetWorldPosition, 163
 - GetWorldRotation, 163
 - GetZValue, 163
 - imageFormat, 171
 - imageHeight, 171
 - imageWidth, 171
 - Init, 164
 - initialized, 172
 - IsInitialized, 164
 - IsVisible, 164
 - lastRenderWallTime, 172
 - Load, 165
 - MoveToPosition, 165
 - MoveToPositions, 165
 - name, 172
 - newData, 172
 - newImageFrame, 172
 - onAnimationComplete, 172
 - pitchNode, 172
 - PostRender, 165
 - prevAnimTime, 172
 - Render, 166
 - RenderImpl, 166
 - renderTarget, 172
 - renderTexture, 172
 - requests, 172
 - RotatePitch, 166
 - RotateYaw, 166
 - saveCount, 173
 - SaveFrame, 166
 - saveFrameBuffer, 173
 - scene, 173
 - sceneNode, 173
 - sdf, 173
 - SetAspectRatio, 167
 - SetCaptureData, 167
 - SetClipDist, 167
 - SetHFOV, 167
 - SetImageHeight, 167
 - SetImageSize, 168
 - SetImageWidth, 168
 - SetName, 168
 - SetRenderRate, 168
 - SetRenderTarget, 168

- SetSaveFramePathname, 169
- SetScene, 169
- SetSceneNode, 169
- SetWindowId, 169
- SetWorldPose, 169
- SetWorldPosition, 169
- SetWorldRotation, 169
- ShowWireframe, 170
- textureHeight, 173
- textureWidth, 173
- ToggleShowWireframe, 170
- TrackVisual, 170
- TrackVisualImpl, 170
- Translate, 171
- Update, 171
- viewport, 173
- windowId, 173
- gazebo::rendering::CameraVisual, 177
 - ~CameraVisual, 179
 - CameraVisual, 178
 - Load, 179
- gazebo::rendering::ContactVisual, 228
 - ~ContactVisual, 230
 - ContactVisual, 229
 - SetEnabled, 230
- gazebo::rendering::Conversions, 230
 - Convert, 231, 232
- gazebo::rendering::DepthCamera, 238
 - ~DepthCamera, 240
 - ConnectNewDepthFrame, 240
 - ConnectNewRGBPointCloud, 240
 - CreateDepthTexture, 240
 - DepthCamera, 239
 - depthTarget, 242
 - depthTexture, 242
 - depthViewport, 242
 - DisconnectNewDepthFrame, 240
 - DisconnectNewRGBPointCloud, 241
 - Fini, 241
 - GetDepthData, 241
 - Init, 241
 - Load, 241
 - PostRender, 241
 - SetDepthTarget, 242
- gazebo::rendering::DynamicLines, 250
 - ~DynamicLines, 252
 - AddPoint, 252
 - Clear, 253
 - CreateVertexDeclaration, 253
 - DynamicLines, 252
 - FillHardwareBuffers, 253
 - GetMovableType, 253
 - getMovableType, 253
 - GetPoint, 253
 - GetPointCount, 253
 - SetPoint, 254
 - Update, 254
- gazebo::rendering::DynamicRenderable, 254
 - ~DynamicRenderable, 256
 - CreateVertexDeclaration, 256
 - DynamicRenderable, 256
 - FillHardwareBuffers, 256
 - getBoundingRadius, 256
 - GetOperationType, 257
 - getSquaredViewDepth, 257
 - indexBufferCapacity, 258
 - Init, 257
 - PrepareHardwareBuffers, 257
 - SetOperationType, 258
 - vertexBufferCapacity, 258
- gazebo::rendering::Events, 280
 - ConnectCreateScene, 280
 - ConnectRemoveScene, 281
 - ConnectViewContacts, 281
 - createScene, 282
 - DisconnectCreateScene, 281
 - DisconnectRemoveScene, 281
 - DisconnectViewContacts, 282
 - removeScene, 282
 - viewContacts, 282
- gazebo::rendering::FPSViewController, 305
 - ~FPSViewController, 306
 - FPSViewController, 306
 - GetTypeString, 306
 - HandleKeyPressEvent, 306
 - HandleKeyReleaseEvent, 307
 - HandleMouseEvent, 307
 - Init, 307
 - Update, 307
- gazebo::rendering::GUIOverlay, 335
 - ~GUIOverlay, 336
 - AttachCameraToImage, 336, 337
 - ButtonCallback, 337
 - CreateWindow, 337
 - GUIOverlay, 336
 - HandleKeyPressEvent, 337
 - HandleKeyReleaseEvent, 338
 - HandleMouseEvent, 338
 - Hide, 338
 - Init, 338
 - IsInitialized, 338
 - LoadLayout, 339
 - Resize, 339
 - Show, 339
 - Update, 339
- gazebo::rendering::GpuLaser, 313
 - ~GpuLaser, 314
 - ConnectNewLaserFrame, 314

- CreateLaserTexture, 314
- DisconnectNewLaserFrame, 315
- Finis, 315
- GetLaserData, 315
- GpuLaser, 314
- Init, 315
- Load, 315
- notifyRenderSingleObject, 315
- PostRender, 315
- SetParentSensor, 316
- SetRangeCount, 316
- gazebo::rendering::Grid, 330
 - ~Grid, 332
 - Enable, 332
 - GetCellCount, 332
 - GetCellLength, 332
 - GetColor, 332
 - GetHeight, 332
 - GetLineWidth, 332
 - GetSceneNode, 333
 - Grid, 331
 - Init, 333
 - SetCellCount, 333
 - SetCellLength, 333
 - SetColor, 333
 - SetHeight, 333
 - SetLineWidth, 334
 - SetUserData, 334
- gazebo::rendering::Heightmap, 339
 - ~Heightmap, 340
 - GetHeight, 340
 - Heightmap, 340
 - Load, 340
 - LoadFromMsg, 340
- gazebo::rendering::JointVisual, 385
 - ~JointVisual, 387
 - JointVisual, 386
 - Load, 387
- gazebo::rendering::LaserVisual, 390
 - ~LaserVisual, 391
 - LaserVisual, 391
 - SetEmissive, 391
- gazebo::rendering::Light, 391
 - ~Light, 393
 - FillMsg, 393
 - GetDiffuseColor, 393
 - GetDirection, 394
 - GetName, 394
 - GetPosition, 394
 - GetSpecularColor, 394
 - GetType, 394
 - Light, 393
 - Load, 394, 395
 - LoadFromMsg, 395
 - OnPoseChange, 395
 - SetAttenuation, 395
 - SetCastShadows, 395
 - SetDiffuseColor, 395
 - SetDirection, 396
 - SetLightType, 396
 - SetName, 396
 - SetPosition, 396
 - SetRange, 396
 - SetSelected, 396
 - SetSpecularColor, 397
 - SetSpotFalloff, 397
 - SetSpotInnerAngle, 397
 - SetSpotOuterAngle, 397
 - ShowVisual, 397
 - ToggleShowVisual, 397
 - UpdateFromMsg, 397
- gazebo::rendering::MovableText, 486
 - ~MovableText, 488
 - _setupGeometry, 488
 - _updateColors, 488
 - GetAABB, 488
 - GetBaseline, 488
 - getBoundingRadius, 488
 - GetCharHeight, 489
 - GetColor, 489
 - GetFont, 489
 - getLights, 489
 - getMaterial, 489
 - getRenderOperation, 489
 - GetShowOnTop, 489
 - GetSpaceWidth, 489
 - getSquaredViewDepth, 489
 - GetText, 489
 - getWorldTransforms, 490
 - H_CENTER, 488
 - H_LEFT, 488
 - HorizAlign, 487
 - Load, 490
 - MovableText, 488
 - SetBaseline, 490
 - SetCharHeight, 490
 - SetColor, 490
 - SetFontName, 490
 - SetShowOnTop, 491
 - SetSpaceWidth, 491
 - SetText, 491
 - SetTextAlignment, 491
 - Update, 491
 - V_ABOVE, 488
 - V_BELOW, 488
 - VertAlign, 488
 - visitRenderables, 491
- gazebo::rendering::OrbitViewController, 517

- ~OrbitViewController, 519
- GetFocalPoint, 519
- GetTypeString, 519
- HandleKeyPressEvent, 519
- HandleKeyReleaseEvent, 520
- HandleMouseEvent, 520
- Init, 520
- OrbitViewController, 519
- SetDistance, 520
- SetDistanceRange, 520
- SetFocalPoint, 521
- SetPitch, 521
- SetYaw, 521
- Update, 521
- gazebo::rendering::Projector, 569
 - ~Projector, 570
 - GetParent, 570
 - Load, 570
 - Projector, 570
 - SetEnabled, 571
 - SetTexture, 571
 - Toggle, 571
- gazebo::rendering::RFIDTagVisual, 618
 - ~RFIDTagVisual, 620
 - RFIDTagVisual, 619
- gazebo::rendering::RFIDVisual, 620
 - ~RFIDVisual, 621
 - RFIDVisual, 621
- gazebo::rendering::RTShaderSystem, 628
 - AddScene, 630
 - ApplyShadows, 630
 - AttachEntity, 630
 - AttachViewport, 630
 - Clear, 631
 - DetachEntity, 631
 - DetachViewport, 631
 - Fini, 631
 - GenerateShaders, 631
 - Init, 631
 - LightingModel, 630
 - RemoveScene, 631
 - RemoveShadows, 631
 - SSLM_NormalMapLightingObjectSpace, 630
 - SSLM_NormalMapLightingTangentSpace, 630
 - SSLM_PerPixelLighting, 630
 - SSLM_PerVertexLighting, 630
 - SetPerPixelLighting, 632
 - UpdateShaders, 632
- gazebo::rendering::RenderEngine, 609
 - AddResourcePath, 611
 - CreateScene, 611
 - DEFERRED, 611
 - dummyContext, 613
 - dummyDisplay, 613
 - dummyWindowId, 613
 - FORWARD, 611
 - Fini, 612
 - GetRenderPathType, 612
 - GetScene, 612
 - GetSceneCount, 612
 - Init, 613
 - Load, 613
 - NONE, 611
 - RENDER_PATH_COUNT, 611
 - RemoveScene, 613
 - RenderPathType, 611
 - root, 613
 - VERTEX, 611
- gazebo::rendering::Road2d, 623
 - ~Road2d, 624
 - Load, 624
 - Road2d, 624
- gazebo::rendering::Scene, 632
 - ~Scene, 635
 - AddVisual, 635
 - Clear, 636
 - CloneVisual, 636
 - CreateCamera, 636
 - CreateDepthCamera, 636
 - CreateGrid, 637
 - CreateUserCamera, 637
 - DrawLine, 637
 - GetAmbientColor, 637
 - GetBackgroundColor, 637
 - GetCamera, 638
 - GetCameraCount, 638
 - GetFirstContact, 638
 - GetGrid, 639
 - GetGridCount, 639
 - GetHeightBelowPoint, 639
 - GetHeightmap, 639
 - GetId, 640
 - GetIdString, 640
 - GetLight, 640
 - GetLightCount, 640
 - GetManager, 641
 - GetModelVisualAt, 641
 - GetName, 641
 - GetSelectedVisual, 641
 - GetShadowsEnabled, 641
 - GetUserCamera, 641
 - GetUserCameraCount, 642
 - GetVisual, 642
 - GetVisualAt, 642
 - GetVisualBelow, 643
 - GetVisualsBelowPoint, 643
 - GetWorldVisual, 643
 - Init, 643

- Load, 643
- PreRender, 644
- PrintSceneGraph, 644
- RemoveVisual, 644
- Scene, 635
- SelectVisual, 644
- SetAmbientColor, 644
- SetBackgroundColor, 644
- SetFog, 644
- SetGrid, 645
- SetShadowsEnabled, 645
- SetVisible, 645
- skyx, 646
- SnapVisualToNearestBelow, 645
- StripSceneName, 645
- ViewContacts, 646
- gazebo::rendering::SelectionObj, 650
 - ~SelectionObj, 651
 - Attach, 651
 - Clear, 651
 - GetVisualName, 651
 - Init, 652
 - IsActive, 652
 - SelectionObj, 651
 - SetActive, 652
 - SetHighlight, 652
- gazebo::rendering::UserCamera, 774
 - ~UserCamera, 777
 - AnimationComplete, 777
 - AttachToVisualImpl, 777
 - EnableViewController, 777
 - Fini, 778
 - GetAvgFPS, 778
 - GetGUIOverlay, 778
 - GetTriangleCount, 778
 - GetVisual, 778
 - HandleKeyPressEvent, 779
 - HandleKeyReleaseEvent, 779
 - HandleMouseEvent, 779
 - Init, 779
 - Load, 779
 - MoveToPosition, 779
 - MoveToVisual, 780
 - PostRender, 780
 - Resize, 780
 - SetFocalPoint, 780
 - SetRenderTarget, 781
 - SetViewController, 781
 - SetViewportDimensions, 781
 - SetWorldPose, 781
 - TrackVisualImpl, 782
 - Update, 782
 - UserCamera, 777
- gazebo::rendering::VideoVisual, 823
 - ~VideoVisual, 824
 - VideoVisual, 824
- gazebo::rendering::ViewController, 825
 - ~ViewController, 826
 - camera, 828
 - enabled, 828
 - GetTypeString, 826
 - HandleKeyPressEvent, 826
 - HandleKeyReleaseEvent, 827
 - HandleMouseEvent, 827
 - Init, 827
 - SetEnabled, 827
 - typeString, 828
 - Update, 828
 - ViewController, 826
- gazebo::rendering::Visual, 828
 - ~Visual, 834
 - AttachAxes, 834
 - AttachLineVertex, 834
 - AttachMesh, 834
 - AttachObject, 834
 - AttachVisual, 834
 - ClearParent, 835
 - Clone, 835
 - CreateDynamicLine, 835
 - DeleteDynamicLine, 835
 - DetachObjects, 835
 - DetachVisual, 835, 836
 - DisableTrackVisual, 836
 - EnableTrackVisual, 836
 - Fini, 836
 - GetAttachedObjectCount, 836
 - GetBoundingBox, 836
 - GetChild, 836
 - GetChildCount, 837
 - GetMaterialName, 837
 - GetName, 837
 - GetNormalMap, 837
 - GetParent, 837
 - GetPose, 837
 - GetPosition, 838
 - GetRootVisual, 838
 - GetRotation, 838
 - GetScale, 838
 - GetScene, 838
 - GetSceneNode, 838
 - GetShaderType, 839
 - GetTransparency, 839
 - GetVisibilityFlags, 839
 - GetVisible, 839
 - GetWorldPose, 839
 - HasAttachedObject, 840
 - Init, 840
 - InsertMesh, 840

- IsPlane, 840
- IsStatic, 840
- Load, 841
- LoadFromMsg, 841
- LoadPlugin, 841
- MakeStatic, 841
- MoveToPosition, 841
- MoveToPositions, 842
- parent, 847
- RemovePlugin, 842
- scene, 847
- sceneNode, 847
- SetAmbient, 842
- SetCastShadows, 842
- SetDiffuse, 842
- SetEmissive, 842
- SetHighlighted, 843
- SetMaterial, 843
- SetName, 843
- SetNormalMap, 843
- SetPose, 843
- SetPosition, 843
- SetRibbonTrail, 844
- SetRotation, 844
- SetScale, 844
- SetScene, 844
- SetShaderType, 844
- SetSkeletonPose, 845
- SetSpecular, 845
- SetTransparency, 845
- SetVisibilityFlags, 845
- SetVisible, 845
- SetWorldPose, 846
- SetWorldPosition, 846
- SetWorldRotation, 846
- ShowBoundingBox, 846
- ShowCOM, 846
- ShowCollision, 846
- ShowJoints, 846
- ShowSkeleton, 847
- ToggleVisible, 847
- Update, 847
- UpdateFromMsg, 847
- Visual, 833
- gazebo::rendering::WindowManager, 849
 - CreateWindow, 850
 - Fini, 850
 - GetAvgFPS, 850
 - GetTriangleCount, 851
 - GetWindow, 851
 - Moved, 851
 - Resize, 851
 - SetCamera, 851
- gazebo::rendering::WireBox, 852
 - ~WireBox, 852
 - Init, 853
 - SetVisible, 853
 - WireBox, 852
- gazebo::sensors, 93
 - CameraSensor_V, 95
 - CameraSensorPtr, 95
 - ContactSensor_V, 95
 - ContactSensorPtr, 95
 - DepthCameraSensor_V, 95
 - DepthCameraSensorPtr, 95
 - GpuRaySensor_V, 95
 - GpuRaySensorPtr, 95
 - RFIDSensor_V, 95
 - RFIDSensorPtr, 95
 - RFIDTag_V, 95
 - RFIDTagPtr, 95
 - RaySensor_V, 95
 - RaySensorPtr, 95
 - Sensor_V, 95
 - SensorFactoryFn, 95
 - SensorPtr, 95
- gazebo::sensors::CameraSensor, 173
 - ~CameraSensor, 175
 - CameraSensor, 175
 - Fini, 175
 - GetCamera, 175
 - GetImageData, 175
 - GetImageHeight, 175
 - GetImageWidth, 176
 - GetTopic, 176
 - Init, 176
 - Load, 176
 - SaveFrame, 177
 - SetParent, 177
 - UpdateImpl, 177
- gazebo::sensors::ContactSensor, 224
 - ~ContactSensor, 226
 - ContactSensor, 226
 - Fini, 226
 - GetCollisionContact, 226
 - GetCollisionContactCount, 226
 - GetCollisionCount, 226
 - GetCollisionName, 227
 - GetContacts, 227
 - Init, 227
 - IsActive, 227
 - Load, 227, 228
 - UpdateImpl, 228
- gazebo::sensors::DepthCameraSensor, 242
 - ~DepthCameraSensor, 244
 - DepthCameraSensor, 244
 - Fini, 244
 - GetDepthCamera, 244

- Init, 244
- Load, 244
- SaveFrame, 245
- SetActive, 245
- SetParent, 245
- UpdateImpl, 245
- gazebo::sensors::GpuRaySensor, 316
 - ~GpuRaySensor, 320
 - cameraCount, 328
 - cameraElem, 328
 - chfov, 329
 - ConnectNewLaserFrame, 320
 - cvfov, 329
 - DisconnectNewLaserFrame, 321
 - far, 329
 - Fini, 321
 - Get1stRatio, 321
 - Get2ndRatio, 321
 - GetAngleMax, 321
 - GetAngleMin, 321
 - GetAngleResolution, 321
 - GetCHFOV, 322
 - GetCVFOV, 322
 - GetCameraCount, 322
 - GetCosHorzFOV, 322
 - GetCosVertFOV, 322
 - GetFiducial, 322
 - GetHAngle, 323
 - GetHFOV, 323
 - GetHorzFOV, 323
 - GetHorzHalfAngle, 323
 - GetLaserCamera, 323
 - GetRange, 323
 - GetRangeCount, 324
 - GetRangeCountRatio, 324
 - GetRangeMax, 324
 - GetRangeMin, 324
 - GetRangeResolution, 324
 - GetRanges, 325
 - GetRayCount, 325
 - GetRayCountRatio, 325
 - GetRetro, 325
 - GetVAngle, 326
 - GetVFOV, 327
 - GetVertFOV, 326
 - GetVertHalfAngle, 326
 - GetVerticalAngleMax, 326
 - GetVerticalAngleMin, 326
 - GetVerticalRangeCount, 326
 - GetVerticalRayCount, 326
 - GpuRaySensor, 320
 - hfov, 329
 - horzElem, 329
 - horzHalfAngle, 329
 - horzRangeCount, 329
 - horzRayCount, 329
 - Init, 327
 - IsHorizontal, 327
 - isHorizontal, 329
 - Load, 327
 - near, 329
 - rangeCountRatio, 329
 - rangeElem, 330
 - rayCountRatio, 330
 - scanElem, 330
 - SetAngleMax, 327
 - SetAngleMin, 328
 - SetVerticalAngleMax, 328
 - SetVerticalAngleMin, 328
 - UpdateImpl, 328
 - vertElem, 330
 - vertHalfAngle, 330
 - vertRangeCount, 330
 - vertRayCount, 330
 - vfov, 330
- gazebo::sensors::ImuSensor, 354
 - ~ImuSensor, 355
 - Fini, 355
 - GetAngularVelocity, 355
 - GetLinearAcceleration, 356
 - ImuSensor, 355
 - Init, 356
 - Load, 356
 - UpdateImpl, 356
- gazebo::sensors::RFIDSensor, 614
 - ~RFIDSensor, 615
 - AddTag, 615
 - Fini, 615
 - Init, 615
 - Load, 615
 - RFIDSensor, 615
 - UpdateImpl, 616
- gazebo::sensors::RFIDTag, 616
 - ~RFIDTag, 617
 - Fini, 617
 - GetTagPose, 617
 - Init, 618
 - Load, 618
 - RFIDTag, 617
 - UpdateImpl, 618
- gazebo::sensors::RaySensor, 596
 - ~RaySensor, 598
 - Fini, 599
 - GetAngleMax, 599
 - GetAngleMin, 599
 - GetAngleResolution, 599
 - GetFiducial, 599
 - GetLaserShape, 599

- GetRange, 600
- GetRangeCount, 600
- GetRangeMax, 600
- GetRangeMin, 600
- GetRangeResolution, 600
- GetRanges, 601
- GetRayCount, 601
- GetRetro, 601
- GetTopic, 601
- GetVerticalAngleMax, 602
- GetVerticalAngleMin, 602
- GetVerticalRangeCount, 602
- GetVerticalRayCount, 602
- Init, 602
- Load, 602
- RaySensor, 598
- UpdateImpl, 603
- gazebo::sensors::Sensor, 652
 - ~Sensor, 655
 - active, 659
 - connections, 659
 - FillMsg, 655
 - Fini, 655
 - GetLastMeasurementTime, 655
 - GetLastUpdateTime, 655
 - GetName, 656
 - GetParentName, 656
 - GetPose, 656
 - GetScopedName, 656
 - GetTopic, 656
 - GetType, 656
 - GetVisualize, 657
 - GetWorldName, 657
 - Init, 657
 - IsActive, 657
 - lastMeasurementTime, 659
 - lastUpdateTime, 659
 - Load, 657, 658
 - node, 659
 - parentName, 659
 - plugins, 659
 - pose, 660
 - poseSub, 660
 - sdf, 660
 - Sensor, 655
 - SetActive, 658
 - SetParent, 658
 - SetUpdateRate, 658
 - Update, 658
 - UpdateImpl, 659
 - updatePeriod, 660
 - world, 660
- gazebo::sensors::SensorFactory, 660
 - GetSensorTypes, 661
 - NewSensor, 661
 - RegisterAll, 661
 - RegisterSensor, 661
- gazebo::sensors::SensorManager, 662
 - CreateSensor, 663
 - Fini, 663
 - GetSensor, 663
 - GetSensorTypes, 664
 - GetSensors, 664
 - Init, 664
 - RemoveSensor, 664
 - RemoveSensors, 664
 - Run, 664
 - SensorsInitialized, 664
 - Stop, 665
 - Update, 665
- gazebo::transport, 95
 - ConnectionPtr, 97
 - NodePtr, 97
 - PublicationPtr, 97
 - PublicationTransportPtr, 97
 - PublisherPtr, 97
 - SubscriberPtr, 97
 - SubscriptionTransportPtr, 97
- gazebo::transport::CallbackHelper, 144
 - ~CallbackHelper, 145
 - CallbackHelper, 145
 - GetLatching, 146
 - GetMsgType, 146
 - HandleData, 146
 - IsLocal, 146
 - latching, 147
- gazebo::transport::CallbackHelperT
 - CallbackHelperT, 148
 - GetMsgType, 148
 - HandleData, 148
 - IsLocal, 148
- gazebo::transport::CallbackHelperT< M >, 147
- gazebo::transport::Connection, 207
 - ~Connection, 210
 - AcceptCallback, 209
 - AsyncRead, 210
 - Cancel, 210
 - Connect, 210
 - ConnectToShutdown, 210
 - Connection, 210
 - DisconnectShutdown, 210
 - EnqueueMsg, 211
 - GetId, 211
 - GetLocalAddress, 211
 - GetLocalHostname, 211
 - GetLocalPort, 211
 - GetLocalURI, 211
 - GetRemoteAddress, 212

- GetRemoteHostname, 212
- GetRemotePort, 212
- GetRemoteURI, 212
- IsOpen, 212
- Listen, 212
- ProcessWriteQueue, 213
- Read, 213
- ReadCallback, 209
- Shutdown, 213
- StartRead, 213
- StopRead, 213
- gazebo::transport::ConnectionManager, 213
 - Advertise, 215
 - ConnectToRemoteHost, 215
 - eventConnections, 218
 - Fini, 215
 - GetAllPublishers, 215
 - GetTopicNamespaces, 216
 - Init, 216
 - IsRunning, 216
 - RegisterTopicNamespace, 216
 - RemoveConnection, 216
 - Run, 216
 - RunUpdate, 217
 - Stop, 217
 - Subscribe, 217
 - Unadvertise, 217
 - Unsubscribe, 217
- gazebo::transport::DebugCallbackHelper, 236
 - DebugCallbackHelper, 237
 - GetMsgType, 237
 - HandleData, 237
 - IsLocal, 237
- gazebo::transport::IOManager, 364
 - ~IOManager, 365
 - DecCount, 365
 - GetCount, 365
 - GetIO, 365
 - IOManager, 365
 - IncCount, 366
 - Stop, 366
- gazebo::transport::Node, 499
 - ~Node, 500
 - Advertise, 501
 - DecodeTopicName, 501
 - EncodeTopicName, 501
 - Fini, 501
 - GetId, 502
 - GetMsgType, 502
 - GetTopicNamespace, 502
 - HandleData, 502
 - Init, 502
 - InsertLatchedMsg, 503
 - Node, 500
 - ProcessIncoming, 503
 - ProcessPublishers, 503
 - Subscribe, 503
- gazebo::transport::Publication, 571
 - ~Publication, 572
 - AddPublisher, 573
 - AddSubscription, 573
 - AddTransport, 573
 - GetCallbackCount, 573
 - GetLocallyAdvertised, 573
 - GetMsgType, 574
 - GetNodeCount, 574
 - GetRemoteSubscriptionCount, 574
 - GetTransportCount, 574
 - HasTransport, 574
 - LocalPublish, 575
 - Publication, 572
 - Publish, 575
 - RemoveSubscription, 575
 - RemoveTransport, 575
 - SetLocallyAdvertised, 575
- gazebo::transport::PublicationTransport, 576
 - ~PublicationTransport, 577
 - AddCallback, 577
 - Fini, 577
 - GetConnection, 577
 - GetMsgType, 577
 - GetTopic, 577
 - Init, 578
 - PublicationTransport, 577
- gazebo::transport::Publisher, 578
 - ~Publisher, 579
 - GetLatching, 579
 - GetMsgType, 579
 - GetOutgoingCount, 579
 - GetPrevMsg, 580
 - GetTopic, 580
 - HasConnections, 580
 - Publish, 580
 - Publisher, 579
 - SendMessage, 581
 - SetPublication, 581
 - WaitForConnection, 581
- gazebo::transport::SubscribeOptions, 716
 - GetLatching, 717
 - GetMsgType, 717
 - GetNode, 717
 - GetTopic, 717
 - Init, 717
 - SubscribeOptions, 717
- gazebo::transport::Subscriber, 718
 - ~Subscriber, 718
 - GetTopic, 719
 - Subscriber, 718

- Unsubscribe, 719
- gazebo::transport::SubscriptionTransport, 719
 - ~SubscriptionTransport, 720
 - GetConnection, 720
 - HandleData, 720
 - Init, 721
 - IsLocal, 721
 - SubscriptionTransport, 720
- gazebo::transport::TopicManager, 754
 - AddNode, 756
 - Advertise, 756
 - ClearBuffers, 756
 - ConnectPubToSub, 757
 - ConnectSubToPub, 757
 - ConnectSubscribers, 757
 - DisconnectPubFromSub, 757
 - DisconnectSubFromPub, 757
 - FindPublication, 758
 - Fini, 758
 - GetTopicNamespaces, 758
 - Init, 758
 - IsAdvertised, 758
 - PauseIncoming, 758
 - ProcessNodes, 759
 - Publish, 759
 - RegisterTopicNamespace, 759
 - RemoveNode, 759
 - SubNodeMap, 756
 - Subscribe, 759
 - Unadvertise, 760
 - Unsubscribe, 760
 - UpdatePublications, 760
- gazebo_core.hh, 920
- gazebo_extensions_
 - urdf2gazebo::URDF2Gazebo, 774
- Gazebo_parser, 65
 - CollisionPtr, 65
 - VisualPtr, 65
- GazeboExtension
 - urdf2gazebo::GazeboExtension, 308
- GazeboGenerator
 - google::protobuf::compiler::cpp::GazeboGenerator, 312
- GazeboGenerator.hh, 920
- gazeboPathsFromEnv
 - gazebo::common::SystemPaths, 730
- GenSphericalTexCoord
 - gazebo::common::Mesh, 450
 - gazebo::common::MeshManager, 459
 - gazebo::common::SubMesh, 711
- Generate
 - google::protobuf::compiler::cpp::GazeboGenerator, 312
- GenerateShaders
 - gazebo::rendering::RTShaderSystem, 631
- GeneratorType
 - gazebo::math, 83
- Get
 - gazebo::common::NodeTransform, 511
 - sdf::Param, 524
- Get1stRatio
 - gazebo::sensors::GpuRaySensor, 321
- Get2ndRatio
 - gazebo::sensors::GpuRaySensor, 321
- get_master_uri
 - Transport, 71
- get_scene
 - Rendering, 63
- get_sensor
 - Sensors, 68
- get_topic_namespaces
 - Transport, 72
- get_world
 - Classes for physics and dynamics, 42
- GetAABB
 - gazebo::common::Mesh, 451
 - gazebo::rendering::MovableText, 488
- GetAbs
 - gazebo::math::Vector3, 804
- GetAcceleration
 - gazebo::physics::LinkState, 418
- GetAllLinks
 - gazebo::physics::Model, 465
- GetAllPublishers
 - gazebo::transport::ConnectionManager, 215
- GetAmbient
 - gazebo::common::Material, 431
- GetAmbientColor
 - gazebo::rendering::Scene, 637
- GetAnchor
 - gazebo::physics::Joint, 371
 - gazebo::physics::ScrewJoint, 648
 - gazebo::physics::SliderJoint, 693
- GetAngle
 - gazebo::physics::Joint, 372
 - gazebo::physics::JointState, 383
- GetAngleCount
 - gazebo::physics::BallJoint, 124
 - gazebo::physics::Hinge2Joint, 347
 - gazebo::physics::HingeJoint, 348
 - gazebo::physics::Joint, 372
 - gazebo::physics::JointState, 384
 - gazebo::physics::ScrewJoint, 648
 - gazebo::physics::SliderJoint, 694
 - gazebo::physics::UniversalJoint, 766
- GetAngleImpl
 - gazebo::physics::Joint, 372
- GetAngleMax

- gazebo::sensors::GpuRaySensor, 321
- gazebo::sensors::RaySensor, 599
- GetAngleMin
 - gazebo::sensors::GpuRaySensor, 321
 - gazebo::sensors::RaySensor, 599
- GetAngleResolution
 - gazebo::sensors::GpuRaySensor, 321
 - gazebo::sensors::RaySensor, 599
- GetAngles
 - gazebo::physics::JointState, 384
- GetAngularDamping
 - gazebo::physics::Link, 405
- GetAngularVelocity
 - gazebo::sensors::ImuSensor, 355
- GetAnimation
 - gazebo::common::Skeleton, 675
- GetAsABGR
 - gazebo::common::Color, 196
- GetAsARGB
 - gazebo::common::Color, 197
- GetAsAxis
 - gazebo::math::Quaternion, 586
- GetAsBGRA
 - gazebo::common::Color, 197
- GetAsDegree
 - gazebo::math::Angle, 109
- GetAsEuler
 - gazebo::math::Quaternion, 586
- GetAsHSV
 - gazebo::common::Color, 197
- GetAsMatrix3
 - gazebo::math::Quaternion, 586
- GetAsMatrix4
 - gazebo::math::Quaternion, 586
- GetAsPose
 - gazebo::math::Matrix4, 443
- GetAsRGBA
 - gazebo::common::Color, 197
- GetAsRadian
 - gazebo::math::Angle, 109
- GetAsString
 - sdf::Param, 524
 - sdf::ParamT, 529
- GetAsYUV
 - gazebo::common::Color, 197
- GetAspectRatio
 - gazebo::rendering::Camera, 157
- GetAttachedObjectCount
 - gazebo::rendering::Visual, 836
- GetAttribute
 - sdf::Element, 261, 262
- GetAttributeCount
 - sdf::Element, 262
- GetAttributeSet
 - sdf::Element, 262
- GetAutoDisable
 - gazebo::physics::Model, 465
- GetAutoDisableFlag
 - gazebo::physics::PhysicsEngine, 535
- GetAvgColor
 - gazebo::common::Image, 351
- GetAvgFPS
 - gazebo::rendering::Camera, 157
 - gazebo::rendering::UserCamera, 778
 - gazebo::rendering::WindowManager, 850
- GetBPP
 - gazebo::common::Image, 351
- GetBackgroundColor
 - gazebo::rendering::Scene, 637
- GetBaseline
 - gazebo::rendering::MovableText, 488
- GetBindShapeTransform
 - gazebo::common::Skeleton, 675
- GetBlendFactors
 - gazebo::common::Material, 431
- GetBlendMode
 - gazebo::common::Material, 431
- GetBoundingBox
 - gazebo::physics::Collision, 184
 - gazebo::physics::Entity, 269
 - gazebo::physics::Link, 405
 - gazebo::physics::Model, 465
 - gazebo::rendering::Visual, 836
- getBoundingBoxRadius
 - gazebo::rendering::DynamicRenderable, 256
 - gazebo::rendering::MovableText, 488
- GetById
 - gazebo::physics::Base, 129
- GetByName
 - gazebo::physics::Base, 129
 - gazebo::physics::World, 857
- GetCHFOV
 - gazebo::sensors::GpuRaySensor, 322
- GetCVFOV
 - gazebo::sensors::GpuRaySensor, 322
- GetCallbackCount
 - gazebo::transport::Publication, 573
- GetCamera
 - gazebo::rendering::Scene, 638
 - gazebo::sensors::CameraSensor, 175
- GetCameraCount
 - gazebo::rendering::Scene, 638
 - gazebo::sensors::GpuRaySensor, 322
- GetCameraToViewportRay
 - gazebo::rendering::Camera, 157
- GetCellCount
 - gazebo::rendering::Grid, 332
- GetCellLength

- gazebo::rendering::Grid, 332
- GetCenter
 - gazebo::math::Box, 137
- GetCharHeight
 - gazebo::rendering::MovableText, 489
- GetChild
 - gazebo::common::SkeletonNode, 686
 - gazebo::physics::Base, 130
 - gazebo::physics::Joint, 372
 - gazebo::rendering::Visual, 836
- GetChildById
 - gazebo::common::SkeletonNode, 686
- GetChildByName
 - gazebo::common::SkeletonNode, 687
- GetChildCollision
 - gazebo::physics::Entity, 269
- GetChildCount
 - gazebo::common::SkeletonNode, 687
 - gazebo::physics::Base, 130
 - gazebo::rendering::Visual, 837
- GetChildJointsLinks
 - gazebo::physics::Link, 405
- GetChildLink
 - gazebo::physics::Entity, 269
- GetCmd
 - gazebo::common::PID, 544
- GetCoG
 - gazebo::physics::Inertial, 359
- GetCollision
 - gazebo::physics::Link, 406
- GetCollisionBoundingBox
 - gazebo::physics::Entity, 270
- GetCollisionById
 - gazebo::physics::Link, 406
- GetCollisionContact
 - gazebo::sensors::ContactSensor, 226
- GetCollisionContactCount
 - gazebo::sensors::ContactSensor, 226
- GetCollisionCount
 - gazebo::sensors::ContactSensor, 226
- GetCollisionName
 - gazebo::sensors::ContactSensor, 227
- GetCollisionState
 - gazebo::physics::LinkState, 418
- GetCollisionStateCount
 - gazebo::physics::LinkState, 419
- GetCollisionStates
 - gazebo::physics::LinkState, 419
- GetCollisions
 - gazebo::physics::Link, 406
- GetColor
 - gazebo::rendering::Grid, 332
 - gazebo::rendering::MovableText, 489
- GetConnection
 - gazebo::transport::PublicationTransport, 577
 - gazebo::transport::SubscriptionTransport, 720
- GetContact
 - gazebo::physics::ContactManager, 223
- GetContactCount
 - gazebo::physics::ContactManager, 223
- GetContactManager
 - gazebo::physics::PhysicsEngine, 535
- GetContactMaxCorrectingVel
 - gazebo::physics::PhysicsEngine, 535
- GetContactSurfaceLayer
 - gazebo::physics::PhysicsEngine, 535
- GetContacts
 - gazebo::physics::ContactManager, 223
 - gazebo::sensors::ContactSensor, 227
- GetContactsEnabled
 - gazebo::physics::Collision, 184
- GetCopyChildren
 - sdf::Element, 262
- GetCosHorzFOV
 - gazebo::sensors::GpuRaySensor, 322
- GetCosVertFOV
 - gazebo::sensors::GpuRaySensor, 322
- GetCount
 - gazebo::transport::IOManager, 365
- GetCurrentDir
 - SystemPaths.hh, 1047
- GetData
 - gazebo::common::Image, 351
- GetDbfNormal
 - gazebo::math::Rand, 595
- GetDbfUniform
 - gazebo::math::Rand, 595
- GetDefaultAsString
 - sdf::Param, 524
 - sdf::ParamT, 529
- GetDefaultValue
 - sdf::ParamT, 529
- GetDepthCamera
 - gazebo::sensors::DepthCameraSensor, 244
- GetDepthData
 - gazebo::rendering::DepthCamera, 241
- GetDepthWrite
 - gazebo::common::Material, 432
- GetDescription
 - sdf::Element, 262
 - sdf::Param, 525
- GetDiffuse
 - gazebo::common::Material, 432
- GetDiffuseColor
 - gazebo::rendering::Light, 393
- GetDirection
 - gazebo::rendering::Camera, 157
 - gazebo::rendering::Light, 394

- GetDirtyPose
 - gazebo::physics::Entity, 270
- GetDistToLine
 - gazebo::math::Vector3, 804
- GetElapsed
 - gazebo::common::Timer, 754
- GetElement
 - sdf::Element, 262
- GetElementDescription
 - sdf::Element, 262
- GetElementDescriptionCount
 - sdf::Element, 262
- GetElementImpl
 - sdf::Element, 262
- GetEmissive
 - gazebo::common::Material, 432
- GetEnablePhysicsEngine
 - gazebo::physics::World, 857
- GetEnabled
 - gazebo::common::DiagnosticManager, 247
 - gazebo::physics::Link, 407
- GetEncoding
 - gazebo::common::LogRecord, 425
- GetEntity
 - gazebo::physics::World, 857
- GetEntityBelowPoint
 - gazebo::physics::World, 857
- GetEntityByName
 - gazebo::physics::World, 858
- GetErrorFile
 - gazebo::common::Exception, 304
- GetErrorStr
 - gazebo::common::Exception, 304
- GetErrors
 - gazebo::common::PID, 544
- GetEulerRotation
 - gazebo::math::Matrix4, 444
- GetExp
 - gazebo::math::Quaternion, 586
- GetFarClip
 - gazebo::rendering::Camera, 158
- GetFiducial
 - gazebo::physics::MultiRayShape, 495
 - gazebo::physics::RayShape, 606
 - gazebo::sensors::GpuRaySensor, 322
 - gazebo::sensors::RaySensor, 599
- GetFilename
 - gazebo::common::Image, 351
 - gazebo::physics::TrimeshShape, 763
 - gazebo::PluginT, 555
- GetFirstContact
 - gazebo::rendering::Scene, 638
- GetFirstElement
 - sdf::Element, 262
- GetFocalPoint
 - gazebo::rendering::OrbitViewController, 519
- GetFont
 - gazebo::rendering::MovableText, 489
- GetForce
 - gazebo::physics::Joint, 372
- GetFrameAt
 - gazebo::common::NodeAnimation, 505
- GetFrameCount
 - gazebo::common::NodeAnimation, 506
- GetFrameFilename
 - gazebo::rendering::Camera, 158
- GetGUIOverlay
 - gazebo::rendering::UserCamera, 778
- GetGazeboPaths
 - gazebo::common::SystemPaths, 729
- getGeometryBoundingBox
 - urdf2gazebo::URDF2Gazebo, 770
- GetGlobalAxis
 - gazebo::physics::Joint, 373
- GetGlobalPoints
 - gazebo::physics::RayShape, 606
- GetGravity
 - gazebo::physics::PhysicsEngine, 535
- GetGravityMode
 - gazebo::physics::Link, 407
- GetGrid
 - gazebo::rendering::Scene, 639
- GetGridCount
 - gazebo::rendering::Scene, 639
- GetHAngle
 - gazebo::sensors::GpuRaySensor, 323
- GetHFOV
 - gazebo::rendering::Camera, 158
 - gazebo::sensors::GpuRaySensor, 323
- GetHandle
 - gazebo::common::SkeletonNode, 687
 - gazebo::PluginT, 555
- GetHeader
 - Messages, 57
- GetHeight
 - gazebo::common::Image, 352
 - gazebo::common::Video, 822
 - gazebo::physics::HeightmapShape, 343
 - gazebo::rendering::Grid, 332
 - gazebo::rendering::Heightmap, 340
- GetHeightBelowPoint
 - gazebo::rendering::Scene, 639
- GetHeightmap
 - gazebo::rendering::Scene, 639
- GetHighStop
 - gazebo::physics::BallJoint, 124
 - gazebo::physics::Joint, 373
- GetHorzFOV

- gazebo::sensors::GpuRaySensor, 323
- GetHorzHalfAngle
 - gazebo::sensors::GpuRaySensor, 323
- GetIO
 - gazebo::transport::IOManager, 365
- GetIXX
 - gazebo::physics::Inertial, 359
- GetIXY
 - gazebo::physics::Inertial, 359
- GetIXZ
 - gazebo::physics::Inertial, 360
- GetIYY
 - gazebo::physics::Inertial, 360
- GetIYZ
 - gazebo::physics::Inertial, 360
- GetIZZ
 - gazebo::physics::Inertial, 360
- GetId
 - gazebo::common::SkeletonNode, 687
 - gazebo::event::Connection, 207
 - gazebo::physics::Base, 130
 - gazebo::rendering::Scene, 640
 - gazebo::transport::Connection, 211
 - gazebo::transport::Node, 502
- GetIdString
 - gazebo::rendering::Scene, 640
- GetImageByteSize
 - gazebo::rendering::Camera, 158
- GetImageData
 - gazebo::rendering::Camera, 159
 - gazebo::sensors::CameraSensor, 175
- GetImageDepth
 - gazebo::rendering::Camera, 159
- GetImageFormat
 - gazebo::rendering::Camera, 159
- GetImageHeight
 - gazebo::rendering::Camera, 159
 - gazebo::sensors::CameraSensor, 175
- GetImageWidth
 - gazebo::rendering::Camera, 159
 - gazebo::sensors::CameraSensor, 176
- GetInclude
 - sdf::Element, 262
- GetIndex
 - gazebo::common::SubMesh, 711
- GetIndexCount
 - gazebo::common::Mesh, 451
 - gazebo::common::SubMesh, 712
- GetInertial
 - gazebo::physics::BoxShape, 142
 - gazebo::physics::CylinderShape, 234
 - gazebo::physics::Link, 407
 - gazebo::physics::Shape, 670
 - gazebo::physics::SphereShape, 696
- GetInitialized
 - gazebo::rendering::Camera, 159
 - gazebo::Server, 667
- GetIntNormal
 - gazebo::math::Rand, 595
- GetIntUniform
 - gazebo::math::Rand, 596
- GetInterpolatedKeyFrame
 - gazebo::common::NumericAnimation, 515
 - gazebo::common::PoseAnimation, 566
- GetIntersection
 - gazebo::physics::RayShape, 606
- GetInverse
 - gazebo::math::Pose, 560
 - gazebo::math::Quaternion, 587
- GetInverseBindTransform
 - gazebo::common::SkeletonNode, 687
- GetJoint
 - gazebo::physics::Model, 466
- GetJointCount
 - gazebo::physics::Model, 466
- GetJointLink
 - gazebo::physics::Joint, 373
- GetJointState
 - gazebo::physics::ModelState, 479
- GetJointStateCount
 - gazebo::physics::ModelState, 480
- GetJointStates
 - gazebo::physics::ModelState, 480
- GetJoints
 - gazebo::physics::Model, 466
- GetKey
 - sdf::Param, 525
- GetKeyFrame
 - gazebo::common::Animation, 117
 - gazebo::common::NodeAnimation, 506
- GetKeyFrameCount
 - gazebo::common::Animation, 117
- GetKeyFramesAtTime
 - gazebo::common::Animation, 117
- getKeyValueAsString
 - urdf2gazebo::URDF2Gazebo, 771
- GetKinematic
 - gazebo::physics::Link, 407
- GetLabel
 - gazebo::common::DiagnosticManager, 247
- GetLaserCamera
 - gazebo::sensors::GpuRaySensor, 323
- GetLaserData
 - gazebo::rendering::GpuLaser, 315
- GetLaserRetro
 - gazebo::physics::Collision, 184
- GetLaserShape
 - gazebo::sensors::RaySensor, 599

- GetLastMeasurementTime
 - gazebo::sensors::Sensor, 655
- GetLastRenderWallTime
 - gazebo::rendering::Camera, 160
- GetLastUpdateTime
 - gazebo::sensors::Sensor, 655
- GetLatching
 - gazebo::transport::CallbackHelper, 146
 - gazebo::transport::Publisher, 579
 - gazebo::transport::SubscribeOptions, 717
- GetLength
 - gazebo::common::Animation, 117
 - gazebo::common::NodeAnimation, 506
 - gazebo::common::SkeletonAnimation, 680
 - gazebo::math::Vector3, 804
 - gazebo::math::Vector4, 815
 - gazebo::physics::CylinderShape, 234
 - gazebo::physics::RayShape, 607
- GetLight
 - gazebo::rendering::Scene, 640
- GetLightCount
 - gazebo::rendering::Scene, 640
- GetLighting
 - gazebo::common::Material, 432
- getLights
 - gazebo::rendering::MovableText, 489
- GetLineWidth
 - gazebo::rendering::Grid, 332
- GetLinearAcceleration
 - gazebo::sensors::ImuSensor, 356
- GetLinearDamping
 - gazebo::physics::Link, 407
- GetLink
 - gazebo::physics::Collision, 185
 - gazebo::physics::Model, 466, 467
- GetLinkById
 - gazebo::physics::Model, 467
- GetLinkForce
 - gazebo::physics::Joint, 373
- GetLinkState
 - gazebo::physics::ModelState, 480
- GetLinkStateCount
 - gazebo::physics::ModelState, 481
- GetLinkStates
 - gazebo::physics::ModelState, 481
- GetLinkTorque
 - gazebo::physics::Joint, 374
- GetLinks
 - gazebo::physics::Model, 467
- GetLocalAddress
 - gazebo::transport::Connection, 211
- GetLocalAxis
 - gazebo::physics::Joint, 374
- GetLocalHostname
 - gazebo::transport::Connection, 211
- GetLocalPort
 - gazebo::transport::Connection, 211
- GetLocalURI
 - gazebo::transport::Connection, 211
- GetLocallyAdvertised
 - gazebo::transport::Publication, 573
- GetLog
 - gazebo::math::Quaternion, 587
- GetLogPath
 - gazebo::common::SystemPaths, 729
- GetLowStop
 - gazebo::physics::BallJoint, 124
 - gazebo::physics::Joint, 374
- GetManager
 - gazebo::rendering::Scene, 641
- GetManifest
 - Common, 33
- GetMass
 - gazebo::physics::BoxShape, 142
 - gazebo::physics::CylinderShape, 235
 - gazebo::physics::Inertial, 360
 - gazebo::physics::Shape, 670
 - gazebo::physics::SphereShape, 696
 - gazebo::physics::TrimeshShape, 763
- GetMaterial
 - gazebo::common::Mesh, 451
- getMaterial
 - gazebo::rendering::MovableText, 489
- GetMaterialCount
 - gazebo::common::Mesh, 451
- GetMaterialIndex
 - gazebo::common::SubMesh, 712
- GetMaterialName
 - gazebo::rendering::Visual, 837
- GetMax
 - gazebo::common::Mesh, 451
 - gazebo::common::SubMesh, 712
 - gazebo::math::Vector3, 804
- GetMaxAngle
 - gazebo::physics::MultiRayShape, 495
- GetMaxColor
 - gazebo::common::Image, 352
- GetMaxContacts
 - gazebo::physics::PhysicsEngine, 536
- GetMaxForce
 - gazebo::physics::Joint, 375
- GetMaxHeight
 - gazebo::physics::HeightmapShape, 343
- GetMaxIndex
 - gazebo::common::SubMesh, 712
- GetMaxRange
 - gazebo::physics::MultiRayShape, 495
- GetMesh

- gazebo::common::MeshManager, 459
- GetMeshAABB
 - gazebo::common::MeshManager, 459
- GetMin
 - gazebo::common::Mesh, 451
 - gazebo::common::SubMesh, 712
 - gazebo::math::Vector3, 804
- GetMinAngle
 - gazebo::physics::MultiRayShape, 496
- GetMinHeight
 - gazebo::physics::HeightmapShape, 343
- GetMinRange
 - gazebo::physics::MultiRayShape, 496
- GetModel
 - gazebo::physics::Collision, 185
 - gazebo::physics::Link, 407
 - gazebo::physics::World, 858
- GetModelBelowPoint
 - gazebo::physics::World, 858
- GetModelByName
 - gazebo::physics::World, 859
- GetModelCount
 - gazebo::physics::World, 859
- GetModelFile
 - Common, 33
- GetModelName
 - Common, 34
- GetModelPath
 - Common, 34
- GetModelPaths
 - gazebo::common::SystemPaths, 729
- GetModelState
 - gazebo::physics::WorldState, 868
- GetModelStateCount
 - gazebo::physics::WorldState, 868
- GetModelStates
 - gazebo::physics::WorldState, 868
- GetModelTransform
 - gazebo::common::SkeletonNode, 687
- GetModelVisualAt
 - gazebo::rendering::Scene, 641
- GetModels
 - Common, 34
 - gazebo::physics::World, 859
- GetMovableType
 - gazebo::rendering::DynamicLines, 253
- getMovableType
 - gazebo::rendering::DynamicLines, 253
- GetMsgType
 - gazebo::transport::CallbackHelper, 146
 - gazebo::transport::CallbackHelperT, 148
 - gazebo::transport::DebugCallbackHelper, 237
 - gazebo::transport::Node, 502
 - gazebo::transport::Publication, 574
- gazebo::transport::PublicationTransport, 577
- gazebo::transport::Publisher, 579
- gazebo::transport::SubscribeOptions, 717
- GetName
 - gazebo::common::DiagnosticTimer, 250
 - gazebo::common::Material, 432
 - gazebo::common::Mesh, 452
 - gazebo::common::NodeAnimation, 507
 - gazebo::common::SkeletonAnimation, 681
 - gazebo::common::SkeletonNode, 688
 - gazebo::physics::Base, 130
 - gazebo::physics::State, 703
 - gazebo::physics::World, 859
 - gazebo::rendering::Camera, 160
 - gazebo::rendering::Light, 394
 - gazebo::rendering::Scene, 641
 - gazebo::rendering::Visual, 837
 - gazebo::sensors::Sensor, 656
 - sdf::Element, 262
- GetNearClip
 - gazebo::rendering::Camera, 160
- GetNearestEntityBelow
 - gazebo::physics::Entity, 270
- GetNextElement
 - sdf::Element, 263
- GetNextFrame
 - gazebo::common::Video, 822
- GetNode
 - gazebo::transport::SubscribeOptions, 717
- GetNodeAssignment
 - gazebo::common::SubMesh, 712
- GetNodeAssignmentsCount
 - gazebo::common::SubMesh, 712
- GetNodeByHandle
 - gazebo::common::Skeleton, 675
- GetNodeById
 - gazebo::common::Skeleton, 675
- GetNodeByName
 - gazebo::common::Skeleton, 676
- GetNodeCount
 - gazebo::common::SkeletonAnimation, 681
 - gazebo::transport::Publication, 574
- GetNodePoseAt
 - gazebo::common::SkeletonAnimation, 681
- GetNodes
 - gazebo::common::Skeleton, 676
- GetNormal
 - gazebo::common::SubMesh, 712
 - gazebo::math::Vector3, 805
 - gazebo::physics::PlaneShape, 551
- GetNormalCount
 - gazebo::common::Mesh, 452
 - gazebo::common::SubMesh, 713
- GetNormalMap

- gazebo::rendering::Visual, 837
- GetNumAnimations
 - gazebo::common::Skeleton, 676
- GetNumJoints
 - gazebo::common::Skeleton, 676
- GetNumNodes
 - gazebo::common::Skeleton, 676
- GetNumPoints
 - gazebo::math::RotationSpline, 626
- GetNumRawTrans
 - gazebo::common::SkeletonNode, 688
- GetNumVertNodeWeights
 - gazebo::common::Skeleton, 676
- GetOgreCamera
 - gazebo::rendering::Camera, 160
- GetOgrePaths
 - gazebo::common::SystemPaths, 729
- GetOperationType
 - gazebo::rendering::DynamicRenderable, 257
- GetOutgoingCount
 - gazebo::transport::Publisher, 579
- GetParent
 - gazebo::common::SkeletonNode, 688
 - gazebo::physics::Base, 131
 - gazebo::physics::Joint, 375
 - gazebo::rendering::Projector, 570
 - gazebo::rendering::Visual, 837
 - sdf::Element, 263
- GetParentId
 - gazebo::physics::Base, 131
- GetParentJointsLinks
 - gazebo::physics::Link, 408
- GetParentModel
 - gazebo::physics::Entity, 270
- GetParentName
 - gazebo::sensors::Sensor, 656
- GetPath
 - gazebo::common::Mesh, 452
- GetPauseTime
 - gazebo::physics::World, 859
- GetPerpendicular
 - gazebo::math::Vector3, 805
- GetPhysicsEngine
 - gazebo::physics::World, 859
- GetPhysicsUpdateMutex
 - gazebo::physics::PhysicsEngine, 536
- GetPitch
 - gazebo::common::Image, 352
 - gazebo::math::Quaternion, 587
- GetPixel
 - gazebo::common::Image, 352
- GetPixelFormat
 - gazebo::common::Image, 352
- GetPluginCount
 - gazebo::physics::Model, 467
- GetPluginPaths
 - gazebo::common::SystemPaths, 729
- GetPoint
 - gazebo::math::RotationSpline, 626
 - gazebo::math::Spline, 699
 - gazebo::rendering::DynamicLines, 253
- GetPointCount
 - gazebo::math::Spline, 699
 - gazebo::rendering::DynamicLines, 253
- GetPointSize
 - gazebo::common::Material, 432
- GetPos
 - gazebo::physics::HeightmapShape, 344
- GetPose
 - gazebo::physics::CollisionState, 191
 - gazebo::physics::Inertial, 360
 - gazebo::physics::LinkState, 419
 - gazebo::physics::ModelState, 481
 - gazebo::rendering::Visual, 837
 - gazebo::sensors::Sensor, 656
- GetPoseAt
 - gazebo::common::SkeletonAnimation, 681
- GetPoseAtX
 - gazebo::common::SkeletonAnimation, 682
- GetPosition
 - gazebo::rendering::Light, 394
 - gazebo::rendering::Visual, 838
- GetPrevMsg
 - gazebo::transport::Publisher, 580
- GetPrimitiveType
 - gazebo::common::SubMesh, 713
- GetPrincipalMoments
 - gazebo::physics::Inertial, 360
- GetProductsofInertia
 - gazebo::physics::Inertial, 361
- GetRGBData
 - gazebo::common::Image, 352
- GetRadius
 - gazebo::physics::CylinderShape, 235
 - gazebo::physics::SphereShape, 696
- GetRange
 - gazebo::physics::MultiRayShape, 496
 - gazebo::sensors::GpuRaySensor, 323
 - gazebo::sensors::RaySensor, 600
- GetRangeCount
 - gazebo::sensors::GpuRaySensor, 324
 - gazebo::sensors::RaySensor, 600
- GetRangeCountRatio
 - gazebo::sensors::GpuRaySensor, 324
- GetRangeMax
 - gazebo::sensors::GpuRaySensor, 324
 - gazebo::sensors::RaySensor, 600
- GetRangeMin

- gazebo::sensors::GpuRaySensor, 324
- gazebo::sensors::RaySensor, 600
- GetRangeResolution
 - gazebo::sensors::GpuRaySensor, 324
 - gazebo::sensors::RaySensor, 600
- GetRanges
 - gazebo::sensors::GpuRaySensor, 325
 - gazebo::sensors::RaySensor, 601
- GetRawTransform
 - gazebo::common::SkeletonNode, 688
- GetRawTransforms
 - gazebo::common::SkeletonNode, 688
- GetRayCount
 - gazebo::sensors::GpuRaySensor, 325
 - gazebo::sensors::RaySensor, 601
- GetRayCountRatio
 - gazebo::sensors::GpuRaySensor, 325
- GetRealTime
 - gazebo::physics::State, 704
 - gazebo::physics::World, 859
- GetRelativeAngularAccel
 - gazebo::physics::Collision, 185
 - gazebo::physics::Entity, 270
 - gazebo::physics::Link, 408
 - gazebo::physics::Model, 467
- GetRelativeAngularVel
 - gazebo::physics::Collision, 185
 - gazebo::physics::Entity, 271
 - gazebo::physics::Link, 408
 - gazebo::physics::Model, 467
- GetRelativeForce
 - gazebo::physics::Link, 408
- GetRelativeLinearAccel
 - gazebo::physics::Collision, 185
 - gazebo::physics::Entity, 271
 - gazebo::physics::Link, 408
 - gazebo::physics::Model, 468
- GetRelativeLinearVel
 - gazebo::physics::Collision, 185
 - gazebo::physics::Entity, 271
 - gazebo::physics::Link, 408
 - gazebo::physics::Model, 468
- GetRelativePoints
 - gazebo::physics::RayShape, 607
- GetRelativePose
 - gazebo::physics::Entity, 271
- GetRelativeTorque
 - gazebo::physics::Link, 409
- GetRemoteAddress
 - gazebo::transport::Connection, 212
- GetRemoteHostname
 - gazebo::transport::Connection, 212
- GetRemotePort
 - gazebo::transport::Connection, 212
- GetRemoteSubscriptionCount
 - gazebo::transport::Publication, 574
- GetRemoteURI
 - gazebo::transport::Connection, 212
- getRenderOperation
 - gazebo::rendering::MovableText, 489
- GetRenderPathType
 - gazebo::rendering::RenderEngine, 612
- GetRenderRate
 - gazebo::rendering::Camera, 160
- GetRenderTexture
 - gazebo::rendering::Camera, 160
- GetRequired
 - sdf::Element, 263
 - sdf::Param, 525
- GetResRange
 - gazebo::physics::MultiRayShape, 496
- GetRetro
 - gazebo::physics::MultiRayShape, 496
 - gazebo::physics::RayShape, 607
 - gazebo::sensors::GpuRaySensor, 325
 - gazebo::sensors::RaySensor, 601
- GetRight
 - gazebo::rendering::Camera, 161
- GetRoll
 - gazebo::math::Quaternion, 587
- GetRootNode
 - gazebo::common::Skeleton, 677
- GetRootVisual
 - gazebo::rendering::Visual, 838
- GetRotation
 - gazebo::common::PoseKeyFrame, 568
 - gazebo::math::Matrix4, 444
 - gazebo::rendering::Visual, 838
- GetRounded
 - gazebo::math::Vector3, 805
- GetSDF
 - gazebo::physics::Actor, 104
 - gazebo::physics::Base, 131
 - gazebo::physics::Model, 468
- GetSID
 - gazebo::common::NodeTransform, 511
- GetSORPGSIlters
 - gazebo::physics::PhysicsEngine, 536
- GetSORPGSPreconIlters
 - gazebo::physics::PhysicsEngine, 536
- GetSORPGSW
 - gazebo::physics::PhysicsEngine, 536
- GetSampleCount
 - gazebo::physics::MultiRayShape, 497
- GetSaveable
 - gazebo::physics::Base, 131
- GetScale
 - gazebo::rendering::Visual, 838

- GetScanResolution
 - gazebo::physics::MultiRayShape, 497
- GetScene
 - gazebo::rendering::Camera, 161
 - gazebo::rendering::RenderEngine, 612
 - gazebo::rendering::Visual, 838
- GetSceneCount
 - gazebo::rendering::RenderEngine, 612
- GetSceneNode
 - gazebo::rendering::Camera, 161
 - gazebo::rendering::Grid, 333
 - gazebo::rendering::Visual, 838
- GetScopedName
 - gazebo::physics::Base, 131
 - gazebo::sensors::Sensor, 656
- GetSeed
 - gazebo::math::Rand, 596
- GetSelectedEntity
 - gazebo::physics::World, 860
- GetSelectedVisual
 - gazebo::rendering::Scene, 641
- GetSelfCollide
 - gazebo::physics::Link, 409
- GetSensor
 - gazebo::sensors::SensorManager, 663
- GetSensorCount
 - gazebo::physics::Link, 409
 - gazebo::physics::Model, 468
- GetSensorName
 - gazebo::physics::Link, 409
- GetSensorTypes
 - gazebo::sensors::SensorFactory, 661
 - gazebo::sensors::SensorManager, 664
- GetSensors
 - gazebo::sensors::SensorManager, 664
- GetSet
 - sdf::Param, 525
- GetSetWorldPoseMutex
 - gazebo::physics::World, 860
- GetShadeMode
 - gazebo::common::Material, 433
- GetShaderType
 - gazebo::rendering::Visual, 839
- GetShadowsEnabled
 - gazebo::rendering::Scene, 641
- GetShape
 - gazebo::physics::Collision, 186
- GetShapeType
 - gazebo::physics::Collision, 186
- GetShininess
 - gazebo::common::Material, 433
- GetShowOnTop
 - gazebo::rendering::MovableText, 489
- GetSimTime
 - gazebo::physics::State, 704
 - gazebo::physics::World, 860
- GetSize
 - gazebo::math::Box, 137
 - gazebo::physics::BoxShape, 142
 - gazebo::physics::HeightmapShape, 344
 - gazebo::physics::PlaneShape, 551
 - gazebo::physics::TrimeshShape, 764
- GetSkeleton
 - gazebo::common::Mesh, 452
- GetSpaceWidth
 - gazebo::rendering::MovableText, 489
- GetSpecular
 - gazebo::common::Material, 433
- GetSpecularColor
 - gazebo::rendering::Light, 394
- GetSquaredLength
 - gazebo::math::Vector3, 805
 - gazebo::math::Vector4, 815
- getSquaredViewDepth
 - gazebo::rendering::DynamicRenderable, 257
 - gazebo::rendering::MovableText, 489
- GetStartTime
 - gazebo::physics::World, 860
- GetState
 - gazebo::physics::Collision, 186
- GetStepTime
 - gazebo::physics::PhysicsEngine, 537
- GetSubMesh
 - gazebo::common::Mesh, 452
- GetSubMeshCount
 - gazebo::common::Mesh, 453
- GetSubSampling
 - gazebo::physics::HeightmapShape, 344
- GetSum
 - gazebo::math::Vector3, 805
- GetSurface
 - gazebo::physics::Collision, 186
- GetTagPose
 - gazebo::sensors::RFIDTag, 617
- GetTangent
 - gazebo::math::Spline, 699
- GetTension
 - gazebo::math::Spline, 699
- GetTexCoord
 - gazebo::common::SubMesh, 713
- GetTexCoordCount
 - gazebo::common::Mesh, 453
 - gazebo::common::SubMesh, 713
- GetText
 - gazebo::rendering::MovableText, 489
- GetTextureHeight
 - gazebo::rendering::Camera, 161
- GetTextureImage

- gazebo::common::Material, 433
- GetTextureWidth
 - gazebo::rendering::Camera, 161
- GetTime
 - gazebo::common::Animation, 118
 - gazebo::common::DiagnosticManager, 247, 248
 - gazebo::common::KeyFrame, 390
- GetTimeAtX
 - gazebo::common::NodeAnimation, 507
- GetTimerCount
 - gazebo::common::DiagnosticManager, 248
- GetTopic
 - gazebo::sensors::CameraSensor, 176
 - gazebo::sensors::RaySensor, 601
 - gazebo::sensors::Sensor, 656
 - gazebo::transport::PublicationTransport, 577
 - gazebo::transport::Publisher, 580
 - gazebo::transport::SubscribeOptions, 717
 - gazebo::transport::Subscriber, 719
- GetTopicNamespace
 - gazebo::transport::Node, 502
- GetTopicNamespaces
 - gazebo::transport::ConnectionManager, 216
 - gazebo::transport::TopicManager, 758
- GetTransform
 - gazebo::common::SkeletonNode, 689
- GetTransforms
 - gazebo::common::SkeletonNode, 689
- GetTranslation
 - gazebo::common::PoseKeyFrame, 568
 - gazebo::math::Matrix4, 444
- GetTransparency
 - gazebo::common::Material, 433
 - gazebo::rendering::Visual, 839
- GetTransportCount
 - gazebo::transport::Publication, 574
- GetTriangleCount
 - gazebo::rendering::Camera, 161
 - gazebo::rendering::UserCamera, 778
 - gazebo::rendering::WindowManager, 851
- GetType
 - gazebo::common::NodeTransform, 511
 - gazebo::physics::Base, 131
 - gazebo::PluginT, 555
 - gazebo::rendering::Light, 394
 - gazebo::sensors::Sensor, 656
- GetTypeName
 - sdf::Param, 525
- GetTypeString
 - gazebo::rendering::FPSViewController, 306
 - gazebo::rendering::OrbitViewController, 519
 - gazebo::rendering::ViewController, 826
- GetURI
 - Common, 35
 - gazebo::physics::HeightmapShape, 344
- GetUp
 - gazebo::rendering::Camera, 162
- GetUpdatePeriod
 - gazebo::physics::PhysicsEngine, 537
- GetUpdateRate
 - gazebo::physics::PhysicsEngine, 537
- GetUserCamera
 - gazebo::rendering::Scene, 641
- GetUserCameraCount
 - gazebo::rendering::Scene, 642
- GetVAngle
 - gazebo::sensors::GpuRaySensor, 326
- GetVFOV
 - gazebo::rendering::Camera, 162
 - gazebo::sensors::GpuRaySensor, 327
- GetValue
 - gazebo::common::NumericKeyFrame, 517
 - sdf::Element, 263
 - sdf::ParamT, 529
- GetValueBool
 - sdf::Element, 263
- GetValueChar
 - sdf::Element, 263
- GetValueColor
 - sdf::Element, 263
- GetValueDouble
 - sdf::Element, 263
- GetValueFloat
 - sdf::Element, 263
- GetValueInt
 - sdf::Element, 263
- GetValuePose
 - sdf::Element, 263
- GetValueQuaternion
 - sdf::Element, 263
- GetValueString
 - sdf::Element, 263
- GetValueTime
 - sdf::Element, 263
- GetValueUInt
 - sdf::Element, 263
- GetValueVector2d
 - sdf::Element, 263
- GetValueVector3
 - sdf::Element, 263
- GetVelocity
 - gazebo::physics::Joint, 375
 - gazebo::physics::LinkState, 419
- GetVertFOV
 - gazebo::sensors::GpuRaySensor, 326
- GetVertHalfAngle
 - gazebo::sensors::GpuRaySensor, 326
- GetVertNodeWeight

- gazebo::common::Skeleton, 677
- GetVertex
 - gazebo::common::SubMesh, 713
- GetVertexCount
 - gazebo::common::Mesh, 453
 - gazebo::common::SubMesh, 713
 - gazebo::physics::HeightmapShape, 344
- GetVertexIndex
 - gazebo::common::SubMesh, 714
- GetVerticalAngleMax
 - gazebo::sensors::GpuRaySensor, 326
 - gazebo::sensors::RaySensor, 602
- GetVerticalAngleMin
 - gazebo::sensors::GpuRaySensor, 326
 - gazebo::sensors::RaySensor, 602
- GetVerticalMaxAngle
 - gazebo::physics::MultiRayShape, 497
- GetVerticalMinAngle
 - gazebo::physics::MultiRayShape, 497
- GetVerticalRangeCount
 - gazebo::sensors::GpuRaySensor, 326
 - gazebo::sensors::RaySensor, 602
- GetVerticalRayCount
 - gazebo::sensors::GpuRaySensor, 326
 - gazebo::sensors::RaySensor, 602
- GetVerticalSampleCount
 - gazebo::physics::MultiRayShape, 497
- GetVerticalScanResolution
 - gazebo::physics::MultiRayShape, 497
- GetViewport
 - gazebo::rendering::Camera, 162
- GetViewportHeight
 - gazebo::rendering::Camera, 162
- GetViewportWidth
 - gazebo::rendering::Camera, 162
- GetVisibilityFlags
 - gazebo::rendering::Visual, 839
- GetVisible
 - gazebo::rendering::Visual, 839
- GetVisual
 - gazebo::rendering::Scene, 642
 - gazebo::rendering::UserCamera, 778
- GetVisualAt
 - gazebo::rendering::Scene, 642
- GetVisualBelow
 - gazebo::rendering::Scene, 643
- GetVisualName
 - gazebo::rendering::SelectionObj, 651
- GetVisualize
 - gazebo::sensors::Sensor, 657
- GetVisualsBelowPoint
 - gazebo::rendering::Scene, 643
- GetWallTime
 - gazebo::common::Time, 737
- gazebo::physics::State, 704
- GetWidth
 - gazebo::common::Image, 353
 - gazebo::common::Video, 822
- GetWindow
 - gazebo::rendering::WindowManager, 851
- GetWindowId
 - gazebo::rendering::Camera, 162
- GetWorld
 - gazebo::physics::Base, 132
- GetWorldAngularAccel
 - gazebo::physics::Collision, 186
 - gazebo::physics::Entity, 271
 - gazebo::physics::Link, 410
 - gazebo::physics::Model, 468
- GetWorldAngularVel
 - gazebo::physics::Collision, 187
 - gazebo::physics::Entity, 272
 - gazebo::physics::Model, 469
- GetWorldCFM
 - gazebo::physics::PhysicsEngine, 537
- GetWorldERP
 - gazebo::physics::PhysicsEngine, 537
- GetWorldForce
 - gazebo::physics::Link, 410
- GetWorldLinearAccel
 - gazebo::physics::Collision, 187
 - gazebo::physics::Entity, 272
 - gazebo::physics::Link, 410
 - gazebo::physics::Model, 469
- GetWorldLinearVel
 - gazebo::physics::Collision, 187
 - gazebo::physics::Entity, 272
 - gazebo::physics::Model, 469
- GetWorldName
 - gazebo::sensors::Sensor, 657
- GetWorldPathExtension
 - gazebo::common::SystemPaths, 729
- GetWorldPointOnPlane
 - gazebo::rendering::Camera, 163
- GetWorldPose
 - gazebo::physics::Entity, 272
 - gazebo::rendering::Camera, 163
 - gazebo::rendering::Visual, 839
- GetWorldPosition
 - gazebo::rendering::Camera, 163
- GetWorldRotation
 - gazebo::rendering::Camera, 163
- GetWorldTorque
 - gazebo::physics::Link, 410
- getWorldTransforms
 - gazebo::rendering::MovableText, 490
- GetWorldVisual
 - gazebo::rendering::Scene, 643

- GetWrench
 - gazebo::physics::LinkState, 419
- GetXAxis
 - gazebo::math::Quaternion, 587
- GetXLength
 - gazebo::math::Box, 138
- GetYAxis
 - gazebo::math::Quaternion, 588
- GetYLength
 - gazebo::math::Box, 138
- GetYaw
 - gazebo::math::Quaternion, 587
- GetZAxis
 - gazebo::math::Quaternion, 588
- GetZLength
 - gazebo::math::Box, 138
- GetZValue
 - gazebo::rendering::Camera, 163
- globalEndPos
 - gazebo::physics::RayShape, 609
- globalStartPos
 - gazebo::physics::RayShape, 609
- google, 97
- google::protobuf, 97
- google::protobuf::compiler, 97
- google::protobuf::compiler::cpp, 98
- google::protobuf::compiler::cpp::GazeboGenerator, 312
 - ~GazeboGenerator, 312
 - GazeboGenerator, 312
 - Generate, 312
- GpuLaser
 - gazebo::rendering::GpuLaser, 314
- GpuLaser.hh, 921
- GpuLaserPtr
 - gazebo::rendering, 92
- GpuRaySensor
 - gazebo::sensors::GpuRaySensor, 320
- GpuRaySensor.hh, 922
- GpuRaySensor_V
 - gazebo::sensors, 95
- GpuRaySensorPtr
 - gazebo::sensors, 95
- gravity
 - urdf2gazebo::GazeboExtension, 309
- Green
 - gazebo::common::Color, 204
- Grid
 - gazebo::rendering::Grid, 331
- Grid.hh, 923
- Gripper
 - gazebo::physics::Gripper, 335
- Gripper.hh, 923
- gzclr_end
 - Common, 30
- gzclr_start
 - Common, 31
- gzdbg
 - Common, 31
- gzerr
 - Common, 31
- gzmsg
 - Common, 31
- gzthrow
 - Common, 31
- gzwarn
 - Common, 31
- H_CENTER
 - gazebo::rendering::MovableText, 488
- H_LEFT
 - gazebo::rendering::MovableText, 488
- HEADER_LENGTH
 - Connection.hh, 899
- HEIGHTMAP_SHAPE
 - gazebo::physics::Base, 128
- HI_STOP
 - gazebo::physics::Joint, 369
- HINGE2_JOINT
 - gazebo::physics::Base, 128
- HINGE_JOINT
 - gazebo::physics::Base, 128
- handle
 - gazebo::common::SkeletonNode, 691
 - gazebo::PluginT, 556
- HandleData
 - gazebo::transport::CallbackHelper, 146
 - gazebo::transport::CallbackHelperT, 148
 - gazebo::transport::DebugCallbackHelper, 237
 - gazebo::transport::Node, 502
 - gazebo::transport::SubscriptionTransport, 720
- HandleKeyPressEvent
 - gazebo::rendering::FPSViewController, 306
 - gazebo::rendering::GUIOverlay, 337
 - gazebo::rendering::OrbitViewController, 519
 - gazebo::rendering::UserCamera, 779
 - gazebo::rendering::ViewController, 826
- HandleKeyReleaseEvent
 - gazebo::rendering::FPSViewController, 307
 - gazebo::rendering::GUIOverlay, 338
 - gazebo::rendering::OrbitViewController, 520
 - gazebo::rendering::UserCamera, 779
 - gazebo::rendering::ViewController, 827
- HandleMouseEvent
 - gazebo::rendering::FPSViewController, 307
 - gazebo::rendering::GUIOverlay, 338
 - gazebo::rendering::OrbitViewController, 520
 - gazebo::rendering::UserCamera, 779
 - gazebo::rendering::ViewController, 827

- HasAttachedObject
 - gazebo::rendering::Visual, 840
- HasAttribute
 - sdf::Element, 263
- HasConnections
 - gazebo::transport::Publisher, 580
- HasElement
 - sdf::Element, 263
- HasElementDescription
 - sdf::Element, 263
- HasMesh
 - gazebo::common::MeshManager, 459
- HasModel
 - Common, 35
- HasModelState
 - gazebo::physics::WorldState, 869
- HasNode
 - gazebo::common::SkeletonAnimation, 682
- HasSkeleton
 - gazebo::common::Mesh, 453
- HasTransport
 - gazebo::transport::Publication, 574
- HasType
 - gazebo::physics::Base, 132
- HasVertex
 - gazebo::common::SubMesh, 714
- Heightmap
 - gazebo::rendering::Heightmap, 340
- Heightmap.hh, 925
- HeightmapShape
 - gazebo::physics::HeightmapShape, 343
- HeightmapShape.hh, 926
- HeightmapShapePtr
 - gazebo::physics, 88
- heights
 - gazebo::physics::HeightmapShape, 345
- Helpers.hh, 927
 - GZ_DBL_MAX, 929
 - GZ_DBL_MIN, 929
 - GZ_FLT_MAX, 929
 - GZ_FLT_MIN, 929
- hfov
 - gazebo::sensors::GpuRaySensor, 329
- Hide
 - gazebo::rendering::GUIOverlay, 338
- Hinge2Joint
 - gazebo::physics::Hinge2Joint, 346
- Hinge2Joint.hh, 929
- HingeJoint
 - gazebo::physics::HingeJoint, 348
- HingeJoint.hh, 931
- HorizAlign
 - gazebo::rendering::MovableText, 487
- horzElem
 - gazebo::physics::MultiRayShape, 498
 - gazebo::sensors::GpuRaySensor, 329
- horzHalfAngle
 - gazebo::sensors::GpuRaySensor, 329
- horzRangeCount
 - gazebo::sensors::GpuRaySensor, 329
- horzRayCount
 - gazebo::sensors::GpuRaySensor, 329
- IDENTITY
 - gazebo::math::Matrix4, 448
- IOManager
 - gazebo::transport::IOManager, 365
- IOManager.hh, 935
- id
 - gazebo::common::SkeletonNode, 691
 - gazebo::physics::TrajectoryInfo, 761
- Image
 - gazebo::common::Image, 351
- Image.hh, 932
- imageFormat
 - gazebo::rendering::Camera, 171
- imageHeight
 - gazebo::rendering::Camera, 171
- imageWidth
 - gazebo::rendering::Camera, 171
- img
 - gazebo::physics::HeightmapShape, 345
- ImuSensor
 - gazebo::sensors::ImuSensor, 355
- ImuSensor.hh, 933
- IncCount
 - gazebo::transport::IOManager, 366
- indexBufferCapacity
 - gazebo::rendering::DynamicRenderable, 258
- Inertial
 - gazebo::physics::Inertial, 359
- inertial
 - gazebo::physics::Link, 415
- Inertial.hh, 934
- InertialPtr
 - gazebo::physics, 89
- Init
 - gazebo::common::LogRecord, 425
 - gazebo::common::PID, 544
 - gazebo::Master, 427
 - gazebo::ModelPlugin, 476
 - gazebo::physics::Actor, 104
 - gazebo::physics::Base, 132
 - gazebo::physics::BoxShape, 143
 - gazebo::physics::Collision, 187
 - gazebo::physics::ContactManager, 223
 - gazebo::physics::CylinderShape, 235
 - gazebo::physics::Gripper, 335

- gazebo::physics::HeightmapShape, 344
- gazebo::physics::HingeJoint, 348
- gazebo::physics::Joint, 375
- gazebo::physics::Link, 410
- gazebo::physics::Model, 469
- gazebo::physics::MultiRayShape, 498
- gazebo::physics::PhysicsEngine, 537
- gazebo::physics::PlaneShape, 552
- gazebo::physics::RayShape, 607
- gazebo::physics::Road, 623
- gazebo::physics::Shape, 670
- gazebo::physics::SphereShape, 697
- gazebo::physics::TrimeshShape, 764
- gazebo::physics::World, 860
- gazebo::rendering::Camera, 164
- gazebo::rendering::DepthCamera, 241
- gazebo::rendering::DynamicRenderable, 257
- gazebo::rendering::FPSViewController, 307
- gazebo::rendering::GpuLaser, 315
- gazebo::rendering::Grid, 333
- gazebo::rendering::GUIOverlay, 338
- gazebo::rendering::OrbitViewController, 520
- gazebo::rendering::RenderEngine, 613
- gazebo::rendering::RTShaderSystem, 631
- gazebo::rendering::Scene, 643
- gazebo::rendering::SelectionObj, 652
- gazebo::rendering::UserCamera, 779
- gazebo::rendering::ViewController, 827
- gazebo::rendering::Visual, 840
- gazebo::rendering::WireBox, 853
- gazebo::SensorPlugin, 666
- gazebo::sensors::CameraSensor, 176
- gazebo::sensors::ContactSensor, 227
- gazebo::sensors::DepthCameraSensor, 244
- gazebo::sensors::GpuRaySensor, 327
- gazebo::sensors::ImuSensor, 356
- gazebo::sensors::RaySensor, 602
- gazebo::sensors::RFIDSensor, 615
- gazebo::sensors::RFIDTag, 618
- gazebo::sensors::Sensor, 657
- gazebo::sensors::SensorManager, 664
- gazebo::Server, 667
- gazebo::SystemPlugin, 731
- gazebo::transport::ConnectionManager, 216
- gazebo::transport::Node, 502
- gazebo::transport::PublicationTransport, 578
- gazebo::transport::SubscribeOptions, 717
- gazebo::transport::SubscriptionTransport, 721
- gazebo::transport::TopicManager, 758
- gazebo::VisualPlugin, 848
- gazebo::WorldPlugin, 865
- Messages, 57
- init
 - gazebo, 77
 - Rendering, 63
 - sdf, 99
 - Sensors, 68
 - Transport, 72
- init_world
 - Classes for physics and dynamics, 43
- init_worlds
 - Classes for physics and dynamics, 43
- initDoc
 - sdf, 99
- initFile
 - sdf, 99
- InitForThread
 - gazebo::physics::PhysicsEngine, 538
- initModelDoc
 - urdf2gazebo::URDF2Gazebo, 771
- initModelFile
 - urdf2gazebo::URDF2Gazebo, 771
- initModelString
 - urdf2gazebo::URDF2Gazebo, 771
- initString
 - sdf, 99
- initXml
 - sdf, 99
- initial_joint_position
 - urdf2gazebo::GazeboExtension, 309
- initialTransform
 - gazebo::common::SkeletonNode, 691
- initialized
 - gazebo::rendering::Camera, 172
- InsertElement
 - sdf::Element, 264
- insertGazeboExtensionCollision
 - urdf2gazebo::URDF2Gazebo, 771
- insertGazeboExtensionJoint
 - urdf2gazebo::URDF2Gazebo, 771
- insertGazeboExtensionLink
 - urdf2gazebo::URDF2Gazebo, 771
- insertGazeboExtensionRobot
 - urdf2gazebo::URDF2Gazebo, 771
- insertGazeboExtensionVisual
 - urdf2gazebo::URDF2Gazebo, 771
- InsertLatchedMsg
 - gazebo::transport::Node, 503
- InsertMesh
 - gazebo::rendering::Visual, 840
- InsertModelFile
 - gazebo::physics::World, 860
- InsertModelSDF
 - gazebo::physics::World, 861
- InsertModelString
 - gazebo::physics::World, 861
- Instance
 - Common, 35

- SingletonT, 672
- Interpolate
 - gazebo::math::RotationSpline, 626, 627
 - gazebo::math::Spline, 700
- interpolateX
 - gazebo::physics::Actor, 105
- invBindTransform
 - gazebo::common::SkeletonNode, 691
- Inverse
 - gazebo::math::Matrix4, 444
- inverseTransformToParentFrame
 - urdf2gazebo::URDF2Gazebo, 771
- Invert
 - gazebo::math::Quaternion, 588
- is_damping_factor
 - urdf2gazebo::GazeboExtension, 309
- is_fudge_factor
 - urdf2gazebo::GazeboExtension, 309
- is_initial_joint_position
 - urdf2gazebo::GazeboExtension, 309
- is_kd
 - urdf2gazebo::GazeboExtension, 309
- is_kp
 - urdf2gazebo::GazeboExtension, 309
- is_laser_retro
 - urdf2gazebo::GazeboExtension, 309
- is_maxVel
 - urdf2gazebo::GazeboExtension, 310
- is_minDepth
 - urdf2gazebo::GazeboExtension, 310
- is_mu1
 - urdf2gazebo::GazeboExtension, 310
- is_mu2
 - urdf2gazebo::GazeboExtension, 310
- is_stop_cfm
 - urdf2gazebo::GazeboExtension, 310
- is_stop_erp
 - urdf2gazebo::GazeboExtension, 310
- is_stopped
 - Transport, 72
- IsActive
 - gazebo::physics::Actor, 104
 - gazebo::rendering::SelectionObj, 652
 - gazebo::sensors::ContactSensor, 227
 - gazebo::sensors::Sensor, 657
- IsAdvertised
 - gazebo::transport::TopicManager, 758
- IsAffine
 - gazebo::math::Matrix4, 444
- IsBool
 - sdf::Param, 525
- IsCanonicalLink
 - gazebo::physics::Entity, 272
- IsChar
 - sdf::Param, 525
- IsColor
 - sdf::Param, 525
- IsDouble
 - sdf::Param, 525
- IsFinite
 - gazebo::math::Pose, 561
 - gazebo::math::Quaternion, 588
 - gazebo::math::Vector2d, 785
 - gazebo::math::Vector2i, 793
 - gazebo::math::Vector3, 806
 - gazebo::math::Vector4, 815
- IsFloat
 - sdf::Param, 525
- IsHorizontal
 - gazebo::sensors::GpuRaySensor, 327
- isHorizontal
 - gazebo::sensors::GpuRaySensor, 329
- IsInitialized
 - gazebo::rendering::Camera, 164
 - gazebo::rendering::GUIOverlay, 338
- IsInt
 - sdf::Param, 525
- IsJoint
 - gazebo::common::SkeletonNode, 689
- IsLocal
 - gazebo::transport::CallbackHelper, 146
 - gazebo::transport::CallbackHelperT, 148
 - gazebo::transport::DebugCallbackHelper, 237
 - gazebo::transport::SubscriptionTransport, 721
- IsOpen
 - gazebo::common::LogPlay, 422
 - gazebo::transport::Connection, 212
- IsPaused
 - gazebo::physics::World, 861
- IsPlaceable
 - gazebo::physics::Collision, 187
- IsPlane
 - gazebo::rendering::Visual, 840
- IsPose
 - sdf::Param, 525
- isPowerOfTwo
 - Math, 49
- IsQuaternion
 - sdf::Param, 525
- IsRootNode
 - gazebo::common::SkeletonNode, 689
- IsRunning
 - gazebo::transport::ConnectionManager, 216
- IsSelected
 - gazebo::physics::Base, 132
- IsStatic
 - gazebo::physics::Entity, 273
 - gazebo::rendering::Visual, 840

- IsStr
 - sdf::Param, 525
- IsTime
 - sdf::Param, 525
- IsUInt
 - sdf::Param, 525
- IsValidFilename
 - gazebo::common::MeshManager, 460
- IsVector2d
 - sdf::Param, 525
- IsVector2i
 - sdf::Param, 525
- IsVector3
 - sdf::Param, 525
- IsVisible
 - gazebo::rendering::Camera, 164
- IsZero
 - gazebo::physics::CollisionState, 192
 - gazebo::physics::JointState, 384
 - gazebo::physics::LinkState, 420
 - gazebo::physics::ModelState, 481
 - gazebo::physics::WorldState, 869
- isnan
 - Math, 48
- JOINT
 - gazebo::common::SkeletonNode, 685
 - gazebo::physics::Base, 128
- Joint
 - gazebo::physics::Joint, 370
- Joint.hh, 936
- Joint_V
 - gazebo::physics, 89
- JointController
 - gazebo::physics::JointController, 380
- JointController.hh, 937
- JointPtr
 - gazebo::physics, 89
- JointState
 - gazebo::physics::JointState, 383
- JointState.hh, 939
- JointVisual
 - gazebo::rendering::JointVisual, 386
- JointVisual.hh, 940
- JointVisualPtr
 - gazebo::rendering, 92
- JointWrench.hh, 940
- kd
 - gazebo::physics::SurfaceParams, 724
 - urdf2gazebo::GazeboExtension, 310
- key
 - sdf::Param, 527
- KeyFrame
 - gazebo::common::KeyFrame, 389
- KeyFrame.hh, 942
- KeyFrame_V
 - gazebo::common::Animation, 116
- keyFrames
 - gazebo::common::Animation, 118
 - gazebo::common::NodeAnimation, 507
- kp
 - gazebo::physics::SurfaceParams, 724
 - urdf2gazebo::GazeboExtension, 310
- L_INT16
 - gazebo::common::Image, 350
- L_INT8
 - gazebo::common::Image, 350
- LEFT
 - gazebo::common::MouseEvent, 484
- LIGHT
 - gazebo::physics::Base, 128
- LINE_MAX_LEN
 - STLLoader.hh, 1039
- LINES
 - gazebo::common::SubMesh, 709
- LINESTRIPS
 - gazebo::common::SubMesh, 709
- LINK
 - gazebo::physics::Base, 128
- LINUX
 - SystemPaths.hh, 1047
- LO_STOP
 - gazebo::physics::Joint, 369
- laser_retro
 - urdf2gazebo::GazeboExtension, 310
- LaserVisual
 - gazebo::rendering::LaserVisual, 391
- LaserVisual.hh, 943
- LaserVisualPtr
 - gazebo::rendering, 92
- lastMeasurementTime
 - gazebo::sensors::Sensor, 659
- lastPos
 - gazebo::physics::Actor, 105
- lastRenderWallTime
 - gazebo::rendering::Camera, 172
- lastScriptTime
 - gazebo::physics::Actor, 105
- lastTraj
 - gazebo::physics::Actor, 105
- lastUpdateTime
 - gazebo::sensors::Sensor, 659
- latching
 - gazebo::transport::CallbackHelper, 147
- length
 - gazebo::common::Animation, 118
 - gazebo::common::NodeAnimation, 507

- gazebo::common::SkeletonAnimation, 682
- Light
 - gazebo::rendering::Light, 393
- Light.hh, 944
- LightFromSDF
 - Messages, 58
- LightPtr
 - gazebo::rendering, 92
- LightingModel
 - gazebo::rendering::RTShaderSystem, 630
- linearAccel
 - gazebo::physics::Link, 415
- Link
 - gazebo::physics::Link, 402
- link
 - gazebo::physics::Collision, 189
- Link.hh, 944
- Link_V
 - gazebo::physics, 89
- LinkPtr
 - gazebo::physics, 89
- LinkState
 - gazebo::physics::LinkState, 417
- LinkState.hh, 946
- listGazeboExtensions
 - urdf2gazebo::URDF2Gazebo, 771
- Listen
 - gazebo::transport::Connection, 212
- Load
 - Common, 35
 - gazebo::common::BVHLoader, 144
 - gazebo::common::ColladaLoader, 180
 - gazebo::common::Image, 353
 - gazebo::common::MeshLoader, 455
 - gazebo::common::MeshManager, 460
 - gazebo::common::STLLoader, 706
 - gazebo::common::Video, 823
 - gazebo::ModelPlugin, 476
 - gazebo::physics::Actor, 104
 - gazebo::physics::BallJoint, 124
 - gazebo::physics::Base, 132
 - gazebo::physics::Collision, 187
 - gazebo::physics::CollisionState, 192
 - gazebo::physics::Entity, 273
 - gazebo::physics::Gripper, 335
 - gazebo::physics::HeightmapShape, 344
 - gazebo::physics::Hinge2Joint, 347
 - gazebo::physics::HingeJoint, 348
 - gazebo::physics::Inertial, 361
 - gazebo::physics::Joint, 375, 376
 - gazebo::physics::JointState, 384
 - gazebo::physics::Link, 410
 - gazebo::physics::LinkState, 420
 - gazebo::physics::Model, 469
 - gazebo::physics::ModelState, 481
 - gazebo::physics::PhysicsEngine, 538
 - gazebo::physics::Road, 623
 - gazebo::physics::ScrewJoint, 648
 - gazebo::physics::SliderJoint, 694
 - gazebo::physics::State, 704
 - gazebo::physics::SurfaceParams, 723
 - gazebo::physics::UniversalJoint, 766
 - gazebo::physics::World, 861
 - gazebo::physics::WorldState, 869
 - gazebo::rendering::ArrowVisual, 120
 - gazebo::rendering::AxisVisual, 122
 - gazebo::rendering::Camera, 165
 - gazebo::rendering::CameraVisual, 179
 - gazebo::rendering::COMVisual, 206
 - gazebo::rendering::DepthCamera, 241
 - gazebo::rendering::GpuLaser, 315
 - gazebo::rendering::Heightmap, 340
 - gazebo::rendering::JointVisual, 387
 - gazebo::rendering::Light, 394, 395
 - gazebo::rendering::MovableText, 490
 - gazebo::rendering::Projector, 570
 - gazebo::rendering::RenderEngine, 613
 - gazebo::rendering::Road2d, 624
 - gazebo::rendering::Scene, 643
 - gazebo::rendering::UserCamera, 779
 - gazebo::rendering::Visual, 841
 - gazebo::SensorPlugin, 666
 - gazebo::sensors::CameraSensor, 176
 - gazebo::sensors::ContactSensor, 227, 228
 - gazebo::sensors::DepthCameraSensor, 244
 - gazebo::sensors::GpuRaySensor, 327
 - gazebo::sensors::ImuSensor, 356
 - gazebo::sensors::RaySensor, 602
 - gazebo::sensors::RFIDSensor, 615
 - gazebo::sensors::RFIDTag, 618
 - gazebo::sensors::Sensor, 657, 658
 - gazebo::SystemPlugin, 731
 - gazebo::VisualPlugin, 849
 - gazebo::WorldPlugin, 865
- load
 - Classes for physics and dynamics, 43
 - gazebo, 77
 - Rendering, 63
 - Sensors, 69
- load_world
 - Classes for physics and dynamics, 43
- load_worlds
 - Classes for physics and dynamics, 43
- LoadFile
 - gazebo::Server, 667
- LoadFromMsg
 - gazebo::rendering::Heightmap, 340
 - gazebo::rendering::Light, 395

- gazebo::rendering::Visual, 841
- LoadLayout
 - gazebo::rendering::GUIOverlay, 339
- LoadPlugin
 - gazebo::physics::World, 861
 - gazebo::rendering::Visual, 841
- LoadPlugins
 - gazebo::physics::Model, 470
- LoadString
 - gazebo::Server, 667
- LocalPublish
 - gazebo::transport::Publication, 575
- LogPlay.hh, 947
- LogRecord.hh, 949
 - GZ_LOG_VERSION, 950
- Logplay, 423
- loop
 - gazebo::common::Animation, 118
 - gazebo::physics::Actor, 105
- m
 - gazebo::math::Matrix3, 441
 - gazebo::math::Matrix4, 448
- MAP_SHAPE
 - gazebo::physics::Base, 128
- MATRIX
 - gazebo::common::NodeTransform, 510
- MAX_COLLIDE_RETURNS
 - Contact.hh, 903
- MAX_CONTACT_JOINTS
 - Contact.hh, 903
- MIDDLE
 - gazebo::common::MouseEvent, 484
- MODEL
 - gazebo::physics::Base, 128
- MODEL_PLUGIN
 - Common, 32
- MODULATE
 - gazebo::common::Material, 430
- MOVE
 - gazebo::common::MouseEvent, 484
- MSleep
 - gazebo::common::Time, 737
- MULTIRAY_SHAPE
 - gazebo::physics::Base, 128
- mainLink
 - gazebo::physics::Actor, 106
- mainpage.html, 950
- MakeStatic
 - gazebo::rendering::Visual, 841
- MapShape.hh, 951
- Master
 - gazebo::Master, 427
- Master.hh, 951
- Material
 - gazebo::common::Material, 431
- material
 - urdf2gazebo::GazeboExtension, 310
- Material.hh, 953, 954
- Math, 46
 - clamp, 48
 - equal, 48
 - isPowerOfTwo, 49
 - isnan, 48
 - max, 49
 - mean, 49
 - min, 49
 - NAN_D, 51
 - NAN_I, 51
 - parseFloat, 50
 - parseInt, 50
 - precision, 50
 - variance, 50
- MathTypes.hh, 954
- Matrix3
 - gazebo::math::Matrix3, 438
- Matrix3.hh, 955
- Matrix4
 - gazebo::math::Matrix4, 443
- Matrix4.hh, 956
- max
 - gazebo::math::Box, 140
 - Math, 49
- maxVel
 - gazebo::physics::SurfaceParams, 724
 - urdf2gazebo::GazeboExtension, 310
- mean
 - Math, 49
- Merge
 - gazebo::math::Box, 138
- Mesh
 - gazebo::common::Mesh, 450
- mesh
 - gazebo::physics::Actor, 106
 - gazebo::physics::TrimeshShape, 765
- Mesh.hh, 957
- MeshLoader
 - gazebo::common::MeshLoader, 455
- MeshLoader.hh, 958
- MeshManager.hh, 960
- MeshShapePtr
 - gazebo::physics, 89
- Messages, 52
 - Convert, 53–56
 - CreateRequest, 56
 - FogFromSDF, 57
 - GUIFromSDF, 57
 - GetHeader, 57

- Init, 57
- LightFromSDF, 58
- SceneFromSDF, 58
- Set, 58–60
- Stamp, 60
- TrackVisualFromSDF, 60
- VisualFromSDF, 60
- MicToNano
 - gazebo::common::Time, 737
- MilToNano
 - gazebo::common::Time, 737
- min
 - gazebo::math::Box, 140
 - Math, 49
- minDepth
 - gazebo::physics::SurfaceParams, 724
 - urdf2gazebo::GazeboExtension, 311
- Model
 - gazebo::physics::Model, 464
- model
 - gazebo::physics::Joint, 379
- Model.hh, 961
- Model_V
 - gazebo::physics, 89
- ModelDatabase.hh, 963
- modelPathsFromEnv
 - gazebo::common::SystemPaths, 730
- ModelPlugin
 - gazebo::ModelPlugin, 476
- ModelPluginPtr
 - gazebo, 77
- ModelPtr
 - gazebo::physics, 89
- ModelState
 - gazebo::physics::ModelState, 478, 479
- ModelState.hh, 963
- modelTransform
 - gazebo::common::SkeletonNode, 691
- MouseEvent
 - gazebo::common::MouseEvent, 484
- MouseEvent.hh, 965
- MovableText
 - gazebo::rendering::MovableText, 488
- MovableText.hh, 966
- moveScale
 - gazebo::common::MouseEvent, 485
- MoveToPosition
 - gazebo::rendering::Camera, 165
 - gazebo::rendering::UserCamera, 779
 - gazebo::rendering::Visual, 841
- MoveToPositions
 - gazebo::rendering::Camera, 165
 - gazebo::rendering::Visual, 842
- MoveToVisual
 - gazebo::rendering::UserCamera, 780
- Moved
 - gazebo::rendering::WindowManager, 851
- msgs.hh, 967
- mu1
 - gazebo::physics::SurfaceParams, 725
 - urdf2gazebo::GazeboExtension, 311
- mu2
 - gazebo::physics::SurfaceParams, 725
 - urdf2gazebo::GazeboExtension, 311
- MultiRayShape
 - gazebo::physics::MultiRayShape, 494
- MultiRayShape.hh, 969
- MultiRayShapePtr
 - gazebo::physics, 89
- NAN_D
 - Math, 51
- NAN_I
 - Math, 51
- NO_BUTTON
 - gazebo::common::MouseEvent, 484
- NO_EVENT
 - gazebo::common::MouseEvent, 484
- NODE
 - gazebo::common::SkeletonNode, 685
- NONE
 - gazebo::rendering::RenderEngine, 611
- NRealGen
 - gazebo::math, 83
- NSleep
 - gazebo::common::Time, 738
- NULL
 - CommonTypes.hh, 896
- name
 - gazebo::common::Animation, 119
 - gazebo::common::Material, 436
 - gazebo::common::NodeAnimation, 507
 - gazebo::common::SkeletonAnimation, 683
 - gazebo::common::SkeletonNode, 691
 - gazebo::physics::State, 705
 - gazebo::rendering::Camera, 172
 - sdf::Plugin, 554
- near
 - gazebo::sensors::GpuRaySensor, 329
- NewContact
 - gazebo::physics::ContactManager, 224
- newData
 - gazebo::rendering::Camera, 172
- newImageFrame
 - gazebo::rendering::Camera, 172
- newLaserScans
 - gazebo::physics::MultiRayShape, 498
- NewPhysicsEngine

- gazebo::physics::PhysicsFactory, 542
- NewSensor
 - gazebo::sensors::SensorFactory, 661
- Node
 - gazebo::transport::Node, 500
- node
 - gazebo::physics::Entity, 276
 - gazebo::physics::PhysicsEngine, 541
 - gazebo::sensors::Sensor, 659
- Node.hh, 971
- NodeAnimation
 - gazebo::common::NodeAnimation, 505
- nodeIndex
 - gazebo::common::NodeAssignment, 508
- NodeMap
 - gazebo::common, 80
- NodeMapIter
 - gazebo::common, 80
- NodePtr
 - gazebo::transport, 97
- NodeTransform
 - gazebo::common::NodeTransform, 510
- nodes
 - gazebo::common::Skeleton, 678
- normal
 - gazebo::math::Plane, 549
- NormalRealDist
 - gazebo::math, 83
- Normalize
 - gazebo::math::Angle, 110
 - gazebo::math::Quaternion, 588
 - gazebo::math::Vector2d, 785
 - gazebo::math::Vector2i, 793
 - gazebo::math::Vector3, 806
 - gazebo::math::Vector4, 815
- normals
 - gazebo::physics::Contact, 221
- notifyRenderSingleObject
 - gazebo::rendering::GpuLaser, 315
- nsec
 - gazebo::common::Time, 752
- NullStream
 - Common, 32
- NumericAnimation
 - gazebo::common::NumericAnimation, 514
- NumericAnimationPtr
 - gazebo::common, 80
- NumericKeyFrame
 - gazebo::common::NumericKeyFrame, 516
- ORDER_MAX
 - STLLoader.hh, 1039
- offset
 - gazebo::physics::MultiRayShape, 498
- Ogre, 98
- ogre, 98
- ogre_gazebo.h, 972
- ogrePathsFromEnv
 - gazebo::common::SystemPaths, 730
- old_link_name
 - urdf2gazebo::GazeboExtension, 311
- oldAction
 - gazebo::physics::Actor, 106
- onAnimationComplete
 - gazebo::rendering::Camera, 172
- OnPhysicsMsg
 - gazebo::physics::PhysicsEngine, 538
- OnPoseChange
 - gazebo::physics::Entity, 273
 - gazebo::physics::Link, 411
 - gazebo::physics::Model, 470
 - gazebo::rendering::Light, 395
- OnRequest
 - gazebo::physics::PhysicsEngine, 538
- Open
 - gazebo::common::LogPlay, 422
- operator<
 - gazebo::common::Time, 745, 746
 - gazebo::math::Angle, 112
- operator<<
 - gazebo::common::Color, 203
 - gazebo::common::Exception, 305
 - gazebo::common::Material, 436
 - gazebo::common::Time, 751
 - gazebo::common::Timer, 754
 - gazebo::math::Angle, 114
 - gazebo::math::Box, 140
 - gazebo::math::Matrix3, 440
 - gazebo::math::Matrix4, 447
 - gazebo::math::Pose, 563
 - gazebo::math::Quaternion, 593
 - gazebo::math::Vector2d, 790
 - gazebo::math::Vector2i, 798
 - gazebo::math::Vector3, 811
 - gazebo::math::Vector4, 820
 - gazebo::physics::CollisionState, 193
 - gazebo::physics::Inertial, 364
 - gazebo::physics::JointState, 385
 - gazebo::physics::LinkState, 421
 - gazebo::physics::ModelState, 482
 - gazebo::physics::WorldState, 870
 - sdf::ParamT, 530
- operator<=
 - gazebo::common::Time, 746, 747
 - gazebo::math::Angle, 112
- operator>
 - gazebo::common::Time, 749
 - gazebo::math::Angle, 113

- operator>>
 - gazebo::common::Color, 203
 - gazebo::common::Time, 752
 - gazebo::math::Angle, 114
 - gazebo::math::Pose, 564
 - gazebo::math::Quaternion, 594
 - gazebo::math::Vector2d, 790
 - gazebo::math::Vector2i, 799
 - gazebo::math::Vector3, 811
 - gazebo::math::Vector4, 821
- operator>=
 - gazebo::common::Time, 750
 - gazebo::math::Angle, 113
- operator*
 - gazebo::common::Color, 198
 - gazebo::common::NodeTransform, 511, 512
 - gazebo::common::Time, 739
 - gazebo::math::Angle, 110
 - gazebo::math::Matrix4, 444, 445
 - gazebo::math::Pose, 561
 - gazebo::math::Quaternion, 589
 - gazebo::math::Vector2d, 785
 - gazebo::math::Vector2i, 794
 - gazebo::math::Vector3, 806
 - gazebo::math::Vector4, 816
 - sdf::ParamT, 529
- operator*=
 - gazebo::common::Color, 198
 - gazebo::common::Time, 740
 - gazebo::math::Angle, 110
 - gazebo::math::Quaternion, 589
 - gazebo::math::Vector2d, 786
 - gazebo::math::Vector2i, 794, 795
 - gazebo::math::Vector3, 807
 - gazebo::math::Vector4, 816, 817
- operator()
 - gazebo::common::NodeTransform, 511
 - gazebo::event::EventT, 297–299
- operator+
 - gazebo::common::Color, 198, 199
 - gazebo::common::Time, 740, 741
 - gazebo::math::Angle, 111
 - gazebo::math::Box, 138
 - gazebo::math::Pose, 561
 - gazebo::math::Quaternion, 589
 - gazebo::math::Vector2d, 786
 - gazebo::math::Vector2i, 795
 - gazebo::math::Vector3, 807
 - gazebo::math::Vector4, 817
 - gazebo::physics::CollisionState, 192
 - gazebo::physics::Inertial, 361
 - gazebo::physics::JointState, 384
 - gazebo::physics::LinkState, 420
 - gazebo::physics::ModelState, 481
- gazebo::physics::WorldState, 869
- operator+=
 - gazebo::common::Color, 199
 - gazebo::common::Time, 741, 742
 - gazebo::math::Angle, 111
 - gazebo::math::Box, 139
 - gazebo::math::Pose, 561
 - gazebo::math::Quaternion, 590
 - gazebo::math::Vector2d, 787
 - gazebo::math::Vector2i, 795
 - gazebo::math::Vector3, 807
 - gazebo::math::Vector4, 817
 - gazebo::physics::Inertial, 361
- operator-
 - gazebo::common::Color, 199
 - gazebo::common::Time, 742
 - gazebo::math::Angle, 111
 - gazebo::math::Box, 139
 - gazebo::math::Pose, 562
 - gazebo::math::Quaternion, 590
 - gazebo::math::Vector2d, 787
 - gazebo::math::Vector2i, 795
 - gazebo::math::Vector3, 808
 - gazebo::math::Vector4, 817
 - gazebo::physics::CollisionState, 192
 - gazebo::physics::JointState, 384
 - gazebo::physics::LinkState, 420
 - gazebo::physics::ModelState, 482
 - gazebo::physics::State, 704
 - gazebo::physics::WorldState, 869
- operator-=
 - gazebo::common::Color, 200
 - gazebo::common::Time, 743
 - gazebo::math::Angle, 111
 - gazebo::math::Pose, 562
 - gazebo::math::Quaternion, 590
 - gazebo::math::Vector2d, 787
 - gazebo::math::Vector2i, 796
 - gazebo::math::Vector3, 808
 - gazebo::math::Vector4, 818
- operator/
 - gazebo::common::Color, 200
 - gazebo::common::Time, 743, 744
 - gazebo::math::Angle, 112
 - gazebo::math::Vector2d, 787, 788
 - gazebo::math::Vector2i, 796
 - gazebo::math::Vector3, 808
 - gazebo::math::Vector4, 818
- operator/=
 - gazebo::common::Color, 200
 - gazebo::common::Time, 744, 745
 - gazebo::math::Angle, 112
 - gazebo::math::Vector2d, 788
 - gazebo::math::Vector2i, 796, 797

- gazebo::math::Vector3, 809
- gazebo::math::Vector4, 819
- operator=
 - gazebo::common::Color, 201
 - gazebo::common::PID, 545
 - gazebo::common::Time, 747
 - gazebo::math::Box, 139
 - gazebo::math::Matrix4, 445
 - gazebo::math::Plane, 548
 - gazebo::math::Quaternion, 591
 - gazebo::math::Vector2d, 788, 789
 - gazebo::math::Vector2i, 797
 - gazebo::math::Vector3, 809
 - gazebo::math::Vector4, 819
 - gazebo::physics::CollisionState, 192
 - gazebo::physics::Contact, 220, 221
 - gazebo::physics::Inertial, 361
 - gazebo::physics::JointState, 385
 - gazebo::physics::JointWrench, 388
 - gazebo::physics::LinkState, 421
 - gazebo::physics::ModelState, 482
 - gazebo::physics::State, 705
 - gazebo::physics::WorldState, 870
- operator==
 - gazebo::common::Color, 201
 - gazebo::common::Time, 748
 - gazebo::math::Angle, 113
 - gazebo::math::Box, 139
 - gazebo::math::Matrix3, 439
 - gazebo::math::Matrix4, 446
 - gazebo::math::Pose, 562
 - gazebo::math::Quaternion, 591
 - gazebo::math::Vector2d, 789
 - gazebo::math::Vector2i, 798
 - gazebo::math::Vector3, 810
 - gazebo::math::Vector4, 820
 - gazebo::physics::Base, 133
- operator[]
 - gazebo::common::Color, 201
 - gazebo::math::Matrix3, 439
 - gazebo::math::Matrix4, 446
 - gazebo::math::Vector2d, 789
 - gazebo::math::Vector2i, 798
 - gazebo::math::Vector3, 810
 - gazebo::math::Vector4, 820
- OrbitViewController
 - gazebo::rendering::OrbitViewController, 519
- OrbitViewController.hh, 973
- PHONG
 - gazebo::common::Material, 431
- PID
 - gazebo::common::PID, 544
- PID.hh, 984
- PLANE_SHAPE
 - gazebo::physics::Base, 128
- POINTS
 - gazebo::common::SubMesh, 709
- PRESS
 - gazebo::common::MouseEvent, 484
- Param
 - sdf::Param, 524
- Param.hh, 973
- Param_V
 - gazebo::common, 80
 - sdf, 99
- ParamPtr
 - sdf, 99
- ParamT
 - sdf::ParamT, 529
- ParamT< T >, 531
- parent
 - gazebo::common::SkeletonNode, 692
 - gazebo::physics::Base, 135
 - gazebo::rendering::Visual, 847
- parentEntity
 - gazebo::physics::Entity, 276
- parentLink
 - gazebo::physics::Joint, 379
- parentName
 - gazebo::sensors::Sensor, 659
- ParseArgs
 - gazebo::Server, 667
- parseFloat
 - Math, 50
- parseGazeboExtension
 - urdf2gazebo::URDF2Gazebo, 771
- parseInt
 - Math, 50
- parseVector3
 - urdf2gazebo::URDF2Gazebo, 772
- parser.hh, 975
- parser_urdf.hh, 976
- pathLength
 - gazebo::physics::Actor, 106
- pause
 - gazebo::event::Events, 293
- pause_incoming
 - Transport, 72
- pause_world
 - Classes for physics and dynamics, 43
- pause_worlds
 - Classes for physics and dynamics, 44
- PauseIncoming
 - gazebo::transport::TopicManager, 758
- Physics.hh, 977
- PhysicsEngine
 - gazebo::physics::PhysicsEngine, 533

PhysicsEngine.hh, 979
 PhysicsEnginePtr
 gazebo::physics, 89
 PhysicsFactory.hh, 980
 PhysicsFactoryFn
 Classes for physics and dynamics, 42
 physicsSub
 gazebo::physics::PhysicsEngine, 541
 PhysicsTypes.hh, 982
 GZ_ALL_COLLIDE, 984
 GZ_FIXED_COLLIDE, 984
 GZ_GHOST_COLLIDE, 984
 GZ_NONE_COLLIDE, 984
 GZ_SENSOR_COLLIDE, 984
 physicsUpdateMutex
 gazebo::physics::PhysicsEngine, 541
 pitchNode
 gazebo::rendering::Camera, 172
 PixelFormat
 gazebo::common::Image, 350
 PlaceOnEntity
 gazebo::physics::Entity, 273
 PlaceOnNearestEntityBelow
 gazebo::physics::Entity, 273
 placeable
 gazebo::physics::Collision, 189
 Plane
 gazebo::math::Plane, 548
 Plane.hh, 985
 PlaneShape
 gazebo::physics::PlaneShape, 551
 PlaneShape.hh, 986
 Play
 gazebo::physics::Actor, 104
 playStartTime
 gazebo::physics::Actor, 106
 Plugin
 sdf::Plugin, 553
 Plugin.hh, 988, 992
 pluginPathsFromEnv
 gazebo::common::SystemPaths, 730
 PluginType
 Common, 32
 plugins
 gazebo::sensors::Sensor, 659
 pointSize
 gazebo::common::Material, 436
 points
 gazebo::math::RotationSpline, 628
 gazebo::math::Spline, 701
 pos
 gazebo::common::MouseEvent, 485
 gazebo::math::Pose, 564
 Pose
 gazebo::math::Pose, 558
 pose
 gazebo::sensors::Sensor, 660
 Pose.hh, 992
 PoseAnimation
 gazebo::common::PoseAnimation, 565
 PoseAnimationPtr
 gazebo::common, 80
 PoseKeyFrame
 gazebo::common::PoseKeyFrame, 568
 poseMsg
 gazebo::physics::Entity, 276
 poseSub
 gazebo::sensors::Sensor, 660
 positions
 gazebo::physics::Contact, 221
 PostRender
 gazebo::rendering::Camera, 165
 gazebo::rendering::DepthCamera, 241
 gazebo::rendering::GpuLaser, 315
 gazebo::rendering::UserCamera, 780
 postRender
 gazebo::event::Events, 293
 PreRender
 gazebo::rendering::Scene, 644
 preRender
 gazebo::event::Events, 293
 precision
 Math, 50
 PrepareHardwareBuffers
 gazebo::rendering::DynamicRenderable, 257
 pressPos
 gazebo::common::MouseEvent, 485
 prevAnimTime
 gazebo::rendering::Camera, 172
 prevAnimationTime
 gazebo::physics::Entity, 276
 prevFrameTime
 gazebo::physics::Actor, 106
 prevPos
 gazebo::common::MouseEvent, 485
 PrimitiveType
 gazebo::common::SubMesh, 709
 Print
 gazebo::common::Exception, 304
 gazebo::physics::Base, 133
 sdf::Plugin, 553
 print_version
 gazebo, 77
 printCollisionGroups
 urdf2gazebo::URDF2Gazebo, 772
 PrintDescription
 sdf::Element, 264
 sdf::SDF, 650

- PrintDoc
 - sdf::SDF, 650
- PrintDocLeftPane
 - sdf::Element, 264
- PrintDocRightPane
 - sdf::Element, 264
- PrintEntityTree
 - gazebo::physics::World, 862
- printMass
 - urdf2gazebo::URDF2Gazebo, 772
- PrintSceneGraph
 - gazebo::rendering::Scene, 644
- PrintSource
 - gazebo::common::NodeTransform, 512
- PrintTransforms
 - gazebo::common::Skeleton, 677
- PrintUsage
 - gazebo::Server, 667
- PrintValues
 - sdf::Element, 264
 - sdf::SDF, 650
- PrintWiki
 - sdf::Element, 264
 - sdf::SDF, 650
- ProcessIncoming
 - gazebo::transport::Node, 503
- ProcessMsg
 - gazebo::physics::BoxShape, 143
 - gazebo::physics::Collision, 188
 - gazebo::physics::CylinderShape, 235
 - gazebo::physics::HeightmapShape, 345
 - gazebo::physics::Inertial, 362
 - gazebo::physics::Link, 411
 - gazebo::physics::Model, 470
 - gazebo::physics::MultiRayShape, 498
 - gazebo::physics::PlaneShape, 552
 - gazebo::physics::RayShape, 607
 - gazebo::physics::Shape, 670
 - gazebo::physics::SphereShape, 697
 - gazebo::physics::SurfaceParams, 723
 - gazebo::physics::TrimeshShape, 764
- ProcessNodes
 - gazebo::transport::TopicManager, 759
- ProcessPublishers
 - gazebo::transport::Node, 503
- ProcessWriteQueue
 - gazebo::transport::Connection, 213
- Projector
 - gazebo::rendering::Projector, 570
- Projector.hh, 993
- provideFeedback
 - urdf2gazebo::GazeboExtension, 311
- Publication
 - gazebo::transport::Publication, 572
- Publication.hh, 994
- PublicationPtr
 - gazebo::transport, 97
- PublicationTransport
 - gazebo::transport::PublicationTransport, 577
- PublicationTransport.hh, 995
- PublicationTransportPtr
 - gazebo::transport, 97
- Publish
 - gazebo::transport::Publication, 575
 - gazebo::transport::Publisher, 580
 - gazebo::transport::TopicManager, 759
- PublishContacts
 - gazebo::physics::ContactManager, 224
- Publisher
 - gazebo::transport::Publisher, 579
- Publisher.hh, 997
- PublisherPtr
 - gazebo::transport, 97
- Purple
 - gazebo::common::Color, 204
- Quaternion
 - gazebo::math::Quaternion, 584, 585
- Quaternion.hh, 999
- r
 - gazebo::common::Color, 204
- R_FLOAT16
 - gazebo::common::Image, 350
- R_FLOAT32
 - gazebo::common::Image, 350
- RAY_SHAPE
 - gazebo::physics::Base, 128
- RELEASE
 - gazebo::common::MouseEvent, 484
- RENDER_PATH_COUNT
 - gazebo::rendering::RenderEngine, 611
- RENDERING_LINE_LIST
 - gazebo::rendering, 93
- RENDERING_LINE_STRIP
 - gazebo::rendering, 93
- RENDERING_MESH_RESOURCE
 - gazebo::rendering, 93
- RENDERING_POINT_LIST
 - gazebo::rendering, 93
- RENDERING_TRIANGLE_FAN
 - gazebo::rendering, 93
- RENDERING_TRIANGLE_LIST
 - gazebo::rendering, 93
- RENDERING_TRIANGLE_STRIP
 - gazebo::rendering, 93
- REPLACE
 - gazebo::common::Material, 430
- RFIDSensor

- gazebo::sensors::RFIDSensor, 615
- RFIDSensor.hh, 1009
- RFIDSensor_V
 - gazebo::sensors, 95
- RFIDSensorPtr
 - gazebo::sensors, 95
- RFIDTag
 - gazebo::sensors::RFIDTag, 617
- RFIDTag.hh, 1009
- RFIDTag_V
 - gazebo::sensors, 95
- RFIDTagPtr
 - gazebo::sensors, 95
- RFIDTagVisual
 - gazebo::rendering::RFIDTagVisual, 619
- RFIDTagVisual.hh, 1010
- RFIDTagVisualPtr
 - gazebo::rendering, 92
- RFIDVisual
 - gazebo::rendering::RFIDVisual, 621
- RFIDVisual.hh, 1011
- RFIDVisualPtr
 - gazebo::rendering, 92
- RGB_FLOAT16
 - gazebo::common::Image, 350
- RGB_FLOAT32
 - gazebo::common::Image, 350
- RGB_INT16
 - gazebo::common::Image, 350
- RGB_INT32
 - gazebo::common::Image, 350
- RGB_INT8
 - gazebo::common::Image, 350
- RGBA
 - gazebo::common::Color, 196
- RGBA_INT8
 - gazebo::common::Image, 350
- RIGHT
 - gazebo::common::MouseEvent, 484
- ROTATE
 - gazebo::common::NodeTransform, 510
- RTShaderSystem.hh, 1014
- Radian
 - gazebo::math::Angle, 113
- Rand.hh, 1000
- rangeCountRatio
 - gazebo::sensors::GpuRaySensor, 329
- rangeElem
 - gazebo::physics::MultiRayShape, 498
 - gazebo::sensors::GpuRaySensor, 330
- rawNW
 - gazebo::common::Skeleton, 678
- RawNodeAnim
 - gazebo::common, 80
- RawNodeWeights
 - gazebo::common, 80
- RawSkeletonAnim
 - gazebo::common, 80
- rawTransforms
 - gazebo::common::SkeletonNode, 692
- rayCountRatio
 - gazebo::sensors::GpuRaySensor, 330
- rayElem
 - gazebo::physics::MultiRayShape, 499
- RaySensor
 - gazebo::sensors::RaySensor, 598
- RaySensor.hh, 1001
- RaySensor_V
 - gazebo::sensors, 95
- RaySensorPtr
 - gazebo::sensors, 95
- RayShape
 - gazebo::physics::RayShape, 605, 606
- RayShape.hh, 1002
- RayShapePtr
 - gazebo::physics, 89
- rays
 - gazebo::physics::MultiRayShape, 499
- Read
 - gazebo::transport::Connection, 213
- ReadCallback
 - gazebo::transport::Connection, 209
- readDoc
 - sdf, 100
- readFile
 - sdf, 100
- readString
 - sdf, 100
- readXml
 - sdf, 100
- realTime
 - gazebo::physics::State, 705
- RecalcTangents
 - gazebo::math::RotationSpline, 627
 - gazebo::math::Spline, 700
- RecalculateMatrix
 - gazebo::common::NodeTransform, 512
- RecalculateNormals
 - gazebo::common::Mesh, 453
 - gazebo::common::SubMesh, 714
- Red
 - gazebo::common::Color, 204
- reduceCollisionToParent
 - urdf2gazebo::URDF2Gazebo, 772
- reduceCollisionsToParent
 - urdf2gazebo::URDF2Gazebo, 772
- reduceFixedJoints
 - urdf2gazebo::URDF2Gazebo, 772

- reduceGazeboExtensionContactSensorFrameReplace
 - urdf2gazebo::URDF2Gazebo, 772
- reduceGazeboExtensionFrameReplace
 - urdf2gazebo::URDF2Gazebo, 772
- reduceGazeboExtensionGripperFrameReplace
 - urdf2gazebo::URDF2Gazebo, 772
- reduceGazeboExtensionJointFrameReplace
 - urdf2gazebo::URDF2Gazebo, 773
- reduceGazeboExtensionPluginFrameReplace
 - urdf2gazebo::URDF2Gazebo, 773
- reduceGazeboExtensionProjectorFrameReplace
 - urdf2gazebo::URDF2Gazebo, 773
- reduceGazeboExtensionProjectorTransformReduction
 - urdf2gazebo::URDF2Gazebo, 773
- reduceGazeboExtensionSensorTransformReduction
 - urdf2gazebo::URDF2Gazebo, 773
- reduceGazeboExtensionToParent
 - urdf2gazebo::URDF2Gazebo, 773
- reduceGazeboExtensionsTransformReduction
 - urdf2gazebo::URDF2Gazebo, 773
- reduceInertialToParent
 - urdf2gazebo::URDF2Gazebo, 773
- reduceJointsToParent
 - urdf2gazebo::URDF2Gazebo, 773
- reduceVisualToParent
 - urdf2gazebo::URDF2Gazebo, 774
- reduceVisualsToParent
 - urdf2gazebo::URDF2Gazebo, 773
- reduction_transform
 - urdf2gazebo::GazeboExtension, 311
- RegisterAll
 - gazebo::physics::PhysicsFactory, 542
 - gazebo::sensors::SensorFactory, 661
- RegisterPhysicsEngine
 - gazebo::physics::PhysicsFactory, 542
- RegisterSensor
 - gazebo::sensors::SensorFactory, 661
- RegisterTopicNamespace
 - gazebo::transport::ConnectionManager, 216
 - gazebo::transport::TopicManager, 759
- relativeEndPos
 - gazebo::physics::RayShape, 609
- relativeStartPos
 - gazebo::physics::RayShape, 609
- Remove
 - gazebo::common::LogRecord, 425
- remove_scene
 - Rendering, 63
- remove_sensor
 - Sensors, 69
- remove_sensors
 - Sensors, 69
- remove_worlds
 - Classes for physics and dynamics, 44
- RemoveChild
 - gazebo::physics::Base, 133
 - gazebo::physics::Model, 470
- RemoveChildJoint
 - gazebo::physics::Link, 411
- RemoveChildren
 - gazebo::physics::Base, 133
- RemoveConnection
 - gazebo::transport::ConnectionManager, 216
- RemoveNode
 - gazebo::transport::TopicManager, 759
- RemoveParentJoint
 - gazebo::physics::Link, 411
- RemovePlugin
 - gazebo::physics::World, 862
 - gazebo::rendering::Visual, 842
- RemoveScene
 - gazebo::rendering::RenderEngine, 613
 - gazebo::rendering::RTShaderSystem, 631
- removeScene
 - gazebo::rendering::Events, 282
- RemoveSensor
 - gazebo::sensors::SensorManager, 664
- RemoveSensors
 - gazebo::sensors::SensorManager, 664
- RemoveShadows
 - gazebo::rendering::RTShaderSystem, 631
- RemoveSubscription
 - gazebo::transport::Publication, 575
- RemoveTransport
 - gazebo::transport::Publication, 575
- RemoveVisual
 - gazebo::rendering::Scene, 644
- Render
 - gazebo::rendering::Camera, 166
- render
 - gazebo::event::Events, 293
- RenderEngine.hh, 1003
- RenderEvents.hh, 1004
- RenderImpl
 - gazebo::rendering::Camera, 166
- RenderOpType
 - gazebo::rendering, 93
- RenderPathType
 - gazebo::rendering::RenderEngine, 611
- renderTarget
 - gazebo::rendering::Camera, 172
- renderTexture
 - gazebo::rendering::Camera, 172
- RenderTypes.hh, 1007
 - GZ_VISIBILITY_ALL, 1008
 - GZ_VISIBILITY_GUI, 1008
 - GZ_VISIBILITY_NOT_SELECTABLE, 1008
 - GZ_VISIBILITY_SELECTION, 1008

- Rendering, 61
 - create_scene, 63
 - fini, 63
 - get_scene, 63
 - init, 63
 - load, 63
 - remove_scene, 63
- rendering.h, 1005
- Rendering.hh, 1005
- request
 - Transport, 73
- requestPub
 - gazebo::physics::Entity, 277
- requestSub
 - gazebo::physics::PhysicsEngine, 541
- requests
 - gazebo::rendering::Camera, 172
- required
 - sdf::Param, 527
- Rescale
 - gazebo::common::Image, 353
- Reset
 - gazebo::common::Color, 201
 - gazebo::common::PID, 545
 - gazebo::common::SkeletonNode, 689
 - gazebo::math::Pose, 562
 - gazebo::ModelPlugin, 477
 - gazebo::physics::Base, 133, 134
 - gazebo::physics::Contact, 221
 - gazebo::physics::Entity, 273
 - gazebo::physics::Inertial, 362
 - gazebo::physics::Joint, 376
 - gazebo::physics::JointController, 380
 - gazebo::physics::Link, 411
 - gazebo::physics::Model, 470
 - gazebo::physics::PhysicsEngine, 538
 - gazebo::physics::World, 862
 - gazebo::SensorPlugin, 666
 - gazebo::SystemPlugin, 732
 - gazebo::VisualPlugin, 849
 - gazebo::WorldPlugin, 865
 - sdf::Element, 264
 - sdf::Param, 526
 - sdf::ParamT, 530
- ResetCount
 - gazebo::physics::ContactManager, 224
- ResetEntities
 - gazebo::physics::World, 862
- ResetTime
 - gazebo::physics::World, 862
- Resize
 - gazebo::rendering::GUIOverlay, 339
 - gazebo::rendering::UserCamera, 780
 - gazebo::rendering::WindowManager, 851
- responsePub
 - gazebo::physics::PhysicsEngine, 541
- Road, 623
 - gazebo::physics::Road, 622
- Road.hh, 1011
- Road2d
 - gazebo::rendering::Road2d, 624
- Road2d.hh, 1013
- RoadPtr
 - gazebo::physics, 89
- root
 - gazebo::common::Skeleton, 678
 - gazebo::rendering::RenderEngine, 613
 - sdf::SDF, 650
- rot
 - gazebo::math::Pose, 564
- Rotate
 - gazebo::physics::Inertial, 362
- rotate
 - gazebo::common::PoseKeyFrame, 569
- RotatePitch
 - gazebo::rendering::Camera, 166
- RotatePositionAboutOrigin
 - gazebo::math::Pose, 562
- RotateVector
 - gazebo::math::Quaternion, 591
- RotateVectorReverse
 - gazebo::math::Quaternion, 591
- RotateYaw
 - gazebo::rendering::Camera, 166
- RotationSpline
 - gazebo::math::RotationSpline, 625
- RotationSpline.hh, 1013
- Round
 - gazebo::math::Pose, 563
 - gazebo::math::Quaternion, 592
 - gazebo::math::Vector3, 810
- Run
 - gazebo::Master, 427
 - gazebo::physics::World, 862
 - gazebo::sensors::SensorManager, 664
 - gazebo::Server, 667
 - gazebo::transport::ConnectionManager, 216
- run
 - gazebo, 77
 - Sensors, 69
 - Transport, 73
- run_once
 - Sensors, 69
- run_world
 - Classes for physics and dynamics, 44
- run_worlds
 - Classes for physics and dynamics, 44
- RunOnce

- gazebo::Master, 427
- RunThread
 - gazebo::Master, 427
- RunUpdate
 - gazebo::transport::ConnectionManager, 217
- SCALE
 - gazebo::common::NodeTransform, 510
- SCREW_JOINT
 - gazebo::physics::Base, 128
- SCROLL
 - gazebo::common::MouseEvent, 484
- SDF
 - sdf::SDF, 650
- SDF.hh, 1018
 - SDF_VERSION, 1019
- SDF_VERSION
 - SDF.hh, 1019
- SDFPtr
 - sdf, 99
- SENSOR_PLUGIN
 - Common, 32
- SHADE_COUNT
 - gazebo::common::Material, 431
- SHAPE
 - gazebo::physics::Base, 128
- SLIDER_JOINT
 - gazebo::physics::Base, 128
- SPHERE_SHAPE
 - gazebo::physics::Base, 128
- SSLM_NormalMapLightingObjectSpace
 - gazebo::rendering::RTShaderSystem, 630
- SSLM_NormalMapLightingTangentSpace
 - gazebo::rendering::RTShaderSystem, 630
- SSLM_PerPixelLighting
 - gazebo::rendering::RTShaderSystem, 630
- SSLM_PerVertexLighting
 - gazebo::rendering::RTShaderSystem, 630
- STLLoader
 - gazebo::common::STLLoader, 706
- STLLoader.hh, 1037
 - COR3_MAX, 1039
 - FACE_MAX, 1039
 - LINE_MAX_LEN, 1039
 - ORDER_MAX, 1039
- STOP_CFM
 - gazebo::physics::Joint, 369
- STOP_ERP
 - gazebo::physics::Joint, 369
- SUSPENSION_CFM
 - gazebo::physics::Joint, 369
- SUSPENSION_ERP
 - gazebo::physics::Joint, 369
- SYSTEM_PLUGIN
 - Common, 32
- Save
 - gazebo::physics::World, 862
- saveCount
 - gazebo::rendering::Camera, 173
- SaveFrame
 - gazebo::rendering::Camera, 166
 - gazebo::sensors::CameraSensor, 177
 - gazebo::sensors::DepthCameraSensor, 245
- saveFrameBuffer
 - gazebo::rendering::Camera, 173
- SavePNG
 - gazebo::common::Image, 353
- Scale
 - gazebo::common::Mesh, 453
 - gazebo::common::NodeAnimation, 507
 - gazebo::common::Skeleton, 677
 - gazebo::common::SkeletonAnimation, 682
 - gazebo::common::SubMesh, 714
 - gazebo::math::Quaternion, 592
- scale
 - gazebo::physics::HeightmapShape, 345
- ScaleXAxis
 - gazebo::rendering::AxisVisual, 122
- ScaleYAxis
 - gazebo::rendering::AxisVisual, 122
- ScaleZAxis
 - gazebo::rendering::AxisVisual, 122
- scanElem
 - gazebo::physics::MultiRayShape, 499
 - gazebo::sensors::GpuRaySensor, 330
- Scene
 - gazebo::rendering::Scene, 635
- scene
 - gazebo::rendering::Camera, 173
 - gazebo::rendering::Visual, 847
- Scene.hh, 1015
- SceneFromSDF
 - Messages, 58
- sceneNode
 - gazebo::rendering::Camera, 173
 - gazebo::rendering::Visual, 847
- ScenePtr
 - gazebo::rendering, 92
- ScrewJoint
 - gazebo::physics::ScrewJoint, 647
- ScrewJoint.hh, 1016
- scriptLength
 - gazebo::physics::Actor, 106
- scroll
 - gazebo::common::MouseEvent, 485
- sdf, 98
 - addNestedModel, 99
 - copyChildren, 99

- ElementPtr, 99
- ElementPtr_V, 99
- gazebo::physics::Base, 135
- gazebo::physics::PhysicsEngine, 541
- gazebo::rendering::Camera, 173
- gazebo::sensors::Sensor, 660
- init, 99
- initDoc, 99
- initFile, 99
- initString, 99
- initXml, 99
- Param_V, 99
- ParamPtr, 99
- readDoc, 100
- readFile, 100
- readString, 100
- readXml, 100
- SDFPtr, 99
- sdf.hh, 1017
- sdf::Converter, 232
 - Convert, 232
- sdf::Element, 258
 - ~Element, 261
 - AddAttribute, 261
 - AddElement, 261
 - AddElementDescription, 261
 - AddValue, 261
 - ClearElements, 261
 - Clone, 261
 - Copy, 261
 - Element, 261
 - GetAttribute, 261, 262
 - GetAttributeCount, 262
 - GetAttributeSet, 262
 - GetCopyChildren, 262
 - GetDescription, 262
 - GetElement, 262
 - GetElementDescription, 262
 - GetElementDescriptionCount, 262
 - GetElementImpl, 262
 - GetFirstElement, 262
 - GetInclude, 262
 - GetName, 262
 - GetNextElement, 263
 - GetParent, 263
 - GetRequired, 263
 - GetValue, 263
 - GetValueBool, 263
 - GetValueChar, 263
 - GetValueColor, 263
 - GetValueDouble, 263
 - GetValueFloat, 263
 - GetValueInt, 263
 - GetValuePose, 263
 - GetValueQuaternion, 263
 - GetValueString, 263
 - GetValueTime, 263
 - GetValueUInt, 263
 - GetValueVector2d, 263
 - GetValueVector3, 263
 - HasAttribute, 263
 - HasElement, 263
 - HasElementDescription, 263
 - InsertElement, 264
 - PrintDescription, 264
 - PrintDocLeftPane, 264
 - PrintDocRightPane, 264
 - PrintValues, 264
 - PrintWiki, 264
 - Reset, 264
 - Set, 264, 265
 - SetCopyChildren, 265
 - SetDescription, 265
 - SetInclude, 265
 - SetName, 265
 - SetParent, 265
 - SetRequired, 265
 - ToString, 265
 - Update, 265
- sdf::Param, 521
 - ~Param, 524
 - Clone, 524
 - description, 527
 - Get, 524
 - GetAsString, 524
 - GetDefaultAsString, 524
 - GetDescription, 525
 - GetKey, 525
 - GetRequired, 525
 - GetSet, 525
 - GetTypeName, 525
 - IsBool, 525
 - IsChar, 525
 - IsColor, 525
 - IsDouble, 525
 - IsFloat, 525
 - IsInt, 525
 - IsPose, 525
 - IsQuaternion, 525
 - IsStr, 525
 - IsTime, 525
 - IsUInt, 525
 - IsVector2d, 525
 - IsVector2i, 525
 - IsVector3, 525
 - key, 527
 - Param, 524
 - required, 527

- Reset, 526
- Set, 526
- set, 527
- setDescription, 526
- setFromString, 526
- setUpdateFunc, 526
- typeName, 527
- Update, 527
- updateFunc, 527
- sdf::ParamT
 - ~ParamT, 529
 - Clone, 529
 - defaultValue, 530
 - getAsString, 529
 - getDefaultAsString, 529
 - defaultValue, 529
 - getValue, 529
 - operator<<, 530
 - operator*, 529
 - ParamT, 529
 - Reset, 530
 - Set, 530
 - setFromString, 530
 - setValue, 530
 - Update, 530
 - value, 530
- sdf::ParamT< T >, 527
- sdf::Plugin, 553
 - Clear, 553
 - data, 554
 - filename, 554
 - name, 554
 - Plugin, 553
 - Print, 553
- sdf::SDF, 649
 - PrintDescription, 650
 - PrintDoc, 650
 - PrintValues, 650
 - PrintWiki, 650
 - root, 650
 - SDF, 650
 - setFromString, 650
 - toString, 650
 - version, 650
 - Write, 650
- sec
 - gazebo::common::Time, 752
- SecToNano
 - gazebo::common::Time, 751
- SelectVisual
 - gazebo::rendering::Scene, 644
- SelectionObj
 - gazebo::rendering::SelectionObj, 651
- SelectionObj.hh, 1019
- self_collide
 - urdf2gazebo::GazeboExtension, 311
- SendMessage
 - gazebo::transport::Publisher, 581
- Sensor
 - gazebo::sensors::Sensor, 655
- Sensor.hh, 1020
- Sensor_V
 - gazebo::sensors, 95
- SensorFactor, 660
- SensorFactory.hh, 1021
- SensorFactoryFn
 - gazebo::sensors, 95
- SensorManager.hh, 1022
- SensorPlugin
 - gazebo::SensorPlugin, 666
- SensorPluginPtr
 - gazebo, 77
- SensorPtr
 - gazebo::sensors, 95
- SensorTypes.hh, 1024
- Sensors, 66
 - create_sensor, 68
 - fini, 68
 - GZ_REGISTER_STATIC_SENSOR, 67
 - get_sensor, 68
 - init, 68
 - load, 69
 - remove_sensor, 69
 - remove_sensors, 69
 - run, 69
 - run_once, 69
 - stop, 69
- Sensors.hh, 1023
- SensorsInitialized
 - gazebo::sensors::SensorManager, 664
- Server
 - gazebo::Server, 667
- Server.hh, 1026
- Set
 - gazebo::common::Color, 201
 - gazebo::common::NodeTransform, 512
 - gazebo::common::Time, 751
 - gazebo::math::Matrix4, 446
 - gazebo::math::Plane, 549
 - gazebo::math::Pose, 563
 - gazebo::math::Quaternion, 592
 - gazebo::math::Vector2d, 789
 - gazebo::math::Vector2i, 798
 - gazebo::math::Vector3, 810
 - gazebo::math::Vector4, 820
 - Messages, 58–60
 - sdf::Element, 264, 265
 - sdf::Param, 526

- sdf::ParamT, 530
- set
 - sdf::Param, 527
- SetActive
 - gazebo::rendering::SelectionObj, 652
 - gazebo::sensors::DepthCameraSensor, 245
 - gazebo::sensors::Sensor, 658
- SetAltitude
 - gazebo::physics::PlaneShape, 552
- SetAmbient
 - gazebo::common::Material, 433
 - gazebo::rendering::Visual, 842
- SetAmbientColor
 - gazebo::rendering::Scene, 644
- SetAnchor
 - gazebo::physics::Joint, 376
 - gazebo::physics::ScrewJoint, 648
 - gazebo::physics::SliderJoint, 694
- SetAngle
 - gazebo::physics::Joint, 376
- SetAngleMax
 - gazebo::sensors::GpuRaySensor, 327
- SetAngleMin
 - gazebo::sensors::GpuRaySensor, 328
- SetAngularAccel
 - gazebo::physics::Link, 411
 - gazebo::physics::Model, 470
- SetAngularDamping
 - gazebo::physics::Link, 411
- SetAngularVel
 - gazebo::physics::Link, 412
 - gazebo::physics::Model, 470
- SetAnimation
 - gazebo::physics::Entity, 274
- SetAspectRatio
 - gazebo::rendering::Camera, 167
- SetAttenuation
 - gazebo::rendering::Light, 395
- SetAttribute
 - gazebo::physics::Joint, 376, 377
- SetAutoCalculate
 - gazebo::math::RotationSpline, 627
 - gazebo::math::Spline, 700
- SetAutoDisable
 - gazebo::physics::Link, 412
 - gazebo::physics::Model, 471
- SetAutoDisableFlag
 - gazebo::physics::PhysicsEngine, 538
- SetAxis
 - gazebo::physics::BallJoint, 124
 - gazebo::physics::Joint, 377
- SetAxisMaterial
 - gazebo::rendering::AxisVisual, 122
- SetBackgroundColor
 - gazebo::rendering::Scene, 644
- SetBaseline
 - gazebo::rendering::MovableText, 490
- SetBindShapeTransform
 - gazebo::common::Skeleton, 677
- SetBlendFactors
 - gazebo::common::Material, 434
- SetBlendMode
 - gazebo::common::Material, 434
- SetCamera
 - gazebo::rendering::WindowManager, 851
- SetCanonicalLink
 - gazebo::physics::Entity, 274
- SetCaptureData
 - gazebo::rendering::Camera, 167
- SetCastShadows
 - gazebo::rendering::Light, 395
 - gazebo::rendering::Visual, 842
- SetCategoryBits
 - gazebo::physics::Collision, 188
- SetCellCount
 - gazebo::rendering::Grid, 333
- SetCellLength
 - gazebo::rendering::Grid, 333
- SetCharHeight
 - gazebo::rendering::MovableText, 490
- SetClipDist
 - gazebo::rendering::Camera, 167
- SetCmd
 - gazebo::common::PID, 545
- SetCmdMax
 - gazebo::common::PID, 545
- SetCmdMin
 - gazebo::common::PID, 545
- SetCoG
 - gazebo::physics::Inertial, 362
- SetCol
 - gazebo::math::Matrix3, 440
- SetCollideBits
 - gazebo::physics::Collision, 188
- SetCollideMode
 - gazebo::physics::Link, 412
 - gazebo::physics::Model, 471
- SetCollision
 - gazebo::physics::Collision, 188
- SetColor
 - gazebo::rendering::Grid, 333
 - gazebo::rendering::MovableText, 490
- SetComponent
 - gazebo::common::NodeTransform, 512
- SetContactMaxCorrectingVel
 - gazebo::physics::PhysicsEngine, 538
- SetContactSurfaceLayer
 - gazebo::physics::PhysicsEngine, 539

- SetContactsEnabled
 - gazebo::physics::Collision, 188
- SetCopyChildren
 - sdf::Element, 265
- SetDGain
 - gazebo::common::PID, 546
- SetDamping
 - gazebo::physics::Joint, 377
- SetDepthTarget
 - gazebo::rendering::DepthCamera, 242
- SetDepthWrite
 - gazebo::common::Material, 434
- SetDescription
 - sdf::Element, 265
 - sdf::Param, 526
- SetDiffuse
 - gazebo::common::Material, 434
 - gazebo::rendering::Visual, 842
- SetDiffuseColor
 - gazebo::rendering::Light, 395
- SetDirection
 - gazebo::rendering::Light, 396
- SetDistance
 - gazebo::rendering::OrbitViewController, 520
- SetDistanceRange
 - gazebo::rendering::OrbitViewController, 520
- SetEmissive
 - gazebo::common::Material, 434
 - gazebo::rendering::LaserVisual, 391
 - gazebo::rendering::Visual, 842
- SetEnabled
 - gazebo::common::DiagnosticManager, 248
 - gazebo::physics::Link, 412
 - gazebo::physics::Model, 471
 - gazebo::rendering::ContactVisual, 230
 - gazebo::rendering::Projector, 571
 - gazebo::rendering::ViewController, 827
- SetFiducial
 - gazebo::physics::RayShape, 607
- SetFilename
 - gazebo::physics::TrimeshShape, 764
- SetFocalPoint
 - gazebo::rendering::OrbitViewController, 521
 - gazebo::rendering::UserCamera, 780
- SetFog
 - gazebo::rendering::Scene, 644
- SetFontName
 - gazebo::rendering::MovableText, 490
- SetForce
 - gazebo::physics::Joint, 377
 - gazebo::physics::Link, 412
- SetFromABGR
 - gazebo::common::Color, 202
- SetFromARGB
 - gazebo::common::Color, 202
- SetFromAxes
 - gazebo::math::Matrix3, 440
- SetFromAxis
 - gazebo::math::Matrix3, 440
 - gazebo::math::Quaternion, 592
- SetFromBGRA
 - gazebo::common::Color, 202
- SetFromData
 - gazebo::common::Image, 353
- SetFromDegree
 - gazebo::math::Angle, 114
- SetFromEuler
 - gazebo::math::Quaternion, 593
- SetFromHSV
 - gazebo::common::Color, 202
- SetFromRGBA
 - gazebo::common::Color, 202
- SetFromRadian
 - gazebo::math::Angle, 114
- SetFromString
 - sdf::Param, 526
 - sdf::ParamT, 530
 - sdf::SDF, 650
- SetFromYUV
 - gazebo::common::Color, 203
- SetGravity
 - gazebo::physics::PhysicsEngine, 539
- SetGravityMode
 - gazebo::physics::Link, 413
 - gazebo::physics::Model, 471
- SetGrid
 - gazebo::rendering::Scene, 645
- SetHFOV
 - gazebo::rendering::Camera, 167
- SetHandle
 - gazebo::common::SkeletonNode, 689
- SetHeight
 - gazebo::rendering::Grid, 333
- SetHighStop
 - gazebo::physics::BallJoint, 125
 - gazebo::physics::Joint, 377
- SetHighlight
 - gazebo::rendering::SelectionObj, 652
- SetHighlighted
 - gazebo::rendering::Visual, 843
- SetIGain
 - gazebo::common::PID, 546
- SetIMax
 - gazebo::common::PID, 546
- SetIMin
 - gazebo::common::PID, 546
- SetIXX
 - gazebo::physics::Inertial, 363

- SetIXY
 - gazebo::physics::Inertial, 363
- SetIXZ
 - gazebo::physics::Inertial, 363
- SetIYY
 - gazebo::physics::Inertial, 363
- SetIYZ
 - gazebo::physics::Inertial, 363
- SetIZZ
 - gazebo::physics::Inertial, 364
- SetId
 - gazebo::common::SkeletonNode, 689
- SetImageHeight
 - gazebo::rendering::Camera, 167
- SetImageSize
 - gazebo::rendering::Camera, 168
- SetImageWidth
 - gazebo::rendering::Camera, 168
- SetInclude
 - sdf::Element, 265
- SetIndexCount
 - gazebo::common::SubMesh, 714
- SetInertiaMatrix
 - gazebo::physics::Inertial, 363
- SetInertial
 - gazebo::physics::Link, 413
- SetInitialRelativePose
 - gazebo::physics::Entity, 274
- SetInitialTransform
 - gazebo::common::SkeletonNode, 690
- SetInverseBindTransform
 - gazebo::common::SkeletonNode, 690
- SetJointAnimation
 - gazebo::physics::Model, 471
- SetJointPosition
 - gazebo::physics::JointController, 381
 - gazebo::physics::Model, 472
- SetJointPositions
 - gazebo::physics::JointController, 381
 - gazebo::physics::Model, 472
- SetKinematic
 - gazebo::physics::Link, 413
- SetLaserRetro
 - gazebo::physics::Collision, 188
 - gazebo::physics::Link, 413
 - gazebo::physics::Model, 472
- SetLength
 - gazebo::common::Animation, 118
 - gazebo::physics::CylinderShape, 235
 - gazebo::physics::RayShape, 608
- SetLightType
 - gazebo::rendering::Light, 396
- SetLighting
 - gazebo::common::Material, 435
- SetLineWidth
 - gazebo::rendering::Grid, 334
- SetLinearAccel
 - gazebo::physics::Link, 413
 - gazebo::physics::Model, 472
- SetLinearDamping
 - gazebo::physics::Link, 413
- SetLinearVel
 - gazebo::physics::Link, 414
 - gazebo::physics::Model, 472
- SetLinkWorldPose
 - gazebo::physics::Model, 473
- SetLocallyAdvertised
 - gazebo::transport::Publication, 575
- SetLowStop
 - gazebo::physics::BallJoint, 125
 - gazebo::physics::Joint, 378
- SetMass
 - gazebo::physics::Inertial, 364
- SetMaterial
 - gazebo::rendering::Visual, 843
- SetMaterialIndex
 - gazebo::common::SubMesh, 714
- SetMaxContacts
 - gazebo::physics::PhysicsEngine, 539
- SetMaxForce
 - gazebo::physics::Joint, 378
- SetModel
 - gazebo::physics::Joint, 378
- SetModelTransform
 - gazebo::common::SkeletonNode, 690
- SetName
 - gazebo::common::Mesh, 453
 - gazebo::common::NodeAnimation, 507
 - gazebo::common::SkeletonAnimation, 682
 - gazebo::common::SkeletonNode, 690
 - gazebo::physics::Base, 134
 - gazebo::physics::Entity, 274
 - gazebo::physics::State, 705
 - gazebo::rendering::Camera, 168
 - gazebo::rendering::Light, 396
 - gazebo::rendering::Visual, 843
 - sdf::Element, 265
- SetNormal
 - gazebo::common::SubMesh, 715
 - gazebo::physics::PlaneShape, 552
- SetNormalCount
 - gazebo::common::SubMesh, 715
- SetNormalMap
 - gazebo::rendering::Visual, 843
- SetNumVertAttached
 - gazebo::common::Skeleton, 678
- SetOperationType
 - gazebo::rendering::DynamicRenderable, 258

- SetPGain
 - gazebo::common::PID, 546
- SetParams
 - gazebo::Server, 667
- SetParent
 - gazebo::common::SkeletonNode, 690
 - gazebo::physics::Base, 134
 - gazebo::sensors::CameraSensor, 177
 - gazebo::sensors::DepthCameraSensor, 245
 - gazebo::sensors::Sensor, 658
 - sdf::Element, 265
- SetParentSensor
 - gazebo::rendering::GpuLaser, 316
- SetPath
 - gazebo::common::Mesh, 454
- SetPaused
 - gazebo::physics::World, 863
- SetPerPixelLighting
 - gazebo::rendering::RTShaderSystem, 632
- SetPitch
 - gazebo::rendering::OrbitViewController, 521
- SetPoint
 - gazebo::rendering::DynamicLines, 254
- SetPointSize
 - gazebo::common::Material, 435
- SetPoints
 - gazebo::physics::RayShape, 608
- SetPose
 - gazebo::rendering::Visual, 843
- SetPosition
 - gazebo::rendering::Light, 396
 - gazebo::rendering::Visual, 843
- SetPrimitiveType
 - gazebo::common::SubMesh, 715
- SetPublication
 - gazebo::transport::Publisher, 581
- SetQuiet
 - Common, 35
- SetRadius
 - gazebo::physics::CylinderShape, 235
 - gazebo::physics::SphereShape, 697
- SetRange
 - gazebo::rendering::Light, 396
- SetRangeCount
 - gazebo::rendering::GpuLaser, 316
- SetRelativePose
 - gazebo::physics::Entity, 274
- SetRenderRate
 - gazebo::rendering::Camera, 168
- SetRenderTarget
 - gazebo::rendering::Camera, 168
 - gazebo::rendering::UserCamera, 781
- SetRequired
 - sdf::Element, 265
- SetRetro
 - gazebo::physics::RayShape, 608
- SetRibbonTrail
 - gazebo::rendering::Visual, 844
- SetRootNode
 - gazebo::common::Skeleton, 678
- SetRotation
 - gazebo::common::PoseKeyFrame, 568
 - gazebo::rendering::Visual, 844
- SetSID
 - gazebo::common::NodeTransform, 512
- SetSORPGSIters
 - gazebo::physics::PhysicsEngine, 539
- SetSORPGSPreconIters
 - gazebo::physics::PhysicsEngine, 539
- SetSORPGSW
 - gazebo::physics::PhysicsEngine, 540
- SetSaveFramePathname
 - gazebo::rendering::Camera, 169
- SetSaveable
 - gazebo::physics::Base, 134
- SetScale
 - gazebo::math::Matrix4, 447
 - gazebo::physics::TrimeshShape, 764
 - gazebo::rendering::Visual, 844
- SetScene
 - gazebo::rendering::Camera, 169
 - gazebo::rendering::Visual, 844
- SetSceneNode
 - gazebo::rendering::Camera, 169
- SetSeed
 - gazebo::math::Rand, 596
- SetSelected
 - gazebo::physics::Base, 134
 - gazebo::physics::Link, 414
 - gazebo::rendering::Light, 396
- setSelectedEntity
 - gazebo::event::Events, 293
- SetSelfCollide
 - gazebo::physics::Link, 414
- SetShadeMode
 - gazebo::common::Material, 435
- SetShaderType
 - gazebo::rendering::Visual, 844
- SetShadowsEnabled
 - gazebo::rendering::Scene, 645
- SetShape
 - gazebo::physics::Collision, 189
- SetShininess
 - gazebo::common::Material, 435
- SetShowOnTop
 - gazebo::rendering::MovableText, 491
- SetSimTime
 - gazebo::physics::World, 863

- SetSize
 - gazebo::physics::BoxShape, 143
 - gazebo::physics::CylinderShape, 236
 - gazebo::physics::PlaneShape, 552
- SetSkeleton
 - gazebo::common::Mesh, 454
- SetSkeletonPose
 - gazebo::rendering::Visual, 845
- SetSourceValues
 - gazebo::common::NodeTransform, 512, 513
- SetSpaceWidth
 - gazebo::rendering::MovableText, 491
- SetSpecular
 - gazebo::common::Material, 435
 - gazebo::rendering::Visual, 845
- SetSpecularColor
 - gazebo::rendering::Light, 397
- SetSpotFalloff
 - gazebo::rendering::Light, 397
- SetSpotInnerAngle
 - gazebo::rendering::Light, 397
- SetSpotOuterAngle
 - gazebo::rendering::Light, 397
- SetState
 - gazebo::physics::Collision, 189
 - gazebo::physics::Joint, 378
 - gazebo::physics::Link, 414
 - gazebo::physics::Model, 473
 - gazebo::physics::World, 863
- SetStatic
 - gazebo::physics::Entity, 275
- setStaticFlag
 - urdf2gazebo::GazeboExtension, 311
- SetStepTime
 - gazebo::physics::PhysicsEngine, 540
- SetSubMeshCenter
 - gazebo::common::SubMesh, 715
- SetTension
 - gazebo::math::Spline, 700
- SetTexCoord
 - gazebo::common::SubMesh, 715
- SetTexCoordCount
 - gazebo::common::SubMesh, 715
- SetText
 - gazebo::rendering::MovableText, 491
- SetTextAlignment
 - gazebo::rendering::MovableText, 491
- SetTexture
 - gazebo::rendering::Projector, 571
- SetTextureImage
 - gazebo::common::Material, 435
- SetThreadPitch
 - gazebo::physics::ScrewJoint, 648
- SetTime
 - gazebo::common::Animation, 118
- SetToldentity
 - gazebo::math::Quaternion, 593
- SetToMax
 - gazebo::math::Vector3, 810
- SetToMin
 - gazebo::math::Vector3, 811
- SetToWallTime
 - gazebo::common::Time, 751
- SetTorque
 - gazebo::physics::Link, 414
- SetTransform
 - gazebo::common::SkeletonNode, 690
- SetTranslate
 - gazebo::math::Matrix4, 447
- SetTranslation
 - gazebo::common::PoseKeyFrame, 568
- SetTransparency
 - gazebo::common::Material, 436
 - gazebo::rendering::Visual, 845
- SetType
 - gazebo::common::NodeTransform, 513
 - gazebo::common::SkeletonNode, 691
- SetUpdateFunc
 - sdf::Param, 526
- SetUpdateRate
 - gazebo::physics::PhysicsEngine, 540
 - gazebo::sensors::Sensor, 658
- SetUserData
 - gazebo::rendering::Grid, 334
- SetValue
 - gazebo::common::NumericKeyFrame, 517
 - sdf::ParamT, 530
- SetVelocity
 - gazebo::physics::Joint, 378
- SetVertex
 - gazebo::common::SubMesh, 716
- SetVertexCount
 - gazebo::common::SubMesh, 716
- SetVerticalAngleMax
 - gazebo::sensors::GpuRaySensor, 328
- SetVerticalAngleMin
 - gazebo::sensors::GpuRaySensor, 328
- SetViewController
 - gazebo::rendering::UserCamera, 781
- SetViewportDimensions
 - gazebo::rendering::UserCamera, 781
- SetVisibilityFlags
 - gazebo::rendering::Visual, 845
- SetVisible
 - gazebo::rendering::Scene, 645
 - gazebo::rendering::Visual, 845
 - gazebo::rendering::WireBox, 853
- SetWindowId

- gazebo::rendering::Camera, 169
- SetWorld
 - gazebo::physics::Base, 134
- SetWorldCFM
 - gazebo::physics::PhysicsEngine, 540
- SetWorldERP
 - gazebo::physics::PhysicsEngine, 540
- SetWorldPose
 - gazebo::physics::Entity, 275
 - gazebo::rendering::Camera, 169
 - gazebo::rendering::UserCamera, 781
 - gazebo::rendering::Visual, 846
- SetWorldPosition
 - gazebo::rendering::Camera, 169
 - gazebo::rendering::Visual, 846
- SetWorldRotation
 - gazebo::rendering::Camera, 169
 - gazebo::rendering::Visual, 846
- SetWorldTwist
 - gazebo::physics::Entity, 275
- SetYaw
 - gazebo::rendering::OrbitViewController, 521
- ShadeMode
 - gazebo::common::Material, 430
- shadeMode
 - gazebo::common::Material, 437
- ShadeModeStr
 - gazebo::common::Material, 437
- Shape
 - gazebo::physics::Shape, 669
- shape
 - gazebo::physics::Collision, 189
- Shape.hh, 1027
- ShapePtr
 - gazebo::physics, 89
- shift
 - gazebo::common::MouseEvent, 485
- shininess
 - gazebo::common::Material, 437
- Show
 - gazebo::rendering::GUIOverlay, 339
- ShowBoundingBox
 - gazebo::rendering::Visual, 846
- ShowCOM
 - gazebo::rendering::Visual, 846
- ShowCollision
 - gazebo::rendering::Visual, 846
- ShowJoints
 - gazebo::rendering::Visual, 846
- ShowRotation
 - gazebo::rendering::ArrowVisual, 120
 - gazebo::rendering::AxisVisual, 123
- ShowSkeleton
 - gazebo::rendering::Visual, 847
- ShowVisual
 - gazebo::rendering::Light, 397
- ShowWireframe
 - gazebo::rendering::Camera, 170
- Shutdown
 - gazebo::transport::Connection, 213
- sid
 - gazebo::common::NodeTransform, 513
- Signal
 - gazebo::event::EventT, 300–302
- simTime
 - gazebo::physics::State, 705
- SingletonT
 - ~SingletonT, 672
 - Instance, 672
 - SingletonT, 672
 - SingletonT, 672
- SingletonT< T >, 671
- SingletonT.hh, 1029
- size
 - gazebo::math::Plane, 549
- skelAnimation
 - gazebo::physics::Actor, 106
- skelNodesMap
 - gazebo::physics::Actor, 106
- Skeleton
 - gazebo::common::Skeleton, 674
- skeleton
 - gazebo::physics::Actor, 106
- Skeleton.hh, 1029
- SkeletonAnimation
 - gazebo::common::SkeletonAnimation, 680
- SkeletonAnimation.hh, 1031
- SkeletonNode
 - gazebo::common::SkeletonNode, 685, 686
- SkeletonNodeType
 - gazebo::common::SkeletonNode, 685
- skinFile
 - gazebo::physics::Actor, 106
- skinScale
 - gazebo::physics::Actor, 107
- SkyX, 100
- skyx
 - gazebo::rendering::Scene, 646
- Slerp
 - gazebo::math::Quaternion, 593
- SliderJoint
 - gazebo::physics::SliderJoint, 693
- SliderJoint.hh, 1032
- slip1
 - gazebo::physics::SurfaceParams, 725
- slip2
 - gazebo::physics::SurfaceParams, 725
- SnapVisualToNearestBelow

- gazebo::rendering::Scene, 645
- source
 - gazebo::common::NodeTransform, 513
- specular
 - gazebo::common::Material, 437
- SphereShape
 - gazebo::physics::SphereShape, 696
- SphereShape.hh, 1034
- SphereShapePtr
 - gazebo::physics, 89
- Spline
 - gazebo::math::Spline, 698
- Spline.hh, 1035
- Squad
 - gazebo::math::Quaternion, 593
- Stamp
 - Messages, 60
- Start
 - gazebo::common::LogRecord, 426
 - gazebo::common::Timer, 754
- startDelay
 - gazebo::physics::Actor, 107
- StartRead
 - gazebo::transport::Connection, 213
- startTime
 - gazebo::physics::TrajectoryInfo, 761
- State
 - gazebo::physics::State, 703
- State.hh, 1036
- Step
 - gazebo::common::LogPlay, 423
- step
 - gazebo::event::Events, 293
- StepWorld
 - gazebo::physics::World, 863
- Stop
 - gazebo::common::LogRecord, 426
 - gazebo::Master, 428
 - gazebo::physics::Actor, 104
 - gazebo::physics::World, 863
 - gazebo::sensors::SensorManager, 665
 - gazebo::Server, 667
 - gazebo::transport::ConnectionManager, 217
 - gazebo::transport::IOManager, 366
- stop
 - gazebo, 77
 - gazebo::event::Events, 294
 - Sensors, 69
 - Transport, 73
- stop_cfm
 - urdf2gazebo::GazeboExtension, 311
- stop_erp
 - urdf2gazebo::GazeboExtension, 311
- stop_world
 - Classes for physics and dynamics, 44
- stop_worlds
 - Classes for physics and dynamics, 44
- StopAnimation
 - gazebo::physics::Entity, 275
 - gazebo::physics::Model, 473
- StopRead
 - gazebo::transport::Connection, 213
- StrStr_M
 - gazebo::common, 80
- StripSceneName
 - gazebo::rendering::Scene, 645
- StripWorldName
 - gazebo::physics::World, 863
- SubMesh
 - gazebo::common::SubMesh, 709
- SubNodeMap
 - gazebo::transport::TopicManager, 756
- subSampling
 - gazebo::physics::HeightmapShape, 345
- Subscribe
 - gazebo::transport::ConnectionManager, 217
 - gazebo::transport::Node, 503
 - gazebo::transport::TopicManager, 759
- SubscribeOptions
 - gazebo::transport::SubscribeOptions, 717
- SubscribeOptions.hh, 1039
- Subscriber
 - gazebo::transport::Subscriber, 718
- Subscriber.hh, 1040
- SubscriberPtr
 - gazebo::transport, 97
- SubscriptionTransport
 - gazebo::transport::SubscriptionTransport, 720
- SubscriptionTransport.hh, 1042
- SubscriptionTransportPtr
 - gazebo::transport, 97
- SurfaceParams
 - gazebo::physics::SurfaceParams, 722
- SurfaceParams.hh, 1044
- SurfaceParamsPtr
 - gazebo::physics, 89
- SystemPaths.hh, 1045
 - GetCurrentDir, 1047
 - LINUX, 1047
- SystemPlugin
 - gazebo::SystemPlugin, 731
- SystemPluginPtr
 - gazebo, 77
- systemPluginsArgc
 - gazebo::Server, 668
- systemPluginsArgv
 - gazebo::Server, 668

- TPtr
 - gazebo::PluginT, 555
- TRANSLATE
 - gazebo::common::NodeTransform, 510
- TRIANGLES
 - gazebo::common::SubMesh, 709
- TRIFANS
 - gazebo::common::SubMesh, 709
- TRIMESH_SHAPE
 - gazebo::physics::Base, 128
- TRISTRIPS
 - gazebo::common::SubMesh, 709
- tangents
 - gazebo::math::RotationSpline, 628
 - gazebo::math::Spline, 701
- tension
 - gazebo::math::Spline, 701
- texImage
 - gazebo::common::Material, 437
- textureHeight
 - gazebo::rendering::Camera, 173
- textureWidth
 - gazebo::rendering::Camera, 173
- threadPitch
 - gazebo::physics::ScrewJoint, 649
- Time
 - gazebo::common::Time, 735, 736
- time
 - gazebo::common::KeyFrame, 390
 - gazebo::physics::Contact, 221
- Time.hh, 1047
- timePos
 - gazebo::common::Animation, 119
- Timer
 - gazebo::common::Timer, 753
- Timer.hh, 1048
- TimerStart
 - gazebo::common::DiagnosticManager, 248
- TimerStop
 - gazebo::common::DiagnosticManager, 248
- Tostring
 - sdf::Element, 265
 - sdf::SDF, 650
- Toggle
 - gazebo::rendering::Projector, 571
- ToggleShowVisual
 - gazebo::rendering::Light, 397
- ToggleShowWireframe
 - gazebo::rendering::Camera, 170
- ToggleVisible
 - gazebo::rendering::Visual, 847
- TopicManager.hh, 1049
- TrackVisual
 - gazebo::rendering::Camera, 170
- TrackVisualFromSDF
 - Messages, 60
- TrackVisualImpl
 - gazebo::rendering::Camera, 170
 - gazebo::rendering::UserCamera, 782
- trajInfo
 - gazebo::physics::Actor, 107
- trajectories
 - gazebo::physics::Actor, 107
- transform
 - gazebo::common::NodeTransform, 513
 - gazebo::common::SkeletonNode, 692
- TransformAffine
 - gazebo::math::Matrix4, 447
- transformToParentFrame
 - urdf2gazebo::URDF2Gazebo, 774
- TransformType
 - gazebo::common::NodeTransform, 510
- Translate
 - gazebo::rendering::Camera, 171
- translate
 - gazebo::common::PoseKeyFrame, 569
- translated
 - gazebo::physics::TrajectoryInfo, 761
- transparency
 - gazebo::common::Material, 437
- Transport, 70
 - CallbackHelperPtr, 71
 - clear_buffers, 71
 - fini, 71
 - get_master_uri, 71
 - get_topic_namespaces, 72
 - init, 72
 - is_stopped, 72
 - pause_incoming, 72
 - request, 73
 - run, 73
 - stop, 73
- Transport.hh, 1051
- TransportTypes.hh, 1053
- TrimeshShape
 - gazebo::physics::TrimeshShape, 763
- TrimeshShape.hh, 1054
- type
 - gazebo::common::MouseEvent, 485
 - gazebo::common::NodeTransform, 513
 - gazebo::common::SkeletonNode, 692
 - gazebo::physics::TrajectoryInfo, 761
 - gazebo::PluginT, 556
- typeName
 - sdf::Param, 527
- typeString
 - gazebo::rendering::ViewController, 828

- UIntGen
 - gazebo::math, 83
- UNIVERSAL_JOINT
 - gazebo::physics::Base, 128
- UNKNOWN
 - gazebo::common::Image, 350
- URDF2Gazebo
 - urdf2gazebo::URDF2Gazebo, 769
- URRealGen
 - gazebo::math, 83
- Unadvertise
 - gazebo::transport::ConnectionManager, 217
 - gazebo::transport::TopicManager, 760
- UniformIntDist
 - gazebo::math, 83
- UniformRealDist
 - gazebo::math, 83
- UniversalJoint
 - gazebo::physics::UniversalJoint, 766
- UniversalJoint.hh, 1055
- Unsubscribe
 - gazebo::transport::ConnectionManager, 217
 - gazebo::transport::Subscriber, 719
 - gazebo::transport::TopicManager, 760
- Update
 - gazebo::common::PID, 546
 - gazebo::physics::Actor, 104
 - gazebo::physics::Base, 135
 - gazebo::physics::Joint, 379
 - gazebo::physics::JointController, 381
 - gazebo::physics::Link, 414
 - gazebo::physics::Model, 473
 - gazebo::physics::MultiRayShape, 498
 - gazebo::physics::RayShape, 608
 - gazebo::physics::TrimeshShape, 764
 - gazebo::rendering::Camera, 171
 - gazebo::rendering::DynamicLines, 254
 - gazebo::rendering::FPSViewController, 307
 - gazebo::rendering::GUIOverlay, 339
 - gazebo::rendering::MovableText, 491
 - gazebo::rendering::OrbitViewController, 521
 - gazebo::rendering::UserCamera, 782
 - gazebo::rendering::ViewController, 828
 - gazebo::rendering::Visual, 847
 - gazebo::sensors::Sensor, 658
 - gazebo::sensors::SensorManager, 665
 - sdf::Element, 265
 - sdf::Param, 527
 - sdf::ParamT, 530
- UpdateChildrenTransforms
 - gazebo::common::SkeletonNode, 691
- UpdateCollision
 - gazebo::physics::PhysicsEngine, 541
- UpdateFromMsg
 - gazebo::rendering::Light, 397
 - gazebo::rendering::Visual, 847
- updateFunc
 - sdf::Param, 527
- UpdateImpl
 - gazebo::sensors::CameraSensor, 177
 - gazebo::sensors::ContactSensor, 228
 - gazebo::sensors::DepthCameraSensor, 245
 - gazebo::sensors::GpuRaySensor, 328
 - gazebo::sensors::ImuSensor, 356
 - gazebo::sensors::RaySensor, 603
 - gazebo::sensors::RFIDSensor, 616
 - gazebo::sensors::RFIDTag, 618
 - gazebo::sensors::Sensor, 659
- UpdateMass
 - gazebo::physics::Link, 415
- UpdateParameters
 - gazebo::physics::Actor, 105
 - gazebo::physics::Base, 135
 - gazebo::physics::Collision, 189
 - gazebo::physics::Entity, 275
 - gazebo::physics::Inertial, 364
 - gazebo::physics::Joint, 379
 - gazebo::physics::Link, 415
 - gazebo::physics::Model, 473
- updatePeriod
 - gazebo::sensors::Sensor, 660
- UpdatePhysics
 - gazebo::physics::PhysicsEngine, 541
- UpdatePoint
 - gazebo::math::RotationSpline, 627
 - gazebo::math::Spline, 701
- UpdatePublications
 - gazebo::transport::TopicManager, 760
- UpdateRays
 - gazebo::physics::MultiRayShape, 498
- UpdateShaders
 - gazebo::rendering::RTShaderSystem, 632
- UpdateStateSDF
 - gazebo::physics::World, 864
- UpdateSurface
 - gazebo::physics::Link, 415
- urdf2gazebo, 100
- urdf2gazebo::GazeboExtension, 307
 - blobs, 309
 - damping_factor, 309
 - fdir1, 309
 - fudge_factor, 309
 - GazeboExtension, 308
 - gravity, 309
 - initial_joint_position, 309
 - is_damping_factor, 309
 - is_fudge_factor, 309
 - is_initial_joint_position, 309

- is_kd, 309
- is_kp, 309
- is_laser_retro, 309
- is_maxVel, 310
- is_minDepth, 310
- is_mu1, 310
- is_mu2, 310
- is_stop_cfm, 310
- is_stop_erp, 310
- kd, 310
- kp, 310
- laser_retro, 310
- material, 310
- maxVel, 310
- minDepth, 311
- mu1, 311
- mu2, 311
- old_link_name, 311
- provideFeedback, 311
- reduction_transform, 311
- self_collide, 311
- setStaticFlag, 311
- stop_cfm, 311
- stop_erp, 311
- urdf2gazebo::URDF2Gazebo, 766
 - ~URDF2Gazebo, 769
 - addKeyValue, 769
 - addTransform, 769
 - copyPose, 769
 - createCollision, 770
 - createCollisions, 770
 - createGeometry, 770
 - createInertial, 770
 - createJoint, 770
 - createLink, 770
 - createSDF, 770
 - createVisual, 770
 - createVisuals, 770
 - gazebo_extensions_, 774
 - getGeometryBoundingBox, 770
 - getKeyValueAsString, 771
 - initModelDoc, 771
 - initModelFile, 771
 - initModelString, 771
 - insertGazeboExtensionCollision, 771
 - insertGazeboExtensionJoint, 771
 - insertGazeboExtensionLink, 771
 - insertGazeboExtensionRobot, 771
 - insertGazeboExtensionVisual, 771
 - inverseTransformToParentFrame, 771
 - listGazeboExtensions, 771
 - parseGazeboExtension, 771
 - parseVector3, 772
 - printCollisionGroups, 772
 - printMass, 772
 - reduceCollisionToParent, 772
 - reduceCollisionsToParent, 772
 - reduceFixedJoints, 772
 - reduceGazeboExtensionContactSensorFrame-Replace, 772
 - reduceGazeboExtensionFrameReplace, 772
 - reduceGazeboExtensionGripperFrameReplace, 772
 - reduceGazeboExtensionJointFrameReplace, 773
 - reduceGazeboExtensionPluginFrameReplace, 773
 - reduceGazeboExtensionProjectorFrameReplace, 773
 - reduceGazeboExtensionProjectorTransformReduction, 773
 - reduceGazeboExtensionSensorTransformReduction, 773
 - reduceGazeboExtensionToParent, 773
 - reduceGazeboExtensionsTransformReduction, 773
 - reduceInertialToParent, 773
 - reduceJointsToParent, 773
 - reduceVisualToParent, 774
 - reduceVisualsToParent, 773
 - transformToParentFrame, 774
 - URDF2Gazebo, 769
 - values2str, 774
 - vector32str, 774
- UserCamera
 - gazebo::rendering::UserCamera, 777
- UserCamera.hh, 1056
- UserCameraPtr
 - gazebo::rendering, 92
- V_ABOVE
 - gazebo::rendering::MovableText, 488
- V_BELOW
 - gazebo::rendering::MovableText, 488
- VEL
 - gazebo::physics::Joint, 369
- VERTEX
 - gazebo::rendering::RenderEngine, 611
- VISUAL
 - gazebo::physics::Base, 128
- VISUAL_PLUGIN
 - Common, 32
- Valid
 - gazebo::common::Image, 354
- value
 - gazebo::common::NumericKeyFrame, 517
 - sdf::ParamT, 530
- values2str
 - urdf2gazebo::URDF2Gazebo, 774
- variance
 - Math, 50
- Vector2d

- gazebo::math::Vector2d, 784
- Vector2d.hh, 1057
- Vector2i
 - gazebo::math::Vector2i, 792
- Vector2i.hh, 1058
- Vector3
 - gazebo::math::Vector3, 802
- Vector3.hh, 1059
- vector32str
 - urdf2gazebo::URDF2Gazebo, 774
- Vector4
 - gazebo::math::Vector4, 814
- Vector4.hh, 1060
- version
 - sdf::SDF, 650
- VertAlign
 - gazebo::rendering::MovableText, 488
- vertElem
 - gazebo::physics::MultiRayShape, 499
 - gazebo::sensors::GpuRaySensor, 330
- vertHalfAngle
 - gazebo::sensors::GpuRaySensor, 330
- vertRangeCount
 - gazebo::sensors::GpuRaySensor, 330
- vertRayCount
 - gazebo::sensors::GpuRaySensor, 330
- vertSize
 - gazebo::physics::HeightmapShape, 345
- vertexBufferCapacity
 - gazebo::rendering::DynamicRenderable, 258
- vertexIndex
 - gazebo::common::NodeAssignment, 508
- vfov
 - gazebo::sensors::GpuRaySensor, 330
- Video
 - gazebo::common::Video, 822
- Video.hh, 1062
- VideoVisual
 - gazebo::rendering::VideoVisual, 824
- VideoVisual.hh, 1063
- ViewContacts
 - gazebo::rendering::Scene, 646
- viewContacts
 - gazebo::rendering::Events, 282
- ViewController
 - gazebo::rendering::ViewController, 826
- ViewController.hh, 1064
- viewport
 - gazebo::rendering::Camera, 173
- visPub
 - gazebo::physics::Entity, 277
- visitRenderables
 - gazebo::rendering::MovableText, 491
- Visual
 - gazebo::rendering::Visual, 833
- Visual.hh, 1065
- VisualFromSDF
 - Messages, 60
- visualMsg
 - gazebo::physics::Entity, 277
- visualName
 - gazebo::physics::Actor, 107
- VisualPlugin
 - gazebo::VisualPlugin, 848
- VisualPluginPtr
 - gazebo, 77
- VisualPtr
 - gazebo::rendering, 92
 - Gazebo_parser, 65
- visuals
 - gazebo::physics::Link, 415
- w
 - gazebo::math::Quaternion, 594
 - gazebo::math::Vector4, 821
- WORLD_PLUGIN
 - Common, 32
- WaitForConnection
 - gazebo::transport::Publisher, 581
- wallTime
 - gazebo::physics::State, 705
- weight
 - gazebo::common::NodeAssignment, 508
- White
 - gazebo::common::Color, 204
- windowId
 - gazebo::rendering::Camera, 173
- WindowManager.hh, 1066
- WireBox
 - gazebo::rendering::WireBox, 852
- WireBox.hh, 1067
- World
 - gazebo::physics::World, 856
- world
 - gazebo::physics::Base, 135
 - gazebo::physics::PhysicsEngine, 541
 - gazebo::sensors::Sensor, 660
- World.hh, 1068
- worldCreated
 - gazebo::event::Events, 294
- WorldPlugin
 - gazebo::WorldPlugin, 865
- WorldPluginPtr
 - gazebo, 77
- WorldPtr
 - gazebo::physics, 89
- WorldState
 - gazebo::physics::WorldState, 867

WorldState.hh, 1069
worldUpdateEnd
 gazebo::event::Events, 294
worldUpdateStart
 gazebo::event::Events, 294
wrench
 gazebo::physics::Contact, 222
Write
 sdf::SDF, 650

x
 gazebo::math::Quaternion, 594
 gazebo::math::Vector2d, 790
 gazebo::math::Vector2i, 799
 gazebo::math::Vector3, 811
 gazebo::math::Vector4, 821
X_POSITION
 BVHLoader.hh, 883
X_ROTATION
 BVHLoader.hh, 883

y
 gazebo::math::Quaternion, 594
 gazebo::math::Vector2d, 790
 gazebo::math::Vector2i, 799
 gazebo::math::Vector3, 812
 gazebo::math::Vector4, 821
Y_POSITION
 BVHLoader.hh, 883
Y_ROTATION
 BVHLoader.hh, 883
Yellow
 gazebo::common::Color, 204

z
 gazebo::math::Quaternion, 594
 gazebo::math::Vector3, 812
 gazebo::math::Vector4, 821
Z_POSITION
 BVHLoader.hh, 883
Z_ROTATION
 BVHLoader.hh, 883
ZERO
 gazebo::math::Matrix4, 448
Zero
 gazebo::math::Pose, 564
 gazebo::math::Vector3, 812